CLAYTON SILVA OLIVEIRA

CLASSIFICADORES BASEADOS EM VETORES DE SUPORTE GERADOS A PARTIR DE DADOS ROTULADOS E NÃO-ROTULADOS

Dissertação Apresentada à Escola Politécnica da Universidade de São Paulo para Obtenção do Título de Mestre em Engenharia

São Paulo

2006

CLAYTON SILVA OLIVEIRA

CLASSIFICADORES BASEADOS EM VETORES DE SUPORTE GERADOS A PARTIR DE DADOS ROTULADOS E NÃO-ROTULADOS

Dissertação Apresentada à Escola Politécnica da Universidade de São Paulo para Obtenção do Título de Mestre em Engenharia

Área de Concentração: Engenharia de Controle e Automação Mecânica

Orientador:
Prof. Dr.
Fabio G. Cozman

São Paulo

2006

Aos meus pais, Marinalva e José, pela minha formação como pessoa. A Nessa, pela eterna "irmanzade". A Érica, pelo amor. À Escola Politécnica, pela formação como Engenheiro. A Deus, por onde consegui chegar.

Agradecimentos

Agradeço ao meu orientador, Prof. Dr. Fábio Gagliardi Cozman, pelas orientações recebidas durante a condução de meu trabalho de mestrado. Gostaria também de agradecê-lo pelos conselhos de vida e pelas inúmeras ajudas que me deu como amigo. Obrigado, professor!

Agradeço à Prof^a. Dr^a. Leliane Nunes de Barros (IME-USP) e ao Prof. Dr. Emílio Del Moral Hernandez (LSI-USP) pelas sugestões recebidas no Exame de Qualificação e por formarem, juntamente com meu orientador, a banca de avaliação desta Dissertação de Mestrado.

Agradeço à FAPESP pelo apoio financeiro prestado através da Bolsa de Mestrado (Processo n^o 03/09653-5).

Agradeço aos companheiros do Laboratório de Tomada de Decisão do PMR-EPUSP, principalmente a André Saheki, Daniel Kikuti, Jaime Ide e Waldemar Oliveira Jr., por criarem um ambiente agradável e bem-humorado. Quantos almoços e discussões (muitas vezes divergentes) que me permitiram amadurecer um pouco mais! Em particular, Waldemar compartilhou comigo várias horas e dias de experimentos e trocas de idéias no início desse trabalho.

Agradeço à minha família e a todos meus amigos que, de uma forma ou outra, me ajudaram durante e na conclusão desse trabalho de mestrado.

Resumo

Treinamento semi-supervisionado é uma metodologia de aprendizado de máquina que conjuga características de treinamento supervisionado e não-supervisionado. Ela se baseia no uso de *bases semi-rotuladas* (bases contendo dados rotulados e não-rotulados) para o treinamento de classificadores. A adição de dados não-rotulados, mais baratos e geralmente disponíveis em maior quantidade do que os dados rotulados, pode *aumentar a acurácia* e/ou *baratear* o custo de treinamento desses classificadores (a partir da diminuição da quantidade de dados rotulados necessários). Esta dissertação analisa duas estratégias para se executar treinamento semi-supervisionado, especificamente em *Support Vector Machines (SVMs)*.

A estratégia *direta* é atualmente mais conhecida e estudada, e permite o uso de dados rotulados e não-rotulados, *ao mesmo tempo*, em tarefas de aprendizagem de classificadores. Entretanto, a inclusão de muitos dados não-rotulados pode tornar o treinamento demasiadamente lento. Já a estratégia *indireta* é mais recente, sendo capaz de agregar os benefícios do treinamento semi-supervisionado direto com *tempos menores* para o aprendizado de classificadores. Esta opção utiliza os dados não-rotulados para *pré-processar* a base de dados previamente à tarefa de aprendizagem do classificador, permitindo, por exemplo, a filtragem de eventuais ruídos e a reescrita da base em espaços de variáveis mais convenientes.

Dentro do escopo da forma *indireta*, está a principal contribuição dessa dissertação: idealização, implementação e análise do algoritmo *split learning*. Foram obtidos ótimos resultados com esse algoritmo, que se mostrou eficiente em treinar SVMs de melhor acurácia e em períodos menores a partir de bases semi-rotuladas.

Abstract

Semi-supervised learning is a machine learning methodology that mixes features of supervised and unsupervised learning. It allows the use of *partially labeled databases* (databases with labeled and unlabeled data) to train classifiers. The addition of unlabeled data, which are cheaper and generally more available than labeled data, can *enhance the accuracy* and/or *decrease the costs* of learning such classifiers (by diminishing the quantity of required labeled data). This work analyzes two strategies to perform semi-supervised learning, specifically with Support Vector Machines (SVMs).

The *direct* strategy is currently more popular and studied; it allows the use of labeled and unlabeled data, *concomitantly*, in learning classifiers tasks. However, the addition of many unlabeled data can lead to very long training times. The *indirect* strategy is more recent; it is able to attain the advantages of the direct semi-supervised learning with *shorter training times*. This alternative uses the unlabeled data to *pre-process* the database prior to the learning task; it allows denoising and rewriting the data in better feature espaces.

The main contribution of this Master thesis lies within the *indirect* strategy: conceptualization, experimentation, and analysis of the *split learning* algorithm, that can be used to perform indirect semi-supervised learning using SVMs. We have obtained promising empirical results with this algorithm, which is efficient to train better accuracy SVMs in shorter times from partially labeled databases.

Sumário

Lista de Figuras

Capítulo 1

INTRODUÇÃO

> *O ano de 1956 é tomado como sendo o ano de nascimento da inteligência artificial e desde então muitos avanços vêm sendo realizados na área [1].*
>
> *Técnicas de inteligência artificial são empregadas tanto em problemas mais simples, como em complexas tentativas de simular a inteligência humana [2].*
>
> *Além de ser uma atividade de ciência da computação, a inteligência artificial constitui uma área da engenharia, que está preocupada como suas técnicas serão implementadas e empregadas na prática [3].*

O presente trabalho de mestrado está dentro do escopo da *inteligência artificial*; mais especificamente, dentro de um nicho chamado de *aprendizado de máquina* (*machine learning*). O trabalho foca em um tópico atualmente estudado e discutido pela comunidade científica de aprendizado de máquina: *treinamento semi-supervisionado.*

Vale citar que algumas áreas de pesquisa inter-relacionadas como mineração de dados (*data mining*), reconhecimento de padrões (*pattern recognition*) e a própria aprendizado de máquina nem sempre são consideradas como nichos dentro de inteligência artificial. Muitas vezes são consideradas áreas independentes, com terminologias e definições próprias. Dessa forma, algumas nomenclaturas que farão parte dessa dissertação podem ser encontradas em outras literaturas não

relacionadas com inteligência artificial.

Para permitir a discussão deste tópico, algumas definições são necessárias. Define-se um universo U de conhecimento, especificado pelos conjuntos $\mathcal{X}$ e $\mathcal{Y}$. O conjunto $\mathcal{X}$ é formado por todas as combinações x possíveis de N variáveis[1], sendo cada combinação $x \in \Re^N$ denominada de *dado*. Além disso, o conjunto $\mathcal{X}$ é dividido em duas *classes*, $classe_1$ e $classe_2$. A cada x é atribuído um rótulo y, que indica a classe a qual x pertence, ou seja, $y = classe_1$ ou $y = classe_2$, sendo $\mathcal{Y} = \{classe_1, classe_2\}$.

Considere um exemplo ilustrativo. Suponha que o universo U é especificado pelo conhecimento de N variáveis do meio ambiente do planeta Marte, que podem prever se uma tempestade ocorrerá ou não sobre uma determinada região. As variáveis podem ser direção das principais correntes de ar, ocorrência de certos gases e estação do ano. Dessa forma, U é formado por todas as combinações possíveis $x \in \mathcal{X}$ das N variáveis daquele meio ambiente, pela $classe_1$ (tempestade) e $classe_2$ (não-tempestade).

Uma sonda espacial vai até este planeta para colher diariamente as N variáveis, enviando-as para a Terra com a mesma periodicidade. A sonda ficará em Marte por um certo período, e enviará n amostras durante este tempo. Na Terra, um especialista em meteorologia analisa as N variáveis e, por deter parte do universo U (já que ele é um *especialista*), classifica aquele *dia* (ou uma amostra das N variáveis) como *tempestuoso* ou *não*. Entretanto, ele precisa de uma semana de muita computação, cálculos e de muitos funcionários para se chegar a um veredicto dado certo *dia*, tendo-se então um alto custo agregado para se classificar se um certo *dia* é *tempestuoso* ou *não* em Marte. Por isso, os diretores do projeto, para que este seja financeiramente viável, decidem que o especialista e sua equipe classificarão *apenas* 10% das amostras que serão enviadas pela sonda durante o tempo de sua permanência naquele planeta.

Tem-se então o conjunto de amostras rotuladas[2] (ou classificadas) pelo espe-

[1] Outros termos comumente usados na literatura como sinônimos de variáveis são atributos, componentes e *features*.

[2] Em inglês, *labeled*.

cialista $X_{labeled} = X_l \subseteq \mathcal{X}$ (mais custosas e em menor número[3]), e o conjunto de amostras não-rotuladas[4] $X_{unlabeled} = X_{ul} \subseteq \mathcal{X}$ (menos custosas e em maior número). O conjunto de rótulos, ou decisões do especialista[5], é denominado Y_l, e forma com X_l uma *base de dados rotulada* $B_l = \{X_l, Y_l\} = \{x_{l,i}, y_{l,i}\}$. Unindo-se $\{X_l, Y_l\}$ e X_{ul}, será então formada uma *base de dados semi-rotulada* $B_{lul} = \{\{X_l, Y_l\}, X_{ul}\}$.

Após o término do projeto, decide-se enviar uma outra sonda extremamente cara para Marte: uma câmera de altíssima definição para capturar imagens e sons daquele planeta. Como ela é um equipamento muito caro, optou-se por fazer um mecanismo de auto-proteção que, ao detectar que aquele dia será tempestuoso, envolve a câmera em uma blindagem extremamente resistente, que corresponde a um mecanismo de defesa a qualquer tipo de intempérie. No entanto, como não se pode mandar o especialista em meteorologia para Marte, define-se que o mecanismo deverá ser *treinado* para classificar os *dias* de Marte; ou seja, deseja-se que aquele mecanismo tenha uma aproximação de U em sua central de processamento (CPU) para se defender de eventuais tempestades. Aquele mecanismo conterá então um *classificador* $c(.)$, definido da seguinte maneira:

$$c(.) : \mathcal{X} \rightarrow \mathcal{Y}, \tag{1.1}$$

sendo $X \subseteq \mathcal{X}$ um conjunto de amostras, e Y um conjunto de rótulos y_i atribuídos a cada $x_i \in X$ pelo classificador $c(.)$.

Entretanto, o classificador deve passar por um *treinamento*, ou *aprendizado de máquina*. No caso, *treinamento* será uma rotina que apresentará certa base de dados B para que, baseado nestas amostras, o classificador consiga *induzir* [4] uma função $c(.)$.

O mecanismo de defesa da sonda deve então aprender a *relação* que existe entre os valores das variáveis de certa amostra de dia e como estará o clima daquele dia. Entretanto, apenas 10% de B_{lul}, X_l, está rotulada. Nesse caso, o método mais "trivial" de se produzir a função $c(.)$ seria utilizar somente a base rotulada

[3] Vide a Seção 2.1 para maiores explanações.
[4] Em inglês, *unlabeled*.
[5] Por hipótese, as decisões do especialista sempre estão corretas.

$\{X_l, Y_l\}$. Isso caracterizaria *treinamento supervisionado*, ou seja, a utilização das amostras e respectivos rótulos para a aprendizagem[6] de classificadores.

Nesse ponto é que entra a questão do *treinamento semi-supervisionado*:

> *Será vantajoso treinar o classificador da sonda $c(.)$ com 100% da base de dados disponível, ou seja, fazer $B = B_{lul}$?*

Para este trabalho, o classificador a receber treinamento semi-supervisionado será uma *support vector machine (SVM)*[7] (que será apresentada posteriormente), cujo treinamento é originalmente *supervisionado*. Dessa forma, o foco deste trabalho de mestrado é:

> Estudo e desenvolvimento de metodologias que permitam o uso de bases semi-rotuladas para o treinamento de *support vector machines*, de modo que estes classificadores, originalmente supervisionados, se beneficiem da informação adicional contida nas amostras não-rotuladas da base de dados, que não seriam trivialmente consideradas.

1.1 Importância prática dos dados não-rotulados

Alguns motivos relevantes podem ser listados para justificar a pesquisa sobre bases semi-rotuladas para tarefas de classificação.

Os seres humanos parecem executar com habilidade certas tarefas de aprendizado com informações semi-rotuladas, como tarefas de discriminação visual [5]. Crianças parecem aprender a linguagem principalmente ouvindo e imitando, com limitado *supervisionamento* dos pais [6].

Além disso, a tecnologia barateou o armazenamento e a aquisição de informações, seja pela Internet ou por uma câmera fotográfica digital. Tanta informação faz com que a sociedade necessite de classificadores inteligentes que filtrem previamente as informações realmente necessárias e/ou desejadas, com a mínima intervenção de especialistas. Um exemplo patente é o problema dos spams[8]: seria

[6] Os termos treinamento e aprendizagem serão utilizados como sinônimos.
[7] Máquina de vetores de suporte.
[8] *Unsolicited commercial e-mail*, ou mensagem eletrônica comercial não-solicitada.

formidável a existência de um classificador que conseguisse aprender com algumas mensagens eletrônicas rotuladas como 'spam' ou 'não-spam' pelo usuário, e com mais 1 000 000 mensagens não-rotuladas, tornando-se cada vez mais preciso em evitar estas mensagens indesejadas. Outros exemplos da importância de se saber tirar vantagem do uso de dados não-rotulados, pelo fato dos dados rotulados serem extremamente custosos ou pouco disponíveis, estão na pesquisa genética, no mercado financeiro e em pesquisas de mercado consumidor [6].

1.2 Tópicos, objetivos e contribuições

Um dos assuntos importantes desse trabalho surgiu após a realização das revisões bibliográficas, pois constatou-se que atualmente existem duas principais vertentes dentro do tópico de treinamento semi-supervisionado. As duas vertentes foram divididas em: treinamento semi-supervisionado *direto* e treinamento semi-supervisionado *indireto*. A seguinte diferenciação é um dos pilares desse trabalho, e se solidificará ao longo do texto:

1. No treinamento semi-supervisionado *direto* (Seção 2.2), os dados não-rotulados são utilizados *concomitantemente* com os dados rotulados durante o treinamento do classificador;

2. No treinamento semi-supervisionado *indireto* (Seção 2.3), os dados disponíveis (rotulados e não-rotulados) são primeiramente pré-processados. Nesse passo, faz-se uso de toda a informação disponibilizada pelos dados para a resolução de problemas estruturais da base de dados (presença de ruídos, variáveis má escolhidas etc) e/ou para reduzir dimensionalmente a base de dados. Posteriormente, os dados agora pré-processados são utilizados no aprendizado de um classificador, que pode ser supervisionado ou semi-supervisionado.

Existem atualmente alguns métodos que implementam treinamento semi-supervisionado *direto* em SVMs. No entanto, não se tem conhecimento de qualquer metodologia que execute treinamento semi-supervisionado *indireto* para este tipo

de classificador. Esse trabalho contribui com um algoritmo de treinamento semi-supervisionado *indireto* (Capítulo 3) que permite o aprendizado de SVMs.

O trabalho pode ser dividido em duas etapas. Primeiramente, implementou-se um classificador *support vector machine (SVM)* capaz de aprender de forma diretamente semi-supervisionada. Esse classificador recebeu o nome de *LUL-SVM*. Entretanto, como pode ser visto no Apêndice B, a LUL-SVM se mostrou válida porém com uma relação custo/benefício desfavorável em relação a um classificador mais simples, o *classificador de Bayes*. Por isso, em uma segunda etapa, buscou-se um algoritmo alternativo de se utilizar dados não-rotulados para o treinamento de SVMs, a partir de treinamento semi-supervisionado indireto. Esse algoritmo utiliza os dados não-rotulados somente em um primeiro passo de pré-processamento não-supervisionado, e os dados rotulados já pré-processados são considerados em um segundo passo supervisionado. Por essa razão, o método recebeu o nome *split learning*, que poderia ser traduzido como "treinamento dividido" (no sentido de separar o treinamento semi-supervisionado em duas componentes distintas: uma totalmente não-supervisionada e outra totalmente supervisionada).

As duas principais contribuições desse trabalho de mestrado são:

- **Contribuição 1:** Identificação e divisão de duas vertentes dentro do tópico de treinamento semi-supervisionado: *treinamento semi-supervisionado direto* e *treinamento semi-supervisionado indireto*.

- **Contribuição 2:** Implementação e experimentação de um algoritmo de treinamento semi-supervisionado *indireto* (*split learning*) que pode ser utilizado para o aprendizado de SVMs.

1.3 Roteiro da dissertação

A dissertação consiste de sete capítulos, cujos conteúdos são a seguir listados:

- **Capítulo 1:** Introdução aos conceitos de aprendizado de máquina, treinamento semi-supervisionado e bases semi-rotuladas. Este capítulo também fornece os principais tópicos, objetivos e contribuições da dissertação.

- **Capítulo 2:** Formaliza os conceitos de treinamento semi-supervisionado direto e treinamento semi-supervisionado indireto, e compara as diferentes características de ambos os métodos.

- **Capítulo 3:** Apresenta a principal contribuição desse trabalho, que é um algoritmo de treinamento semi-supervisionado indireto denominado *split learning*. Lista alguns algoritmos de extração de variáveis.

- **Capítulo 4:** Introdução e formalização dos conceitos e teorias dos classificadores *support vector machines (SVMs)*, e apresentação de exemplos de SVMs *diretamente* semi-supervisionadas.

- **Capítulo 5:** Experimentação e resultados do método de treinamento semi-supervisionado indireto *split learning* aplicado a SVMs. Comenta alguns resultados obtidos.

- **Capítulo 6:** Discussão dos resultados obtidos listados no Capítulo 5.

- **Capítulo 7:** Conclusões desse trabalho de mestrado.

Além disso, o Apêndice B apresenta a LUL-SVM, bem como experimentos, resultados e conclusões sobre esse método, e o Apêndice C anexa um artigo publicado na 22^{nd} *International Conference on Machine Learning*, que ocorreu em 2005 em Bonn, Alemanha.

Capítulo 2

TREINAMENTO SEMI-SUPERVISIONADO

Este capítulo pretende apresentar o que é a metodologia de treinamento semi-supervisionado (Seção 2.1), bem como dividí-lo em duas grandes vertentes: treinamento semi-supervisionado *direto* (Seção 2.2) e treinamento semi-supervisionado *indireto* (Seção 2.3). Por fim, o capítulo traz algumas comparações entre ambas as vertentes de treinamento semi-supervisionado (Seção 2.4).

2.1 Tópicos introdutórios sobre o treinamento semi-supervisionado

O treinamento semi-supervisionado se situa entre dois "extremos": treinamento supervisionado e treinamento não-supervisionado. A Figura 2.1 ilustra características de ambos os tipos de treinamento. Observe que no treinamento não-supervisionado (painel à esquerda da Figura 2.1) os dados não possuem rótulos, e portanto não existe nenhum tipo de diferenciação explícita entre eles. No caso da figura, o classificador não-supervisionado está segregando as amostras de acordo com a proximidade espacial. No treinamento supervisionado (painel à direita), os dados estão diferenciados explicitamente de acordo com os rótulos 'o' e '×'; cabe então ao classificador utilizar os rótulos como "supervisores" para

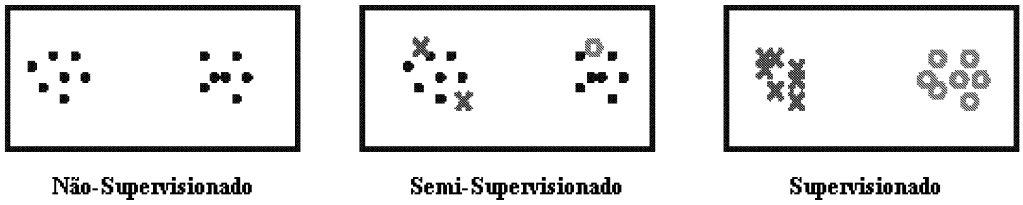

Figura 2.1. Ilustração de treinamento não-supervisionado, treinamento semi-supervisionado e treinamento supervisionado.

a geração do modelo que explique a base de dados. Por fim, no treinamento semi-supervisionado (painel central) existem dados rotulados e não-rotulados, e o classificador deve utilizar ambos os tipos de dados para gerar aprendizado.

O *treinamento supervisionado* utiliza bases rotuladas (amostra X_l e seus respectivos rótulos Y_l) para a aprendizagem de classificadores. Esses classificadores modelarão as relações existentes entre as amostras tendo os respectivos rótulos como "supervisores", isto é, os rótulos servirão como guias de verificação de quão correto está o modelo aprendido. Algumas máquinas classificadoras de treinamento supervisionado são as SVMs (Capítulo 4) e as redes neurais artificiais (RNAs) [7, 8, 9, 10]. Como exemplo, o algoritmo de treinamento das RNAs não pára de iterar até que a RNA esteja classificando as amostras da base de dados conforme os respectivos rótulos, dentro de uma certa margem de tolerância.

Já o *treinamento não-supervisionado*[1] utiliza somente dados não-rotulados (amostra X_{ul}) durante o treinamento de classificadores. Como nesse tipo de treinamento o algoritmo não tem "supervisores", a idéia é agrupar dados da base que supostamente possuam características comuns como, por exemplo, proximidade espacial. Exemplos de algoritmos de treinamento não-supervisionado são mistura de gaussianas (MoG) [11], análise de componentes principais (PCA) (Seção 3.2.2) e mapas auto-organizáveis [12].

A idéia de se treinar um classificador de forma semi-supervisionada é fazer uso tanto dos dados rotulados quanto dos dados não-rotulados, a fim de que o classificador se beneficie dessa maior quantidade de dados e, portanto, apresente

[1] Um termo muito utilizado para designar esse tipo de aprendizagem é *clustering* (que em português pode ser traduzido como agrupando).

melhor acurácia quando da classificação de novos dados.

Os dados rotulados são em geral mais caros, pois freqüentemente como exemplificado na Introdução, os rótulos desses dados são definidos por um especialista, que representa um custo inexistente na captação de dados não-rotulados. Dessa forma, por serem mais baratos, os dados não-rotulados geralmente estão disponíveis em maiores quantidades do que os dados rotulados. Entretanto, na maioria dos casos os dados rotulados são mais valiosos, em termos de informação disponibilizada sobre o problema a ser aprendido, do que os dados sem rótulo [13, 14].

A idéia de treinamento semi-supervisionado se iniciou com Vapnik [4], e tem sido extensivamente pesquisada na esperança de se construir um algoritmo que permita o uso de dados não-rotulados juntamente com dados rotulados. Entretanto, teorias sobre esta modalidade de aprendizado de máquina ainda são escassas na literatura científica. Há trabalhos que argumentam em favor do uso de dados não-rotulados em tarefas de classificação [13, 14, 15, 16, 17, 18], e trabalhos nos quais dados não-rotulados trouxeram prejuízo ao classificador [19, 20].

Mas, afinal, como saber se os dados não-rotulados trarão benefício ou prejuízo ao classificador? Um estudo sobre esta questão pode ser encontrado em [21]:

- Os dados não-rotulados parecem definitivamente beneficiar o classificador quando os dados rotulados disponíveis são poucos para se construir um classificador de acurácia razoável, *e* as hipóteses sobre o *modelo* do qual foram amostrados os dados estão *corretas*. Por exemplo, os dados de uma certa tarefa de aprendizado são amostras de uma mistura de gaussianas; caso tente-se fazer treinamento semi-supervisionado aprendendo uma *mistura de gaussianas*, provavelmente essa tarefa terá sucesso.

- Os dados não rotulados podem beneficiar ou prejudicar a acurácia final do classificador quando a quantidade de dados rotulados já é suficiente para se aprender um classificador de acurácia razoável, *ou* quando as hipóteses sobre o modelo da base de dados estão erradas.

Assim, o treinamento semi-supervisionado se torna interessante somente quan-

do existem poucos dados rotulados (mais caros e com mais informação) frente à quantidade de dados não-rotulados (mais baratos e com menos informação), e quando esses poucos dados rotulados disponíveis não são suficientes para se treinar um bom classificador; essa dissertação foca exatamente nesses casos. No entanto, não existe na literatura a definição de um limiar de quão menor deve ser a quantidade de dados rotulados em relação à quantidade de dados não-rotulados; isso pode depender de fatores como método de treinamento selecionado, características da base de dados e classificador.

Esse trabalho argumenta que os métodos atuais que implementam treinamento semi-supervisionado podem ser divididos em dois grandes grupos (Contribuição 1, segundo a Introdução): treinamento semi-supervisionado *direto* e *indireto*. Essa foi uma idéia obtida a partir de uma constatação extraída das revisões bibliográficas realizadas. Ambos os tipos de aprendizagem semi-supervisionada serão apresentados nas Seções 2.2 e 2.3.

2.2 Treinamento semi-supervisionado direto

Pretende-se implementar um classificador $c(.)$. Primeiramente, suponha que o classificador aproximado será treinado de forma supervisionada; assim, o classificador pode ser representado como:

$$\tilde{y} = c(x, B_l), \tag{2.1}$$

na qual $\tilde{y}$ é um rótulo dado x e $B_l = \{X_l, Y_l\}$ é a base rotulada.

No treinamento semi-supervisionado direto o classificador será treinado *concomitantemente* com os dados não-rotulados, o que faz com que o classificador diretamente semi-supervisionado $c_{direto} = c_{dir}$ seja representado da seguinte maneira:

$$\tilde{y} = c_{dir}(x, B_{lul}), \tag{2.2}$$

na qual $B_{lul} = \{\{X_l, Y_l\}, X_{ul}\}$ é a base semi-rotulada.

Até o momento, não se tem uma metodologia única e sempre eficaz para a implementação de treinamento semi-supervisionado direto em tarefas de apren-

dizado de máquina [6, 21]. Formulações gerativas[2] de classificadores (como redes bayesianas [3]) geralmente conseguem tirar proveito de dados não-rotulados durante o treinamento semi-supervisionado direto [6]. Em formulações estritamente discriminativas (caso das *support vector machines*), há casos de sucessos na utilização de dados não-rotulados [22, 23], embora alguns trabalhos defendam o contrário [24]. De qualquer forma, pode-se citar um número crescente de algoritmos bem-sucedidos em utilizar dados não-rotulados a partir de treinamento semi-supervisionado direto [25, 26, 27, 28, 29, 30, 31].

2.3 Treinamento semi-supervisionado indireto

O *treinamento semi-supervisionado indireto* representa um tipo de implementação de treinamento semi-supervisionado que ainda foi pouco explorado pela comunidade científica. Por isso, essa opção de treinamento de classificadores possui pouca literatura disponível [32, 33].

O aprendizado semi-supervisionado indireto pode ser dividido em dois momentos:

- Em um primeiro momento, os dados disponíveis (rotulados e não-rotulados) são pré-processados. Nesse passo, faz-se uso dos dados para a resolução de problemas estruturais da base (presença de ruídos, variáveis má escolhidas etc) e/ou para reduzir dimensionalmente a base de dados. Por exemplo, esse primeiro passo pode permitir que a base de dados seja reescrita em um espaço de variáveis mais conveniente, que melhor evidencie características do problema a ser aprendido. A idéia é que os dados sejam melhor condicionados para o próximo passo que é o treinamento do classificador.

- Posteriormente, os dados agora melhor condicionados (pré-processados) são utilizados no aprendizado de um classificador, que pode ser supervisionado ou semi-supervisionado.

[2]Formulações *gerativas* são aquelas que estimam a distribuição $p(y, x)$, enquanto que formulações *discriminativas* são as que não se situam dentro da definição de *gerativa*.

O passo de pré-processamento é dado pela seguinte equação:

$$\{X'_l, X'_{ul}\} = q(B_{lul}) = q(\{\{X_l, Y_l\}, X_{ul}\}), \tag{2.3}$$

sendo X'_l e X'_{ul} as amostras rotuladas e não-rotuladas, respectivamente, pré-processadas. Observe então que os dados pré-processados estão em função dos dados não-rotulados X_{ul}.

O classificador $c_{indireto} = c_{ind}$ treinado a partir de treinamento semi-supervisionado indireto pode ser equacionado como segue:

$$\tilde{y} = c_{ind}(x, B'), \tag{2.4}$$

na qual B' é a base de dados pré-processada a ser utilizada para aprendizagem do classificador, e pode ser $B' = B'_l = \{X'_l, Y_l\}$ ou $B' = B'_{lul} = \{\{X'_l, Y_l\}, X'_{ul}\}$, já que c_{ind} pode ser supervisionado ou semi-supervisionado, respectivamente. Note que em $B'_l = \{X'_l, Y_l\}$ os rótulos Y_l não são reescritos.

Dessa forma, o termo *indireto* é utilizado nesse trabalho para designar algoritmos de treinamento semi-supervisionado cujo primeiro passo garante que qualquer amostra da base de dados pré-processada estará em função de *toda* a base de dados sem pré-processamento. Assim, os dados rotulados pré-processados estarão em função dos dados não-rotulados; mesmo que o segundo passo utilize um classificador supervisionado (isto é, descarte os dados não-rotulados), os dados rotulados garantirão que esse classificador seja *indiretamente* condicionado aos dados não-rotulados disponíveis.

Pode-se referenciar os trabalhos [32, 33] como exemplos de métodos bem-sucedidos de treinamento semi-supervisionado indireto. Entretanto, atualmente não existe nenhuma metodologia que implemente treinamento semi-supervisionado indireto em *support vector machines* (Capítulo 4).

2.4 Comparações entre treinamento semi-supervisionado direto e indireto

Primeiramente, comparando-se as Equações (2.4) e (2.2), nota-se que ambas são similares no sentido de que incorporam informação dos dados não-rotulados para

construir os classificadores $c_{ind}(.)$ e $c_{dir}(.)$, respectivamente. Por isso mesmo, considera-se nesse trabalho que ambas as equações estão dentro do conjunto de algoritmos de treinamento semi-supervisionado.

Uma das vantagens mais evidentes da modalidade indireta é o fato dessa executar um passo de pré-processamento com os dados. Nesse passo, há a possibilidade de detecção de ruídos e erros na seleção das variáveis para descrever o problema (e portanto erros na base de dados que o representa), e redefinição dos dados em espaços de variáveis melhores e reduzidos. Todas essas características fazem com que o treinamento semi-supervisionado indireto seja uma maneira mais organizada de se fazer aprendizagem: antes de se aprender propriamente o classificador, tenta-se primeiro resolver problemas e detectar possíveis aproveitamentos de informações fornecidas por toda a massa de dados (rotulados ou não) em prol de uma tarefa de aprendizagem mais eficiente e ágil. Além disso, após a execução desse passo preliminar, as análises e melhorias feitas sobre a base de dados podem ser aproveitadas por qualquer classificador em um segundo passo; de outra forma, pode-se dizer que a fase de análise é *intercambiável*.

Na modalidade direta de treinamento semi-supervisionado toda a massa de dados é utilizada concomitantemente durante o treinamento do classificador. Assim, perde-se uma valiosa oportunidade de se fazer uma análise mais apurada das informações disponibilizadas pela base de dados. Por exemplo, talvez a informação que permitiria reduzir dimensionalmente a base de dados foi simplesmente descartada durante o treinamento do classificador, que não tem a função específica de reduzir dimensões. Isso faz com que muitas vezes algoritmos de treinamento semi-supervisionado direto sejam custosos do ponto de vista computacional, pois além de processarem dados rotulados e não-rotulados ao mesmo tempo, ainda não oferecem uma opção evidente de se reduzir o tamanho dos dados. A utilização de bases de dados de grande porte em treinamento semi-supervisionado direto pode então se tornar inviável, pois o classificador teria de lidar com muitos dados de muitas variáveis.

Um outro tópico de comparação entre treinamento semi-supervisionado direto e indireto é que o segundo pode treinar classificadores que não necessariamente

sejam semi-supervisionados, enquanto que o primeiro requer a disponibilidade de um classificador que possa ser treinado de forma semi-supervisionada ou cujo algoritmo possa ser adaptado para ser treinado dessa maneira, o que nem sempre é tarefa fácil [34].

No próximo capítulo, tem-se a principal contribuição desse trabalho que é o desenvolvimento de uma nova metodologia de treinamento semi-supervisionado indireto, denominada *split learning*, que pode ser utilizada para treinar SVMs a partir de bases semi-rotuladas.

Note que nos primeiros meses desse trabalho de mestrado um novo algoritmo de treinamento semi-supervisionado direto, denominado LUL-SVM, também foi desenvolvido e experimentado. Entretanto, o algoritmo *split learning* se mostrou mais promissor em termos de conceitos e resultados e por essa razão atraiu o foco da dissertação. De qualquer forma, o desenvolvimento do LUL-SVM é apresentado no Apêndice B.

Capítulo 3

NOVO MÉTODO PARA TREINAMENTO SEMI-SUPERVISIONADO INDIRETO: *SPLIT LEARNING*

O presente capítulo apresenta uma das contribuições do trabalho (Contribuição 2 segundo a Introdução), que é uma nova metodologia de treinamento semi-supervisionado indireto denominada de *split learning* (Seção 3.1). Em seguida (Seção 3.2) serão listados alguns métodos de reescrita de variáveis que, como será visto a seguir, podem ser utilizados para execução do passo de pré-processamento (primeiro passo) do *split learning*.

É relevante antecipar que o método *split learning* é composto de um segundo passo que é o treinamento do classificador. No caso desse trabalho o classificador será uma *support vector machine* (SVM), cuja complexidade e o fato de representar um tema central do trabalho fazem necessária sua apresentação em capítulo separado (Capítulo 4).

3.1 *Split learning*

Esse trabalho de mestrado contribui com uma metodologia de treinamento semi-supervisionado indireto denominada *split learning*[1] [34], que pode ser utilizada para o treinamento de SVMs (Capítulo 4).

Como o *split learning* é um método de treinamento semi-supervisionado indireto, seu primeiro passo é o pré-processamento da base de dados. Em específico, o primeiro passo do *split learning* é a reescrita da base de dados em um novo espaço de variáveis, supostamente mais conveniente. Posteriormente, os dados rotulados são reescritos nesse novo espaço de variáveis, e fornecidos para o treinamento de um classificador *supervisionado*. A idéia é utilizar uma estratégia comum em aprendizado de máquina: a forma como a base de dados é escrita em seu espaço de variáveis influencia a acurácia final do classificador [12]. Assim, pretende-se *reescrever* X_l em um novo espaço de variáveis mais conveniente, fazendo-se uso de alguma metodologia de extração de variáveis (Seção 3.2), cujo treinamento é *não-supervisionado*.

Dado $x \in \mathcal{X}$, este pode ser reescrito para x' da seguinte forma:

$$x' = g(x), \tag{3.1}$$

na qual $g(.)$ é a função que representa a mudança de variáveis exercida pelo método de extração de variáveis aprendido durante o pré-processamento dos dados. Entretanto, considera-se $g = g(., X_l, X_{ul})$, pois o método de extração de variáveis é não-supervisionado e pode utilizar dados rotulados e não-rotulados durante seu treinamento. Assim, pode-se reescrever a Equação (3.1) para:

$$x' = g(x, X_l, X_{ul}). \tag{3.2}$$

Substituindo-se a Equação (3.2) na Equação (2.1), chega-se à seguinte expressão:

$$\tilde{y} = c_{split} = c_{spl}(x', B'_l) \Rightarrow \tilde{y} = c_{spl}(g(x, X_l, X_{ul}), \{g(X_l, X_l, X_{ul}), Y_l\}) \tag{3.3}$$

[1] O artigo [34], sobre *split learning*, foi publicado na 22^{nd} *International Conference on Machine Learning* (ICML-2005) – *Learning with Partially Classified Training Data Workshop* pelo autor e outros pesquisadores, e encontra-se anexado no Apêndice C.

Algoritmo 1: *Split Learning*

Entrada: Base de dados rotulada $\{X_l, Y_l\}$, base de dados não-rotulada X_{ul}

Saída: Classificador $c_{spl}(.)$ aprendido utilizando-se $\{X'_l, Y_l\}$

01. Encontrar $g(.)$ a partir de X_l and X_{ul}

02. Reescrever X_l para X'_l através de $g(.)$

03. Aprender o classificador $c_{spl}(.)$ a partir de $\{X'_l, Y_l\}$

Figura 3.1. Procedimentos para execução de *split learning*.

A partir da Equação (3.2), verifica-se que qualquer dado pré-processado (reescrito) estará em função de todos os dados disponíveis, o que faz com que os dados rotulados reescritos fiquem em função dos dados não-rotulados. Isso caracteriza o treinamento semi-supervisionado indireto, pois qualquer classificador que for treinado, mesmo que somente com os dados rotulados, estará processando *indiretamente* informações dos dados não-rotulados (Equação 3.3).

A Figura 3.1 esquematiza o algoritmo *split learning*.

Dado que *split* quer dizer "separar" em inglês, o termo *split learning* indica a principal característica do método: dividir o treinamento semi-supervisionado em uma componente totalmente não-supervisionada e outra totalmente supervisionada.

Espera-se que com a inclusão de X_{ul} os métodos de extração de variáveis tenham a disposição uma quantidade razoável de dados para computar $g(.)$, e então X_l seja melhor reescrito em um espaço de variáveis mais conveniente. Posteriormente, X'_l será utilizado para treinar o classificador supervisionado $c_{spl}(.)$. Esse classificador então considerará *indiretamente* os dados não-rotulados para modelar a relação $\tilde{y} = c_{spl}(x')$, existindo então a possibilidade de que $c_{spl}(.)$ (Equação 3.3) seja melhor condicionado em um espaço de variáveis mais conveniente para X_l.

Dentro do escopo desse trabalho, $g(.)$ e $c_{spl}(.)$ serão implementados pelos métodos *kernel principal component analysis* (apresentado na próxima seção) e *support vector machine* (apresentado no próximo capítulo), respectivamente. Entretanto,

note que $g(.)$ e $c_{spl}(.)$ poderiam ser implementados por inúmeros outros métodos[2], o que reforça o fato de $g(.)$ ser intercambiável com qualquer que seja o modelo adotado para $c_{spl}(.)$.

A seguir, apresentamos dois exemplos de como se implementar a função $g(.)$; entre eles, está o método *kernel principal component analysis*.

3.2 Metodologias para a reescrita de bases

As metodologias que serão a seguir apresentadas se destinam a reescrever um conjunto de vetores X em um espaço de variáveis mais conveniente e mais compacto, seguindo certos critérios. Além disso, seguindo-se a idéia de treinamento semi-supervisionado indireto, os algoritmos de extração de variáveis funcionam como uma "ponte" entre os dados não-rotulados e o classificador.

O método chamado de *Riemannian manifolds* será apresentado neste capítulo como ilustração da idéia de como se utilizar dados não-rotulados de forma indireta em classificadores supervisionados. Já o método *kernel principal component analysis* será considerado para a implementação do *split learning* em SVMs (Capítulo 5).

3.2.1 *Riemannian manifolds*

Esta seção exemplifica como um método de extração de variáveis é utilizado para se implementar treinamento semi-supervisionado indireto. A base teórica e resultados sobre este método de se implementar treinamento semi-supervisionado indireto podem ser encontrados em [32].

Considera-se que $p(x)$ está contido em um *manifold*[3] de poucas dimensões em $\Re^N$, sendo N a dimensão original da base de dados. Assim, os dados não-rotulados podem ser utilizados para se estimar o *manifold*, e posteriormente o classificador pode ser treinado a partir da base de dados reescrita segundo aquele *manifold*.

[2]Por exemplo, $c_{spl}(.)$ poderia ser implementado por redes neurais artificiais.
[3]Sub-espaços não-lineares inscritos em um domínio $\Re^d$ de alta dimensão d.

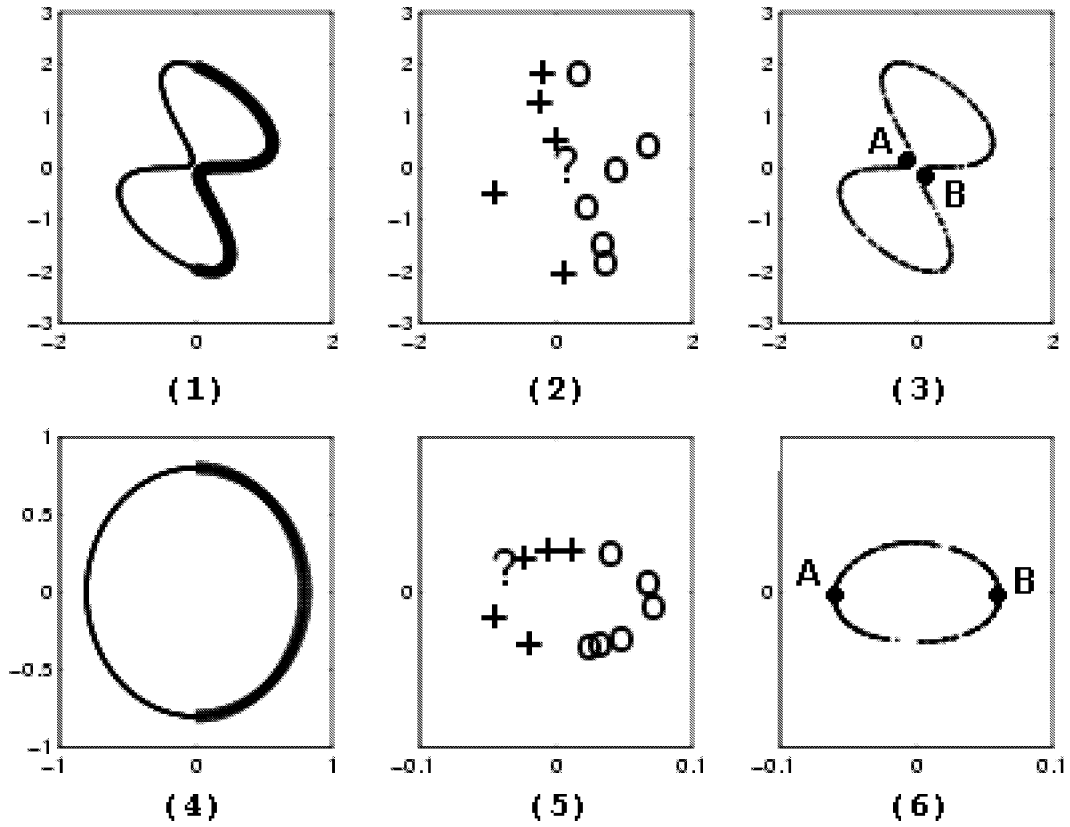

Figura 3.2. Incorporando dados não-rotulados através de *Riemannian manifolds* para tarefas de aprendizagem.

A Figura 3.2 (figura baseada em [32]) ilustra a motivação do porquê reescrever a base de dados em uma estrutura de *manifold*. O painel (1) ilustra um problema dividido em duas classes (a classe formada pelo conjunto de dados que descrevem a curva de linha fina e a classe formada pelos dados que descrevem a linha grossa). O painel (2) apresenta uma tarefa de aprendizado supervisionado, com a missão de aprender o classificador a partir dos dados de rótulos 'o' e '+'. Verifica-se que a pequena quantidade disponível de dados rotulados não é suficiente para revelar a estrutura do problema ilustrada pelo painel (1), o que dificulta a classificação do dado '?'. O painel (3) ilustra uma grande disponibilidade de dados não-rotulados. Os painéis (4), (5) e (6) repetem as idéias dos painéis (1), (2) e (3), respectivamente; porém os dados estão descritos em diferentes sistemas de coordenadas.

Ainda sobre a Figura 3.2, algumas observações podem ser feitas:

1. Observando-se o painel (2) da Figura 3.2, verifica-se que não se pode garantir a correta classificação do ponto '?' somente com os dados rotulados. Porém, o problema torna-se mais fácil dados os pontos não-rotulados do painel (3);

2. Como os dados formam um *manifold* "latente" no painel (3), observa-se que possivelmente é mais conveniente descrever os dados baseando-se em distâncias geodésicas (intracurva) do que utilizar as distâncias euclidianas do plano $\Re^2$. Por exemplo, o ponto A e B estão próximos em termos de distância euclidiana, mas distantes em termos de distância geodésica. Dessa forma, construir classificadores baseados na curva que naturalmente os dados formam pode ser uma boa alternativa ao aprendizado do problema descrito por coordenadas em $\Re^2$;

3. Mesmo os dados sugerindo uma estrutura "latente", o problema ainda não pode ser considerado trivial pois as duas partes da curva se aproximam confusamente uma da outra. Assim, seria preferível uma representação dos dados que transmitisse o fato de que sua estrutura é uma curva fechada. Mais especificamente, seria desejável uma representação na qual as coordenadas variassem o menos possível conforme esta fosse percorrida, exemplificada pelo painel (4). Nota-se que ambas as representações (painéis 1 e 4) ilustram a mesma estrutura, mas em diferentes sistemas de coordenadas. O painel (5) mostra as coordenadas dos dados rotulados no novo espaço de variáveis (base reescrita). Verifica-se que o dado '?' agora está claramente entre os dados de rótulo '+'.

Dessa forma, aprender classificadores no espaço de variáveis mais conveniente pode trazer benefícios à acurácia final de classificação. Para se descobrir o *manifold* mais conveniente, pode-se usar os dados não-rotulados (têm-se então mais dados disponíveis, juntamente com os dados rotulados). Além disso, considerando-se uma alta dimensão N, os dados escritos em seu espaço de variáveis original tornam o aprendizado de classificadores computacionalmente custoso. Nesse caso,

pode-se explorar um *manifold* M de baixa dimensão (compacto), no qual $x_i \in M$. Assim, os dados rotulados estarão escritos em um espaço de variáveis mais conveniente e compacto, favorecendo o classificador em termos de acurácia (melhores variáveis) e tempo de treinamento (menos variáveis).

3.2.2 *Kernel principal component analysis* (kPCA)

O método *principal component analysis (PCA)*[4] [12] é um caso particular do kPCA, como será evidenciado adiante. Para facilitar o entendimento do kPCA, pode-se iniciar pelo equacionamento do PCA, cujo objetivo pode ser descrito da seguinte maneira:

> Dada a base $[X_o]_{N\times n}$ de média zero, na qual $\{x_{o,i}\}_{i=1}^{n}$ é a i-ésima coluna de X_o e representa uma instância de X_o, e $\{X_{o,k}\}_{k=1}^{N}$ é a k-ésima variável de X_o, encontre uma matriz T que faça a transformação $T : X_o \rightarrow X'_o$, ou seja, $X'_o = TX_o$, tal que as variáveis de X'_o sejam *independentes* e, em conseqüência, a matriz de covariância de X'_o, $Cov(X'_o)$, seja uma matriz *diagonal* (variáveis não-correlacionadas).

O PCA assume *linearidade* do problema, de forma que encontrar $Cov(X'_o)$ diagonal pode ser vista como um problema de *mudança de base*. Uma outra suposição importante que o PCA adota é que a média e a variância de X_o são *estatisticamente suficientes*, ou seja, o PCA assume que estes parâmetros descrevem inteiramente a distribuição $p(X_o)$. Entretanto, a única distribuição que pode ser completamente descrita por estes parâmetros é a *distribuição gaussiana*, e desvios desta condição podem invalidar o uso do PCA. Esta suposição pode ser considerada a idéia-chave do PCA, pois um caso importante no qual *não-correlação* ($Cov(X_o)$ é diagonal) e *independência estatística* ($p(X_o) = p_1(X_{o,1})\ldots p_N(X_{o,N})$) são equivalentes ocorre quando $p(X_o)$ é dada por uma *distribuição gaussiana*.

O PCA também adota que grandes variâncias representam direções importantes na distribuição espacial de X_o em $\Re^N$, e que pequenas variâncias indicam

[4] Em português, análise de componentes principais.

ruído nos dados [35]. Para facilitar a solução do PCA, é assumido que as *componentes principais (PCs)* são *ortogonais*, de forma que técnicas de decomposição de álgebra linear podem ser utilizadas.

Essas suposições fazem com que a solução do PCA seja simples: as componentes principais (PCs) da base X_o são definidas pelos *auto-vetores* v_j da matriz $Cov(X_o)$; estes vetores são as linhas da matriz de transformação T. A cada componente principal, ou *auto-vetor* v_j, é associado um *auto-valor*, Ω_j, cuja magnitude fornece uma medida da contribuição dos respectivos auto-vetores para a variância total de X_o [35]. Assim, para se *reduzir a dimensão* de X_o seleciona-se os M auto-vetores associados aos M maiores auto-valores, sendo então M a nova dimensionalidade (ou o novo número de variáveis) da base de dados.

As características e suposições do PCA fazem dele um ótimo método para redefinição da base de dados em melhores espaços de variáveis, cuja eficiência é reconhecida pela comunidade de aprendizado de máquina. Além disso, como o PCA ordena a "importância" das PCs, procede-se a redução dimensional da base de dados com certa facilidade, pois basta descartar as componentes de menor peso na variância de X_o. Aliás, essa redução dimensional, como supostamente está retirando ruído da base de dados, faz do PCA um algoritmo de pré-processamento e tratamento de dados.

Matematicamente, a matriz de covariância é definida da seguinte maneira:

$$Cov(X_o) = \frac{1}{n}\sum_{i=1}^{n} x_{o,i}x_{o,i}^T \tag{3.4}$$

Os PCs (ou v_j) de X_o são determinados a partir do seguinte problema de auto-valor/auto-vetor:

$$\Omega_j v_j = Cov(X_o)v_j \tag{3.5}$$

A Figura 3.3 apresenta uma ilustração da execução do PCA sobre uma base de dados de média zero. A "nuvem" de dados está originalmente escrita em relação às componentes $Eixo_1$ e $Eixo_2 \perp Eixo_1$. Com a execução do PCA, é determinada a primeira componente principal PC_1, e seu respectivo auto-valor Ω_1, e a segunda componente principal $PC_2 \perp PC_1$ e seu respectivo auto-valor

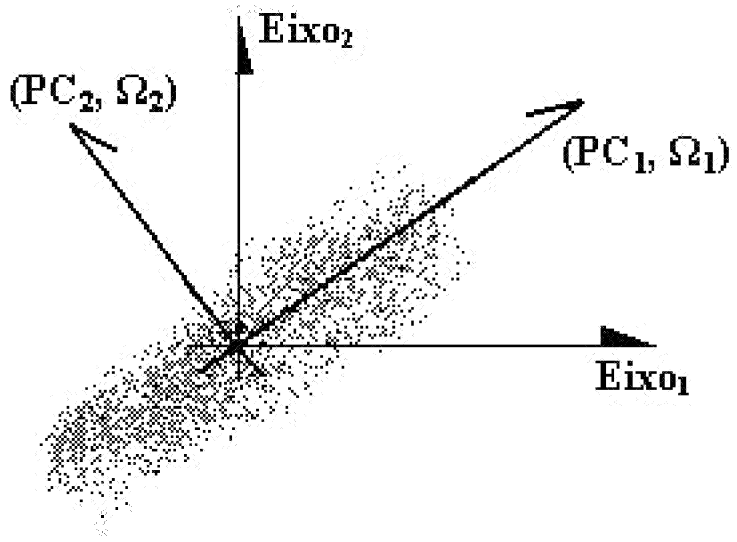

Figura 3.3. Execução do método *principal component analysis* (PCA) sobre uma base de dados de média zero.

$\Omega_2 < \Omega_1$. A direção de PC_1 é igual a direção da maior variância da "nuvem" de dados, ou seja, uma "direção importante" segundo o PCA. Nesse caso, para se proceder redução dimensional de duas ($Eixo_1$ e $Eixo_2$) para uma componente (PC_1), basta descartar a componente de menor peso PC_2.

Como visto anteriormente, o PCA assume linearidade do problema. Entretanto, em alguns casos a transformação de $Cov(X_o)$ (não necessariamente uma matriz diagonal) para $Cov(X'_o)$ (matriz diagional) não pode ser encontrada assumindo-se linearidade, e então o PCA falha. É possível então transformar o PCA em um método não-linear; para isso, definem-se *funções de base* não-lineares $h(.) : x_{o,i} \rightarrow h(x_{o,i}) = (h_1(x_{o,i}), \ldots, h_m(x_{o,i}))$ e $h(x_{o,i}) \in \Re^m$, de tal forma que $x_{o,i}$ seja levado de um espaço linear a um espaço não-linear. Dessa forma, a nova matriz de covariância será assim definida:

$$Cov'(X_o) = \frac{1}{n} \sum_{i=1}^{n} h(x_{o,i}) h(x_{o,i})^T, \tag{3.6}$$

na qual Cov' é a matriz de covariância no espaço não-linear.

O uso de $h(x)$ é necessário somente através de produtos internos. De fato, nenhum tipo de especificação da transformação $h(x)$ é necessária; basta a definição

de uma função *kernel*[5] $K(.,.)$. Portanto, definindo-se a função *kernel* como sendo o seguinte produto interno entre dois vetores $x_{o,i}$ e $x_{o,i'}$:

$$K(x_{o,i}, x_{o,i'}) = \langle h(x_{o,i}), h(x_{o,i'}) \rangle, \tag{3.7}$$

a projeção de $x_{o,i}$, no espaço não-linear dado por $h(x_{o,i})$, será dado pela seguinte equação:

$$v'_j.h(x_{o,i}) = \sum_{i=1}^{n} \alpha_{j,i} K(x_{o,i'}, x_{o,i}), \tag{3.8}$$

na qual v'_j é um PC no espaço não-linear, e $\Omega_j(\alpha_j.\alpha_j) = 1$.

Quatro tipos de função *kernel* são os mais utilizados:

- linear: $K(x_{o,i}, x_{o,i'}) = x_{o,i}^T x_{o,i'}$ (não modifica o espaço de variáveis);
- polinomial de grau d: $K(x_{o,i}, x_{o,i'}) = (1 + \langle x_{o,i}, x_{o,i'} \rangle)^d$;
- funções de base radial: $K(x_{o,i}, x_{o,i'}) = e^{-||x_{o,i} - x_{o,i'}||^2 / 2\sigma^2}$;
- rede neural: $K(x_{o,i}, x_{o,i'}) = \tanh(\kappa_1 \langle x_{o,i}, x_{o,i'} \rangle + \kappa_2)$.

As Equações (3.6), (3.7) e (3.8) conjugam uma metodologia PCA não-linear chamada de *kernel principal component analysis - kPCA* [36]. O kPCA é modular no sentido de que a escolha de uma certa função *kernel* $K(.,.)$ muda o comportamento do kPCA a ser implementado (*kernel-based*). Por exemplo, a escolha da função *kernel* linear ($K(x_{o,i}, x_{o,i'}) = \langle x_{o,i}, x_{o,i'} \rangle$) transforma o kPCA em PCA (o que evidencia o fato do PCA ser um caso particular de kPCA). Com o algoritmo kPCA, pode-se resolver problemas que não obedeçam a hipótese de linearidade do PCA linear, nos quais o PCA pode falhar. Dessa forma, o kPCA alia as boas características do PCA como método de redefinição de melhores espaços de variáveis e de tratamento de dados com algoritmos não-lineares.

Por ser mais geral e manter as mesmas características do PCA, o kPCA será utilizado para implementar a função $g(.)$ do *split learning*, como será visto no Capítulo 5.

[5] A Seção 4.2 repetirá uma outra formulação *kernel* para SVMs.

Assim, está definido o método que será utilizado para implementar $g(.)$, que está no primeiro passo do *split learning*. No próximo capítulo, será descrito o algoritmo que modelará $c_{spl}(.)$: *support vector machine*. Esse tópico será descrito em um capítulo a parte por ter maior relevância para esse trabalho do que os métodos descritos nessa seção (*Riemannian manifolds* e kPCA), o que permitirá analisá-lo mais profundamente.

Foram obtidos resultados promissores com o *split learning*, utilizando-se kPCA para implementar $g(.)$ e SVM para $c_{spl}(.)$. O Capítulo 5 relata esses resultados e os experimentos realizados.

Capítulo 4

MÁQUINAS DE VETORES DE SUPORTE (SVMs)

As máquinas de vetores de suporte [37], ou *support vector machines*, são classificadores supervisionados que vêm sendo pesquisados com bastante interesse pela comunidade científica em aprendizado de máquina, por possuírem uma teoria muito bem fundamentada e um relevante histórico de sucesso em várias tarefas de classificação de dados, como classificação de textos, objetos e caracteres manuscritos. Em específico, são reconhecidamente um dos melhores tipos de algoritmos para classificação de textos [23]. Este tipo de classificador possui principalmente dois aspectos positivos importantes:

1. Robustez a erros de modelos.
2. Modularidade, ou seja, pode ser adaptado em função das características da base de dados (*kernel-based*).

Entretanto, é um tipo de classificador de treinamento custoso em termos computacionais por ser treinado através de rotinas de otimização.

Este capítulo pretende apresentar o equacionamento desses classificadores, já que esse tipo de classificador implementará o classificador supervisionado $c_{spl}(.)$ do método *split learning* (Equação 3.3). A Seção 4.1 pretende introduzir a noção de *support vector classifiers*, que pode ser vista como o caso linear de *support vector*

machines, que por sua vez serão introduzidos na Seção 4.2. Finalmente, a Seção 4.3 apresentará exemplos de métodos existentes que implementam treinamento semi-supervisionado direto em SVMs.

Além disso, vale lembrar que o Apêndice B apresenta uma SVM de treinamento semi-supervisionado direto que foi desenvolvida durante o trabalho de mestrado, denominada LUL-SVM. Por outro lado, o próximo capítulo apresentará resultados obtidos com uma SVM treinada a partir de *split learning*, que como visto no capítulo anterior, é um método de treinamento semi-supervisionado indireto.

Maiores detalhes e deduções completas abrangendo o equacionamento de SVMs podem ser encontrados em [12].

4.1 *Support vector classifiers* (SVCs)

Suponha dados de treinamento fornecidos por uma base de dados *rotulada* e *linearmente separável* $\{X, Y\} = \{x_i, y_i\}_{i=1}^{n}$, sendo x_i vetor-coluna, de forma que cada x_i esteja classificado pelo rótulo y_i. Adicionalmente, $x_i \in \Re^N$ e $y_i \in \{+1, -1\}$. Define-se então um hiperplano pela seguinte expressão [12]:

$$f(x) = x^T\beta + \beta_o = 0, \tag{4.1}$$

sendo que β é um vetor unitário, ou seja, $\|\beta\| = 1$, e β_o é uma constante. Um classificador $c(x)$ pode ser construído da seguinte forma:

$$c(x) = sign(x^T\beta + \beta_o). \tag{4.2}$$

A função $f(x)$ na Equação (4.1) fornece a distância "com sinal" entre o hiperplano ($x^T\beta + \beta_o = 0$) e x. Como o problema é linearmente separável, pode-se encontrar uma função $f(x)$ tal que, $\forall i$, $y_i f(x_i) > 0$. Assim, é possível definir um hiperplano que maximiza a "margem" entre os pontos x de classes diferentes (Figura 4.1). Esse seria um problema de otimização dado por:

$$\max_{\beta,\beta_o,||\beta||=1} C, \tag{4.3}$$

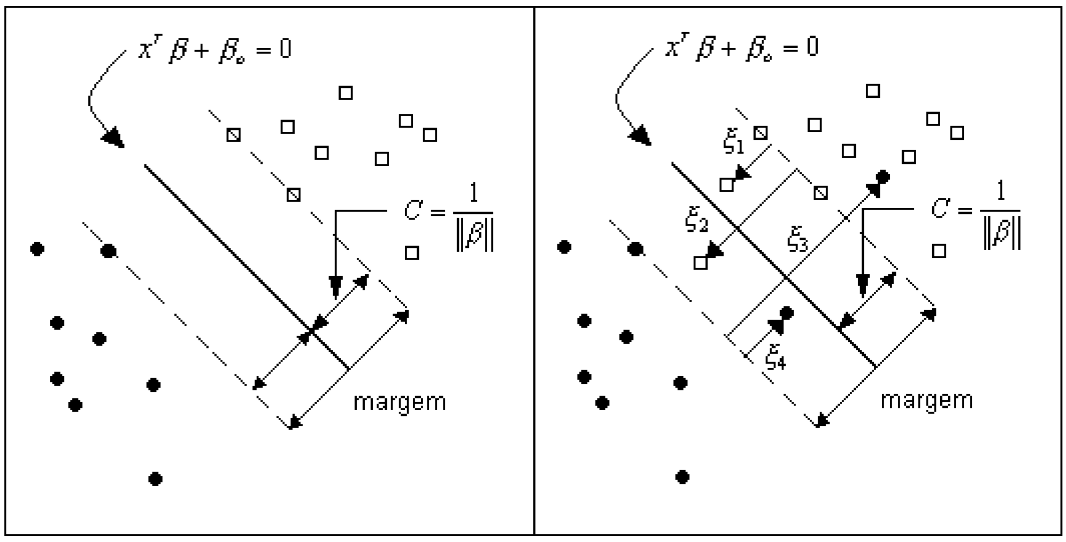

Figura 4.1. Principais parâmetros de um *support vector classifier*.

tal que,

$$y_i(x_i^T\beta + \beta_o) \geq C, \quad i = 1, \ldots, n.$$

A banda na Figura 4.1 possui C unidades de distância de cada lado do hiperplano, e $2C$ de tamanho; esse "tamanho" é chamado de *margem*.

O problema de otimização dado pela Equação (4.3) pode ser reescrito de forma mais conveniente, fazendo-se uso da informação de que $C = 1/\|\beta\|$ [12]:

$$\min_{\beta,\beta_o} \|\beta\|, \tag{4.4}$$

tal que,

$$y_i(x_i^T\beta + \beta_o) \geq 1, \quad i = 1, \ldots, n.$$

Agora, suponha que as classes são *não-linearmente separáveis*. O lado direito da Figura 4.1 ilustra esse caso. Para resolver este problema, pode-se ainda maximizar C, porém permitindo que alguns pontos estejam do "lado errado" da margem. Definem-se então algumas variáveis de "relaxação" $\xi = (\xi_1, \ldots, \xi_n)$. É possível então modificar a restrição do funcional de otimização da Equação (4.3) como segue:

$$y_i(x_i^T\beta + \beta_o) \geq C(1 - \xi_i), \ \xi_i \geq 0, \ \sum_{i=1}^{n} \xi_i \leq K, \ \forall i. \tag{4.5}$$

O valor de ξ_i é proporcional ao quanto $f(x_i)$ está do lado errado da margem. Assim, pode-se restringir a relaxação de otimização através da condição de que $\sum_{i=1}^{n} \xi_i \leq K$. Então, é possível escrever um funcional de otimização para uma SVC que consiga aprender problemas não-linearmente separáveis:

$$\min_{\beta,\beta_o,\xi_i} \|\beta\|, \tag{4.6}$$

tal que,

$$y_i(x_i^T\beta + \beta_o) \geq 1,\ \xi_i \geq 0,\ \sum \xi_i \leq K,\ \forall i.$$

Uma outra forma de se definir o funcional de uma SVC é através da minimização de uma *função perda (hinge loss)* $H(.)$ [12]:

$$\min \|\beta\|^2/2 + \Phi \sum_{i=1}^{n} H(1 - y_i(x_i^T\beta + \beta_o)), \tag{4.7}$$

sendo $H(z) = max(1 - z, 0)$ e Φ uma constante.

Os funcionais de otimização de SVCs têm em comum o fato de que dados mais próximos à margem são mais ponderados do que os mais distantes. Essa é uma propriedade característica das SVCs. Isso faz com que estes classificadores sejam sensíveis somente aos pontos próximos à margem, que efetivamente decidirão como esta será definida. Por essa razão, SVCs são ditas *robustas a erros de modelo*: como a minoria dos vetores de uma base de dados definirão a SVC (vetores próximos à margem), a presença de ruídos na base de dados tende a não influenciar relevantemente a forma final do hiperplano $f(.)$.

4.2 *Support vector machines* (SVMs)

Os SVCs encontram fronteiras lineares no espaço de variáveis original. Entretanto, pode-se construir classificadores mais flexíveis fazendo com o que o espaço de variáveis original se torne maior, através de "expansores de bases", como polinômios e *splines* [12]. Geralmente, estas transformações de base para espaços de variáveis maiores produzem melhor separação entre classes, fazendo com que classificadores lineares tenham melhor acurácia nesse novo espaço. Uma vez que as *funções de base* não-lineares $h_k(x), k = 1, \ldots, m$ são definidas, os procedimentos

de otimização da SVC não mudam; simplesmente, faz-se a otimização utilizando novos vetores de entrada $h(.) : x_i \rightarrow h(x_i) = (h_1(x_i), \ldots, h_m(x_i))$ e $h(x_i) \in \Re^m$, o que produz naturalmente um classificador não-linear $f(x) = h(x)^T\beta + \beta_o$.

Support vector machines (SVMs) são classificadores que estendem a idéia dos SVCs. As SVMs utilizam transformação de variáveis, de forma que a nova dimensão desses vetores seja bem grande, infinita em alguns casos. Entretanto, as SVMs podem se tornar *computacionalmente custosas* por lidarem com espaços muito grandes. É possível representar o funcional de otimização de uma SVC de forma que os vetores x_i da base de dados apareçam somente via *produto interno*. Este funcional pode ser escrito como um lagrangiano dual a partir do problema original (Equação 4.6) [12]:

$$\max L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}, \tag{4.8}$$

tal que,

$$0 \leq \alpha_i \leq \gamma \quad e \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Além disso, tem-se que:

$$\beta = \sum_{i=1}^{n} \alpha_i y_i x_i, \tag{4.9}$$

sendo que $\alpha_i \neq 0$ somente para x_i que seja *support vector* (vetor de suporte), dado que β é representado somente em termos desses vetores. Além disso, esses vetores "especiais" podem ser posteriormente utilizados para que β_o seja definido. Dados então β e β_o, pode-se escrever o hiperplano $f(.)$.

A partir desse ponto é que se iniciam as diferenças entre as SVCs e as SVMs. No caso do funcional dual de uma SVM, substitui-se o produto escalar $x_i^T x_{i'}$ por um produto interno no novo espaço de variáveis, como a seguir:

$$\max L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle. \tag{4.10}$$

Seguindo-se a Equação (4.9), a solução $f(x)$ pode ser escrita como:

$$f(x) = h(x)^T\beta + \beta_o = \sum_{i=1}^{n} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_o. \tag{4.11}$$

Assim, em ambas as Equações (4.10) e (4.11), o uso de $h(x)$ é necessário somente através de produtos internos. De fato, nenhum tipo de especificação da

transformação $h(x)$ é necessária; basta somente a definição de uma função *kernel* $K(.,.)$, que pode ser definida como a seguir:

$$K(x, x_i) = \langle h(x), h(x_i) \rangle. \tag{4.12}$$

Observe que quando a função *kernel* é linear ($K(x, x_i) = \langle x, x_i \rangle$), a SVM se transforma em uma SVC (o que evidencia o fato da SVC ser um caso particular de SVM).

A escolha da função *kernel*[1] influenciará a acurácia da SVM, dada uma base de dados. Dessa forma, pode-se dizer que SVMs são SVCs modulares: através da troca entre diferentes funções $K(.,.)$, têm-se vários tipos diferentes de SVMs. Um exemplo de como a escolha correta de uma certa função *kernel*, dada certa base de dados, pode influenciar na acurácia de uma futura SVM está ilustrado na Figura 4.2. Observe que no painel direito da figura os dados estão linearmente segregados em duas classes distintas, a partir da transformação de variáveis que o *kernel* oferece, diferentemente do painel esquerdo, que ilustra os dados não-linearmente separados em duas classes. A opção de reescrever os dados como mostra o painel direito tende a facilitar o aprendizado de uma SVM a partir destes dados reescritos, quando se compara com os dados escritos de acordo com o painel esquerdo da mesma figura.

Conforme discutido no Capítulo 2, atualmente não existe uma metodologia de treinamento semi-supervisionado indireto aplicado a SVMs. Entretanto, esse trabalho de mestrado contribui com o método de aprendizagem indiretamente semi-supervisionada *split learning*, que pode ser utilizado para o treinamento de *support vector machines*. O próximo capítulo apresentará resultados de experimentos realizados com SVMs treinadas a partir de *split learning*.

No caso do treinamento semi-supervisionado direto, alguns algoritmos permitem o treinamento de SVMs a partir de bases semi-rotuladas. A próxima seção apresentará alguns desses métodos.

[1] Quatro tipos mais utilizados de *kernel* estão listados na Seção 3.2.2.

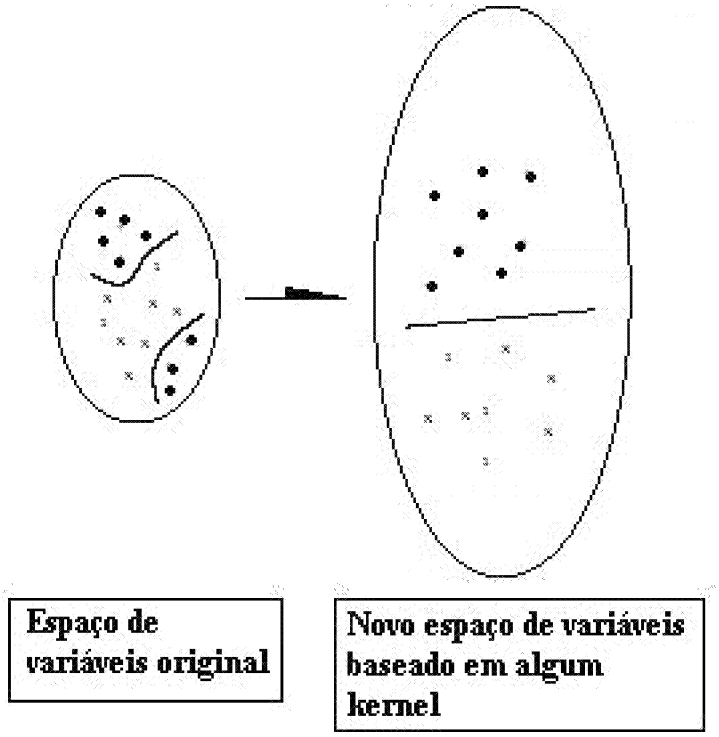

Figura 4.2. Influência da função *kernel* no aprendizado de uma SVM.

4.3 SVMs de treinamento semi-supervisionado direto

Uma solução proposta para se implementar treinamento semi-supervisionado *direto* em métodos de aprendizagem de classificadores é a partir de *transdução*. O significado deste termo foi proposto por Vapnik [4], e pode ser ilustrado pela Figura 4.3.

Num caso trivial, o classificador seria aprendido através dos dados rotulados (indução), sendo este posteriormente utilizado para rotular os dados não-rotulados (dedução.) No caso de transdução, o classificador se utiliza dos dados rotulados e não-rotulados concomitantemente: durante a execução da rotina de treinamento, os dados não-rotulados são rotulados iterativamente, de maneira que, juntamente com os dados já rotulados, minimizem o erro de classificação. Ao final do treinamento os dados não-rotulados estarão rotulados de maneira ótima (minimizam o erro de classificação), e o classificador aprendido será considerado semi-supervisionado. Ou seja, a transdução permite que os dados não-rotulados sejam rotulados ao mesmo tempo em que o classificador semi-supervisionado é treinado.

Joachims [23] implementou uma SVM diretamente semi-supervisionada a partir da idéia de transdução. Esse algoritmo recebeu o nome de *TSVM (transductive SVM)*. A TSVM tenta encontrar rótulos para os dados não-rotulados, enquanto define um hiperplano que maximiza a separação entre os dados disponíveis, rotulados ou não. Geralmente, TSVMs são eficientes em implementar treinamento semi-supervisionado direto para bases de texto.

Um outro método diretamente semi-supervisionado, ainda sob a hipótese de transdução, foi projetado por Bennett e Demiriz [22], chamado $S^3\,VM$ (*semi-supervised SVM*). A idéia de treinamento semi-supervisionado de uma S^3VM é minimizar o erro de se considerar um dado vetor não-rotulado como sendo de certa classe ($+1$ ou -1), ao mesmo tempo em que se otimiza uma SVM supervisionada utilizando os dados rotulados disponíveis e os dados não-rotulados com os rótulos estimados. Ao final do treinamento, os dados não-rotulados estarão classificados

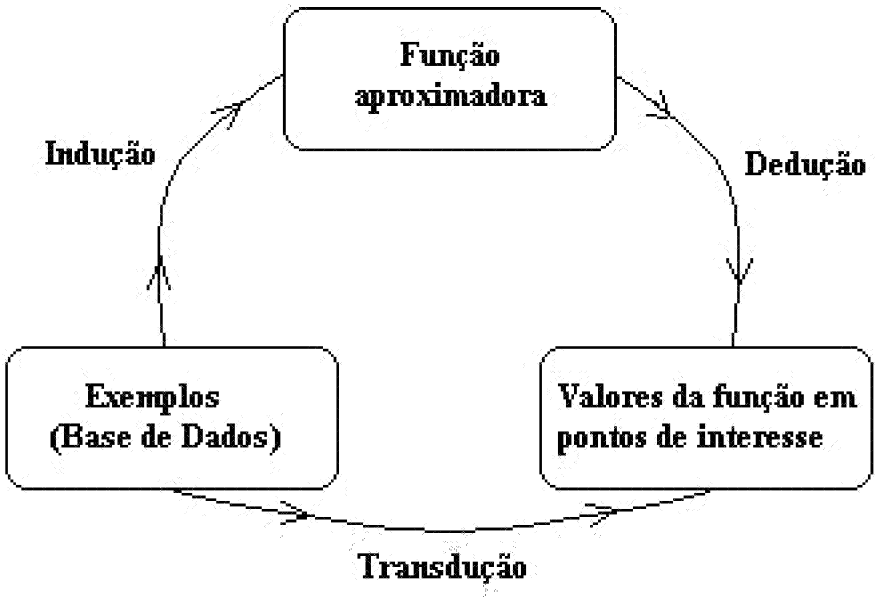

Figura 4.3. Ilustração dos significados dos termos *dedução*, *indução* e *transdução*.

de acordo com os rótulos que minimizem aquele erro.

Tanto a TSVM quanto a S^3VM exemplificam a metodologia de treinamento semi-supervisionado *direto*: os dados não-rotulados são considerados durante as rotinas de treinamento das SVMs.

Alguns métodos de treinamento semi-supervisionado direto não utilizam o conceito de transdução. O apêndice B discutirá a concepção, a experimentação e os resultados de um método de treinamento semi-supervisionado direto para SVMs, desenvolvido nos primeiros meses desse trabalho de mestrado sem partir da idéia de transdução, denominado *LUL-SVM*.

O próximo capítulo apresentará resultados obtidos com uma SVM treinada a partir de *split learning*, ou seja, a partir de um método de treinamento semi-supervisionado indireto.

Capítulo 5

SVM INDIRETAMENTE SEMI-SUPERVISIONADA A PARTIR DE *SPLIT LEARNING*

O presente capítulo relata a experimentação da principal contribuição deste trabalho, ou seja, implementação de treinamento *indiretamente* semi-supervisionado em SVMs, através da metodologia *split learning*. Os principais resultados obtidos com essa metodologia de aprendizagem também serão listados neste capítulo. Pretende-se demonstrar a validade experimental dessa metodologia que permite a utilização indireta de dados não-rotulados no treinamento supervisionado de classificadores; nesse caso, SVMs. Uma análise mais profunda dos resultados listados nesse capítulo está reservada para o Capítulo 6.

A metodologia *split learning* foi apresentada no Capítulo 3. O principal objetivo desse trabalho é investigar a validade dessa opção de treinamento semi-supervisionado em SVMs. O extrator de variáveis $g(.)$ será implementado por *kernel principal component analysis (kPCA)* (Seção 3.2.2), enquanto que o classificador $c_{spl}(.)$ será uma SVM. Como ambos os métodos são *kernel-based*, foram feitas várias combinações entre o *kernel* utilizado para se processar o kPCA e o *kernel* para se treinar a SVM.

O método kPCA foi implementado através do pacote *Statistical Pattern Recog-*

nition Toolbox for Matlab - STPRtool[1]. A SVM foi implementada através do pacote *Support Vector Machine Toolbox for Matlab* [38]. A ligação entre ambos os métodos, para formar o algoritmo de *split learning*, foi desenvolvido e escrito em *Matlab* como parte desse trabalho de mestrado.

Para se proceder com os experimentos com SVMs e *split learning*, foram utilizadas três bases de dados reais do repositório de dados UCI Database Repository [39]: *Adult* [40], *Spam* [39] e *Isolet* [41]. Além disso, foi utilizada a base de dados *20 Newsgroups*[2]. Essas bases são bem conhecidas na comunidade de aprendizado de máquina.

Foram fixadas certas quantidades de dados rotulados e não-rotulados para cada experimento, observando-se a hipótese de que o número de dados rotulados deveria ser menor do que o número de dados não-rotulados (Seção 2.1).

Para cada base, foram estudados seis casos diferentes de combinações entre SVM e método de extração de variáveis, que representarão cada curva nos gráficos apresentados adiante:

- **kPCA Linear Lab**[3]: com os dados rotulados, executava-se o kPCA de *kernel* linear e, em seguida, selecionava-se um certo número de PCs (componentes principais do kPCA), ordenadas pelas respectivas magnitudes dos auto-valores; assim, a dimensionalidade da base rotulada reescrita pelo kPCA era reduzida de acordo com o número selecionado de componentes. Com a base rotulada reescrita, aprendia-se uma SVM.

- **kPCA Linear LUL**[4]: os mesmos procedimentos realizados no caso *kPCA Linear Lab*, porém utilizando-se a base semi-rotulada para se implementar *somente* o kPCA de kernel linear. A base rotulada reescrita pelo kPCA era utilizada para se aprender uma SVM.

- **kPCA RBF Lab**[5]: com os dados rotulados, executava-se o kPCA de *kernel* RBF e, em seguida, selecionava-se um certo número de PCs, ordenadas pelas

[1] O pacote STPRtool pode ser encontrado em *http://cmp.felk.cvut.cz*.
[2] http://people.csail.mit.edu/jrennie/20newsgroups.
[3] *Lab* é abreviatura para *labeled*.
[4] *LUL* é abreviatura para *labeled + unlabeled*.
[5] A sigla, em inglês, *RBF* vem de *radial basis function*, ou função de base radial.

respectivas magnitudes dos auto-valores. Com a base rotulada reescrita, aprendia-se uma SVM.

- **kPCA RBF LUL**: os mesmos procedimentos realizados no caso *kPCA RBF Lab*, porém utilizando-se a base semi-rotulada para implementar *somente* o kPCA de *kernel* RBF. A base rotulada reescrita pelo kPCA era utilizada para se aprender uma SVM.

- **kPCA Poly(2) Lab**[6]: com os dados rotulados, executava-se o kPCA de *kernel* polinomial de grau 2 e, em seguida, selecionava-se um certo número de PCs, ordenadas pelas respectivas magnitudes dos auto-valores. Com a base rotulada reescrita, aprendia-se uma SVM.

- **kPCA Poly(2) LUL**: os mesmos procedimentos realizados no caso *kPCA Poly(2) Lab*, porém utilizando-se a base semi-rotulada para implementar *somente* o kPCA de *kernel* polinomial de grau 2. A base rotulada reescrita pelo kPCA era utilizada para se aprender uma SVM.

Cada ponto do gráfico, que representa a acurácia da SVM treinada com base rotulada reescrita a partir de um dos seis casos, é a média de 10 testes diferentes, realizados a partir de *10-fold cross validation* (ou validação cruzada dividindo-se a base de dados em 10 setores). Foram utilizados 20 dados rotulados e 100 dados não-rotulados para os testes e para cada setor do *cross validation*, a menos da base *Spam*, da qual foram extraídos 22 dados rotulados e 100 não-rotulados para a execução dos testes e para cada setor do *cross validation*.

Nos gráficos que serão apresentados, o eixo das ordenadas representa o *aumento* médio de acertos, em porcentagem (%), *sobre* os valores médios de acerto de uma SVM supervisionada "trivial", ou seja, *fazendo-se uso somente dos dados rotulados sem nenhum processamento prévio de extração de variáveis e/ou redução de dimensionalidade*; este caso será referido como *"SVM trivial"*. O valor médio de acertos (em porcentagem %, acompanhado de seu respectivo desvio-padrão logo após o ponto-e-vírgula) para o caso trivial de utilização da base de

[6] *Poly(2)* vem de *polynomial(2)*, ou *kernel* polinomial de grau 2.

dados rotulada encontra-se entre parênteses no título de cada gráfico. Assim, 5% no eixo das ordenadas representa o valor da "SVM trivial" *mais* 5%; i. e., *aumento médio de acurácia de 5% em relação à "SVM trivial".*

O eixo das abscissas fornece o número de PCs selecionados para cada curva do gráfico. Entre parênteses, encontra-se o número de componentes original da base, isto é, sem nenhum tipo de processamento.

Os resultados são apresentados nas seções seguintes. Alguns gráficos não contemplam todos os seis casos de teste; quando isso ocorre, o treinamento da SVM para os casos faltantes se apresentou demasiadamente longo, ou seja, a convergência do algoritmo de aprendizagem da SVM acontecia a taxas muito pequenas, o que inviabilizava o experimento. Um detalhamento maior sobre estes resultados (incluindo respectivos desvios-padrões) pode ser encontrado no Apêndice A.

Alguns comentários serão feitos sobre os resultados; entretanto, o próximo capítulo apresenta uma discussão mais prolongada e generalizada sobre os resultados obtidos.

5.1 Base de dados *20 Newsgroups*

A base de dados *20 Newsgroups* é formada por vários diretórios, sendo que cada um desses contém mensagens compiladas de um certo fórum de um dado tópico de discussão. Para esse trabalho de mestrado, foram utilizados os diretórios *rec.autos* e *rec.motorcycles*, que contêm mensagens sobre automóveis e motocicletas, respectivamente. A idéia é classificar automaticamente uma dada mensagem como sendo referente ao fórum de automóveis ou motocicletas.

Fazendo-se uso do pacote *Rainbow* [42], cada componente da base *20 Newsgroups* representava o número de vezes que uma certa palavra aparecia (*tokenization*) em uma dada mensagem ou amostra. O número de componentes (palavras) da base foi limitado a 1250, a partir do *Rainbow.*

Os resultados estão apresentados nas Figuras 5.1, 5.2 e 5.3.

Observe que pela Figura 5.1, utilizar *kernel* polinomial de grau 2 no kPCA (combinado ao *kernel* RBF da SVM) somente com os dados rotulados e considerar

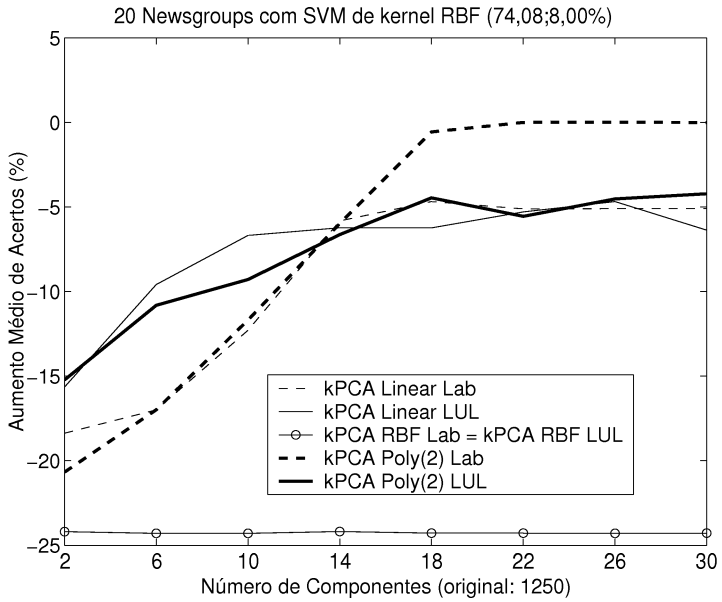

Figura 5.1. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* RBF.

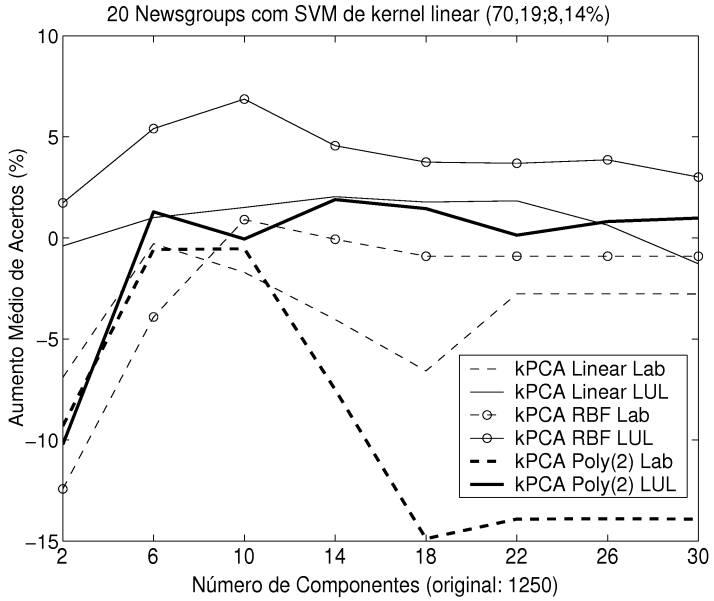

Figura 5.2. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* linear.

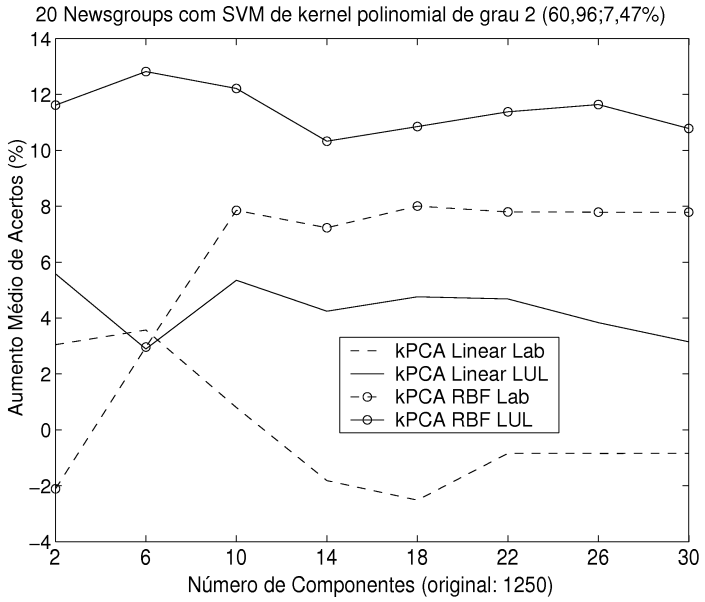

Figura 5.3. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* polinomial 2.

a partir de 18 PCs (1,5% do número original de atributos) garante pelo menos a manutenção da acurácia da "SVM trivial" (em média, 74,08% de acertos). No caso desta figura, a inclusão de dados não-rotulados traz prejuízos ao classificador porque 20 dados rotulados da base de dados *20 Newsgroups* já é suficiente para o treinamento de um bom classificador, o que demonstra empiricamente a conclusão da Seção 2.1 de que a inclusão de dados não-rotulados pode prejudicar o classificador quando os dados rotulados já são suficientes.

Já pelas Figuras 5.2 e 5.3, a inclusão de dados não-rotulados foi benéfica para o classificador porque 20 dados rotulados não eram suficientes para produzir "SVMs triviais" de boa acurácia (média de acertos de 70,19% para SVM de *kernel* linear e 60,96% para SVM de *kernel* polinomial de grau 2) quando comparadas ao caso de SVM de *kernel* RBF. Note que o *split learning* mais ajuda conforme pior é a acurácia da "SVM trivial" (caso da Figura 5.3).

5.2 Base de dados *Spam*

A base de dados *Spam* consiste de uma coleção de mensagens eletrônicas (*e-mails*) cujos conteúdos foram classificados como 'spam' (mensagem não-solicitada) ou 'não-spam'. As 4601 mensagens (amostras) são descritas a partir de 56 componentes, sendo que a grande maioria desses atributos indicam a freqüência com que certa palavra ocorre em uma dada amostra (*tokenization*). Dada uma mensagem, a tarefa é classificá-la como 'spam' ou 'não-spam'. Os resultados obtidos com esta base de dados estão compilados nas Figuras 5.4, 5.5 e 5.6.

Observando as figuras, nota-se o mesmo fênomeno ocorrido para base *20 Newsgroups*: a inclusão de dados não-rotulados aumenta mais a acurácia do classificador conforme menor é a acurácia da "SVM trivial" (por exemplo, Figura 5.4 contra Figura 5.6).

As Figuras 5.5 e 5.6 demonstram a eficiência do método *split learning*: a inclusão de 100 dados não-rotulados garante o treino de SVMs com apenas 20 dados rotulados de 2 variáveis, cuja acurácia atinge médias de 82% de acertos. De acordo com os 3 gráficos apresentados, médias de 82% de acertos não seriam

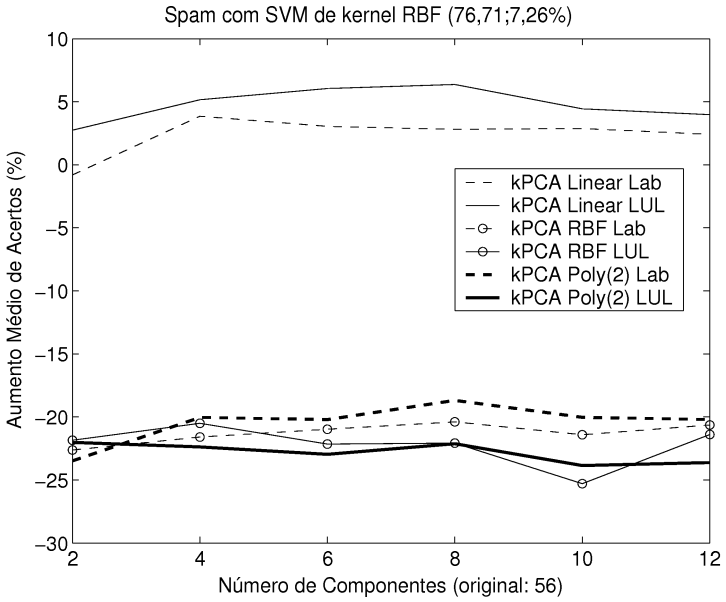

Figura 5.4. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Spam* e SVM de *kernel* RBF.

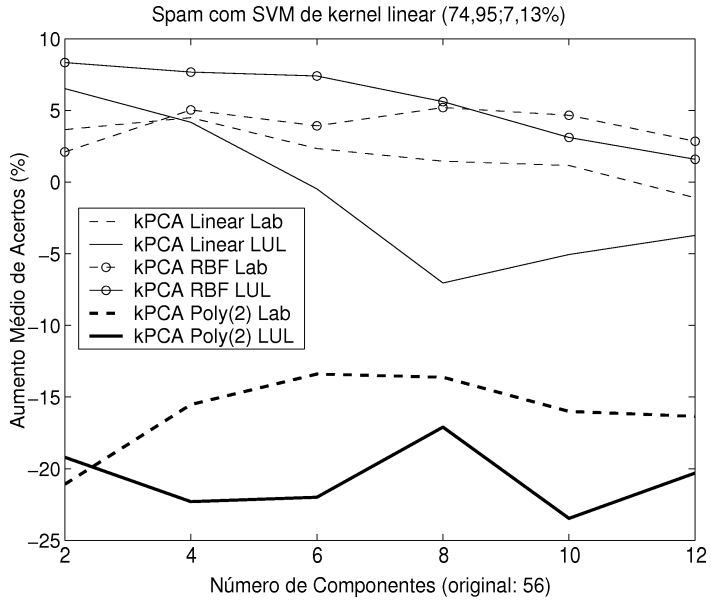

Figura 5.5. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Spam* e SVM de *kernel* linear.

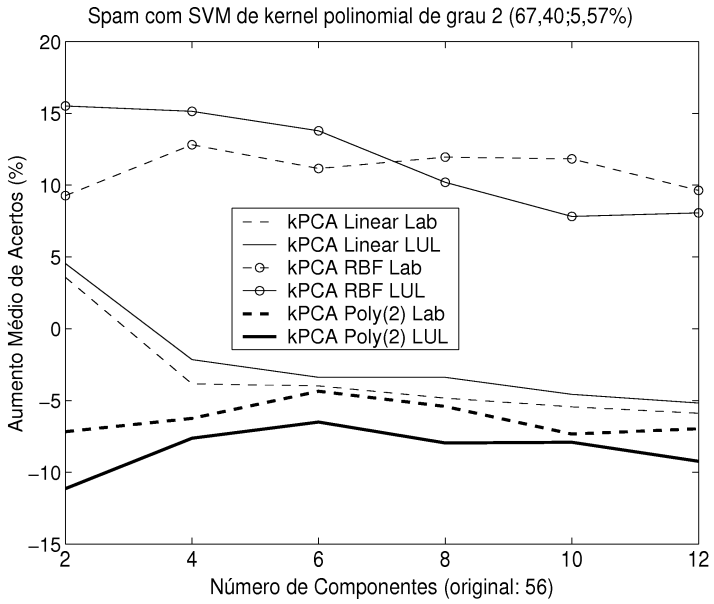

Figura 5.6. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Spam* e SVM de *kernel* polinomial 2.

atingidas somente com os dados rotulados disponíveis.

5.3 Base de dados *Adult*

Essa base de dados é uma compilação de quase 50000 amostras do censo norte-americano de 1994. Contém 14 componentes, sendo que algumas delas correspondem à idade do entrevistado, escolaridade, estado civil, profissão e sexo. A renda de cada indivíduo é classificada como sendo superior a US$ 50000 ou inferior ou igual a US$ 50000 anuais; dessa forma, a tarefa é treinar uma SVM que classifique automaticamente a renda de um indivíduo dadas algumas de suas características (atributos).

Alguns atributos dessa base de dados que eram categóricos (exemplo do atributo profissão) foram transformados para numéricos (a cada profissão era atribuído um número inteiro) a fim de permitir sua utilização nas rotinas de Matlab que implementam o *split learning*, que não permitem o uso de variáveis não-numéricas.

Os resultados obtidos com a base de dados *Adult* estão agrupados pelas Figuras 5.7, 5.8 e 5.9.

Note que, como nos casos anteriores, a inclusão de dados não-rotulados mais beneficia um classificador quanto pior é a acurácia de sua utilização "trivial". Entretanto, mesmo nos piores casos de "SVM trivial", a inclusão de dados não-rotulados beneficiou menos o classificador do que nos casos das bases *20 Newsgroups* (Seção 5.1) e *Spam* (Seção 5.2).

De qualquer forma, o *split learning* permitiu, através da adição de 100 dados não-rotulados, treinar SVMs de melhores médias de acertos com apenas 20 dados rotulados de 1 atributo (7% do número original de componentes).

5.4 Base de dados *Isolet*

As 7700 amostras da base de dados *Isolet* representam letras do alfabeto (inclusive K, W e Y) pronunciadas por indivíduos, posteriormente gravadas e decompostas

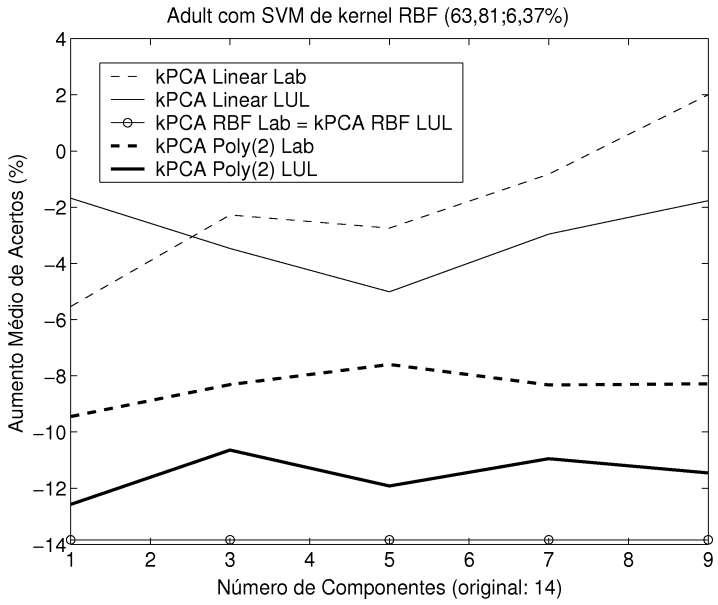

Figura 5.7. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Adult* e SVM de *kernel* RBF.

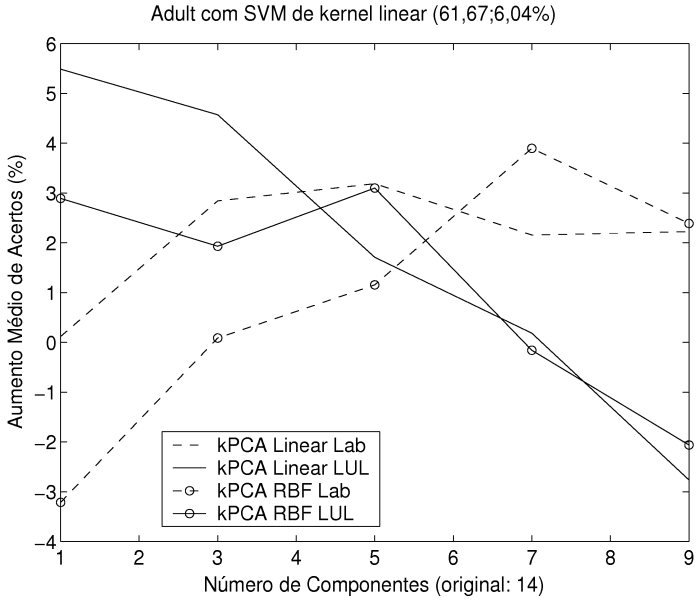

Figura 5.8. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Adult* e SVM de *kernel* linear.

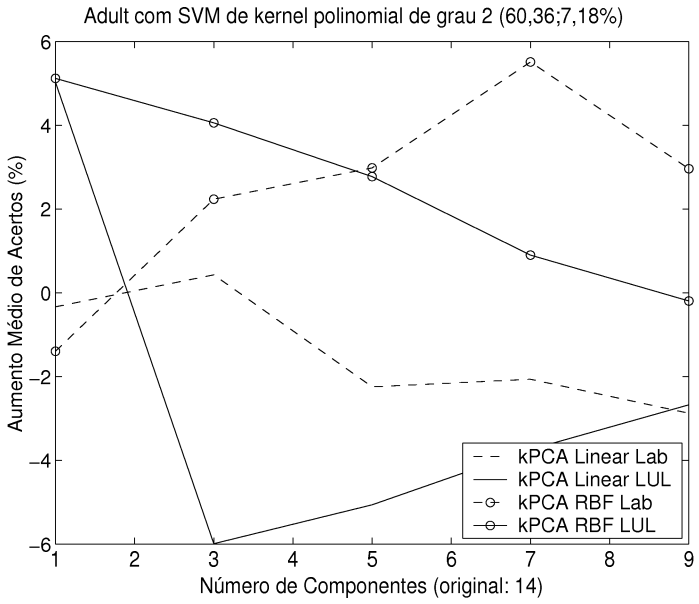

Figura 5.9. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Adult* e SVM de *kernel* polinomial 2.

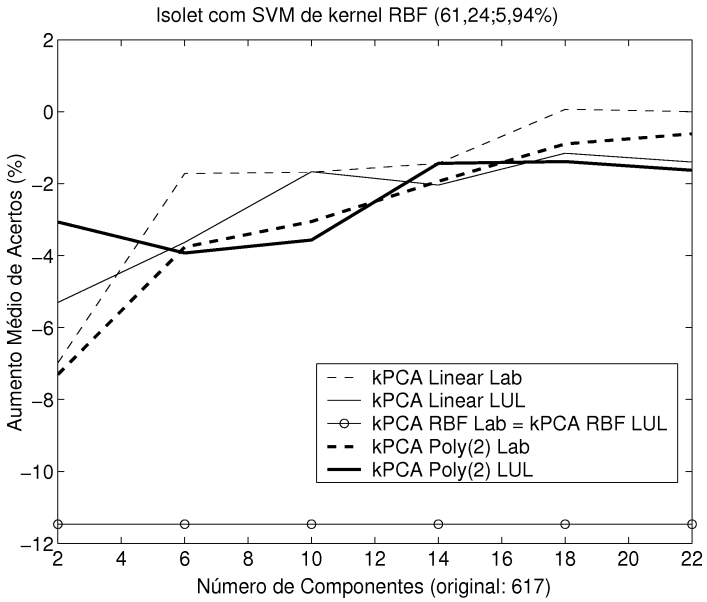

Figura 5.10. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Isolet* e SVM de *kernel* RBF.

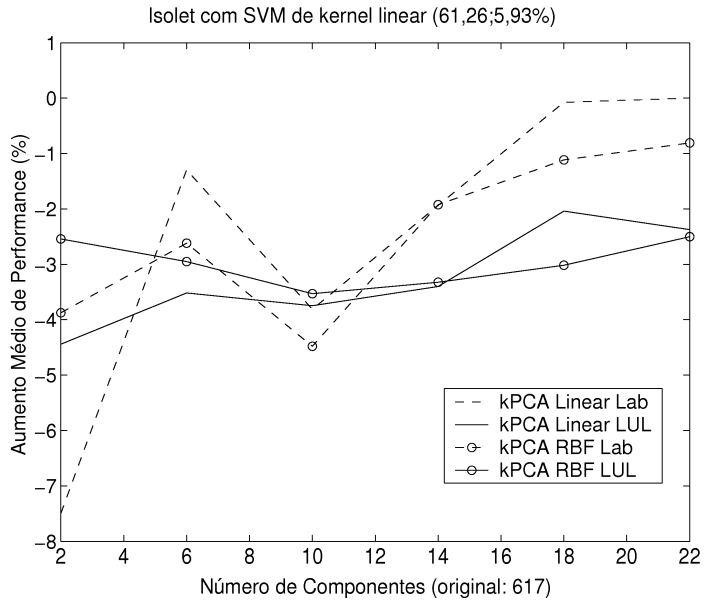

Figura 5.11. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Isolet* e SVM de *kernel* linear.

(descritas) em 617 componentes (variáveis). A tarefa era classificar uma certa amostra como sendo uma letra entre A e M (classe 1), ou entre N e Z (classe 2).

Os resultados obtidos com esta base de dados estão compilados nas Figuras 5.10, 5.11 e 5.12.

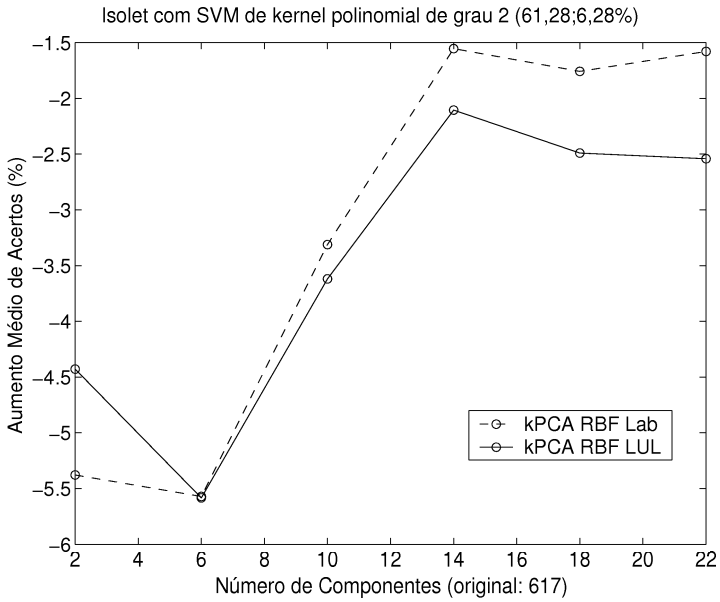

Figura 5.12. Resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Isolet* e SVM de *kernel* polinomial 2.

Note que para essa base de dados, o método *split learning* não foi eficiente em incluir dados não-rotulados para o treinamento da SVM, sendo que a presença dos dados não-rotulados prejudicou a acurácia da SVM frente ao caso "trivial".

O próximo capítulo apresentará uma discussão mais detalhada sobre os resultados listados nesse capítulo.

Capítulo 6

DISCUSSÃO DOS RESULTADOS OBTIDOS A PARTIR DE *SPLIT LEARNING* E SVM

Os resultados obtidos através dos experimentos feitos a partir de SVMs treinadas com *split learning*, mostram que o método é válido para a inclusão de dados não-rotulados em classificadores supervisionados; neste caso, SVMs.

O método de extração de variáveis kPCA é eficiente em reescrever melhores versões das bases de dados em novos espaços de variáveis, possibilitando, através dos dados rotulados melhor reescritos, o aprendizado de SVMs supervisionadas de melhor acurácia. Em casos em que isso não ocorre, o kPCA no mínimo permite redução da dimensionalidade das bases de dados. Por exemplo, no caso da base de dados *20 Newsgroups*, cujos dados possuem 1250 dimensões (variáveis), o kPCA foi capaz de reduzir a dimensionalidade dessa base para valores próximos a apenas 0,5% do número de variáveis original, ao mesmo tempo que possibilitava o aprendizado de SVM de acurácia de até, em média, 13% melhor quando comparada à "SVM trivial".

Além disso, a partir dos resultados obtidos, quanto mais *esparsa* é uma base de dados, mais o kPCA permite, através da inclusão de dados não-rotulados, melhores versões dos dados em novos espaços de variáveis, permitindo SVMs de maior acurácia. A seguir, as bases são listadas em ordem decrescente de

esparsidade:

1. *20 Newsgroups:* 98,7% dos elementos da base de dados[1] são nulos.
2. *Spam:* 76% dos elementos são nulos.
3. *Adult:* 13% dos elementos são nulos.
4. *Isolet:* apenas 0,4% dos elementos são nulos.

A base *20 Newsgroups* é a mais esparsa das quatro, e a que possui o maior número de variáveis. É exatamente esse tipo de base de dados que mais pode se beneficiar da utilização do *split learning*. Isso ocorre porque em uma base esparsa (se possível com um grande número de componentes), a utilização de poucos dados rotulados permite ao classificador aprender muito pouco sobre o problema, já que a maioria das componentes são nulas. Assim, a inclusão de dados não-rotulados para se processar *split learning* faz com que os poucos dados rotulados sejam reescritos em função dos muitos dados não-rotulados, permitindo que a SVM aprenda, *indiretamente*, que as componentes nulas nos dados rotulados nem sempre serão nulas quando se aumenta o tamanho da amostra do problema a ser aprendido. Permite-se então que o classificador, utilizando a mesma quantidade de dados rotulados, tenha mais informação sobre o problema do que disponibilizam esses mesmos dados rotulados.

Como exemplo, tem-se o caso dos experimentos com a base *20 Newsgroups* da Seção 5.1, em que a "SVM trivial" é aprendida com apenas 20 dados rotulados. Em 20 dados, tem-se 25000 componentes, o que totaliza 1,3% de 25000, ou seja, apenas 325 componentes não-nulas. Dentre esses 20 dados, pode ser que todos possuam a última componente nula. Dessa forma, qualquer dado de teste posterior que possua a última componente não-nula trará um estado espúrio a SVM, que aprendeu que a última componente da base era sempre nula. Ao se incluir dados não-rotulados para se processar o *split learning*, os mesmos 20 dados rotulados foram agora condicionados ao fato de que nem sempre a última componente será nula; por conseguinte, essa informação é levada à SVM no decorrer

[1] Considerando a base de dados como uma matriz, na qual cada coluna é uma amostra e cada linha uma componente.

da cadeia do *split learning*. Não obstante, quando se aumenta de 100 para 1000 dados não-rotulados e se processa o *split learning* pode-se obter acurácias médias de até 82% de acertos com apenas 20 dados rotulados e 1% da quantidade de componentes originais[2].

Por outro lado, a base *Isolet* não se beneficia da execução de *split learning* para acarretar melhor acurácia ao classificador, pois não é uma base de dados esparsa. De qualquer forma, há o benefício da redução de dimensionalidade. As bases de dados *Spam* e *Adult* se situam como casos intermediários entre o caso da base *20 Newsgroups* e *Isolet*.

No geral, o uso do *split learning* sempre permite uma redução eficiente de dimensionalidade da base de dados, ou seja, ocorre redução de dimensionalidade sem prejuízo à acurácia do classificador[3], ou de outra forma, não ocorre perda de informação. Isso implica em vetores menores e, dado que o método foi eficiente em reescrever a base de dados (caso de bases mais esparsas), melhor escritos em seu novo espaço de variáveis. Assim, como a SVM é treinada de forma supervisionada com dados menores e melhor escritos, isso gera um treinamento mais ágil da SVM (por parte da reduzida quantidade de dados rotulados e de baixa dimensionalidade) e uma melhor média de acertos final (por parte dos dados rotulados reestruturados). Em comparação, algoritmos de treinamento semi-supervisionado direto requerem a utilização de dados não-rotulados adicionais (e concomitantes) aos dados rotulados durante a aprendizagem do classificador. Dessa forma, o treinamento semi-supervisionado direto tende a ser mais lento (maior quantidade de dados). Obviamente que no caso do *split learning* (caso particular de algoritmo de treinamento semi-supervisionado indireto), previamente à SVM existe a execução do kPCA; no entanto, o tempo conjunto da execução e treinamento supervisionado do kPCA e SVM, respectivamente, geralmente é menor quando comparado ao tempo de treinamento semi-supervisionado direto.

Do parágrafo anterior e dos resultados com a base *20 Newsgroups* (Seção

[2]O experimento foi realizado separadamente dos testes listados no Capítulo 5. Por isso não se encontra registrado nos gráficos daquele capítulo.

[3]Observe que no caso dos experimentos com a base de dados *Isolet*, a execução do *split learning* somente com os dados rotulados permitiu redução de dimensionalidade sem acarretar prejuízos à acurácia da SVM.

5.1), verifica-se que o *split learning* é uma ferramenta importante para lidar com situações em que a base de dados seja de grande porte. No caso da base *20 Newsgroups*, e considerando algum algoritmo de treinamento semi-supervisionado direto, seria computacionalmente custoso executar o treinamento de um classificador semi-supervisionado com 20 dados rotulados e mais 1000 dados não-rotulados, considerando que essa base de dados possui 1250 atributos. O *split learning*, além de utilizar os dados não-rotulados para melhorar o espaço de variáveis da base de dados e "livrar" o classificador de processar também os dados não-rotulados, ainda permitiu uma redução dimensional drástica no tamanho da base, fazendo com que uma base de grande porte fosse transformada em uma base de pequeno porte e mitigando a carga computacional que o classificador enfrentaria se tivesse de ser treinado a partir da base *20 Newsgroups* sem nenhum tipo de pré-processamento.

Outra importante discussão é a comparação dos resultados obtidos a partir do *split learning* com outros métodos de aprendizado semi-supervisionado indireto. Tendo como ponto de comparação a base de dados *20 Newsgroups*, o *split learning* apresentou resultados ligeiramente superiores aos obtidos pelos métodos de treinamento semi-supervisionado indireto apresentados nos textos [32, 33]. Entretanto, a partir da constatação de que o *split learning* é menos complexo do que os métodos em comparação, evidencia-se que as características deste algoritmo fazem dele uma metodologia simples, porém eficiente em incorporar dados não-rotulados a classificadores supervisionados.

Um fato que favorece o *split learning*, em específico quando se utiliza uma SVM como classificador e bases de texto para o treinamento (como as bases de dados *20 Newsgroups* e *Spam*), é que as SVMs são reconhecidas pela comunidade de aprendizado de máquina como os melhores classificadores quando se deseja aprender bases de texto [23]. Portanto, quando se consegue um incremento de acurácia sobre a SVM por meio do *split learning* e da adição de dados não-rotulados, dificilmente este incremento seria obtido através somente dos dados rotulados e algum método supervisionado.

Capítulo 7

CONCLUSÕES

Um método relativamente simples de treinamento semi-supervisionado indireto, *split learning* (Capítulo 3), é eficiente e ágil para utilizar dados não-rotulados e treinar classificadores supervisionados de acurácias superiores. Esse método pode utilizar os dados não-rotulados para reestruturar a base de dados em um espaço de variáveis mais conveniente e reduzir sua dimensionalidade. Assim, criam-se condições para tempos reduzidos de treinamento supervisionado, conforme discutido no Capítulo 6, enquanto permite o aprendizado de classificadores supervisionados de melhor acurácia. Esse método se mostrou mais eficiente quanto mais esparsa é uma dada base.

Atualmente há pouca literatura e um pequeno número de algoritmos que contemplam a idéia de treinamento semi-supervisionado indireto, como por exemplo [32, 33] e o próprio *split learning*. Maiores esforços deveriam ser direcionados para se buscar novos modelos de treinamento semi-supervisionado indireto, já que este tipo de treinamento semi-supervisionado apresenta algumas vantagens sobre o treinamento semi-supervisionado direto, conforme discutido na Seção 2.4.

Uma das vantagens mais evidentes da modalidade indireta é a existência de um passo de pré-processamento dos dados. Existe então a possibilidade de detecção de ruídos e erros na seleção de variáveis, e definição de espaços de variáveis melhores e reduzidos. Todas essas características fazem com que o treinamento semi-supervisionado indireto seja uma maneira mais organizada de se fazer aprendizagem, pois previamente à aprendizagem do classificador resolvem-se problemas

e utilizam-se informações fornecidas por toda a massa de dados (rotulados ou não) em prol de uma tarefa de aprendizagem mais eficiente e ágil. Além disso, essa fase preliminar é intercambiável, pois as análises e melhorias feitas sobre a base de dados podem ser aproveitadas por qualquer classificador em um segundo passo.

Na modalidade direta de treinamento semi-supervisionado, toda a massa de dados é utilizada concomitantemente durante o treinamento do classificador. Assim, perde-se a oportunidade de análise da base de dados. Isso faz com que muitas vezes algoritmos de treinamento semi-supervisionado direto sejam custosos do ponto de vista computacional, pois processam muitos dados ao mesmo tempo sem a opção de redução do tamanho dos dados, o que dificulta a utilização de bases de dados de grande porte.

Conforme o Capítulo 6, a base de dados *20 Newsgroups* pode ser considerada uma base de dados de grande porte. A partir do passo preliminar do algoritmo *split learning*, a base *20 Newsgroups* pré-processada teve sua dimensionalidade reduzida drasticamente, e seu problema estrutural de alta esparsidade foi mitigado redefinindo-se os dados em um espaço de variáveis mais conveniente, que evidenciava ao classificador o fato de que nem sempre os atributos nulos nos dados rotulados serão também nulos na base de dados completa. Isso permitiu que uma SVM supervisionada fosse aprendida de maneira rápida e eficiente a partir de poucos dados rotulados de poucas variáveis. No caso de se utilizar a base *20 Newsgroups* na aprendizagem de um classificador diretamente semi-supervisionado, provavelmente esse seria demasiadamente lento, devido ao tamanho da base de dados. Em contrapartida, um classificador supervisionado apresentaria baixa acurácia devido à esparsidade dessa base.

O *split learning*, para executar treinamento semi-supervisionado indireto, utiliza uma SVM supervisionada, sem requerer nenhum tipo de adaptação nesse classificador. Isso evidencia a vantagem de que algoritmos de treinamento semi-supervisionado indireto podem treinar classificadores que não necessariamente sejam semi-supervisionados, diferentemente do caso direto. Visto que modelar classificadores semi-supervisionados nem sempre é tarefa fácil [34], métodos de aprendizagem indiretamente semi-supervisionada podem facilitar a utilização de

bases semi-rotuladas.

Em específico sobre esse trabalho, seria proveitosa uma análise futura sobre como a combinação *kernel* do kPCA e *kernel* da SVM influencia a acurácia do método *split learning* como um todo. Além disso, poderiam ser testados novos algoritmos para se implementar as funções de extração de variáveis ($g(.)$) e de classificador ($c_{spl}(.)$). Uma forma talvez promissora de se implementar $g(.)$ seria a partir de funções *kernel* extraídas em função de modelos probabilísticos da base de dados, como por exemplo a função *Fisher kernel* [43].

Esse trabalho de mestrado oferece como principal contribuição um novo algoritmo, denominado *split learning*, que se situa dentro da metodologia de treinamento semi-supervisionado indireto. Esse algoritmo, de fácil uso, permite incorporar indiretamente dados não-rotulados a classificadores supervisionados, fazendo com que esses classificadores se tornem mais eficientes e ágeis. Em alguns casos, o *split learning* é superior a outros métodos mais complexos de treinamento semi-supervisionado indireto. Além disso, a contribuição secundária desse trabalho é a constatação de que os métodos de treinamento semi-supervisionado podem ser divididos em *diretamente* e *indiretamente* semi-supervisionados.

Referências Bibliográficas

[1] DURKIN, J., **Expert Systems: Design and Development**. New York: Macmillan, 1994.

[2] RUSSELL, S. J.; NORVIG, P., **Artificial Intelligence. A Modern Approach**. Englewood Cliffs: Prentice-Hall, 1995.

[3] IDE, J. S., **Geração de Redes Bayesianas Uniformemente Distribuídas**. Dissertação de mestrado, São Paulo, 2002, Escola Politécnica da Universidade de São Paulo - Departamente de Engenharia Mecatrônica.

[4] VAPNIK, V., **Statistical Learning Theory**. Wiley, 1998.

[5] FAHLE, M.; EDELMAN, S.; POGGIO, T., *Fast Perceptual Learning in Visual Hyperacuity*, **Vision Research**, n. 35, p. 3003–3013, 1995.

[6] SZUMMER, M. O., **Learning from Partially Labeled Data**. Tese de doutorado, setembro 2002, Massachusetts Institute of Technology (MIT) - Department of Electrical Engineering and Computer Science.

[7] HAYKIN, S., **Neural networks: a comprehensive foundation**. Upper Saddle River, NJ, EUA: Prentice Hall, 2a. ed., 1999.

[8] OLIVEIRA, C. S.; HERNANDEZ, E. D. M., *MLP+H: a Hybrid Neural Architecture Formed by the Interaction of Hopfield and Multi-Layer Perceptron Neural Networks*, em **Proceedings of the 7th International Work Conference on Artificial and Natural Neural Networks (IWANN)**, (Menorca, Spain), Computational Methods in Neural Modeling – Lecture

Notes in Computer Science (vols. 2686 and 2687) – Springer-Verlag, junho 2003.

[9] OLIVEIRA, C. S.; HERNANDEZ, E. D. M., *As Redes Neurais de Hopfield e Multi-Layer Perceptrons Formando uma Arquitetura Neural Híbrida (MLP+H) com Características Próprias*, em **Proceedings do** 6^o **Congresso Brasileiro de Redes Neurais (CBRN)**, (Centro Universitário da FEI, São Paulo, Brasil), junho 2003.

[10] OLIVEIRA, C. S.; HERNANDEZ, E. D. M., *Forms of Adapting Patterns to Hopfield Neural Networks with Larger Number of Nodes and Higher Storage Capacity*, em **Proceedings of the International Joint Conference on Neural Networks (IJCNN)**, (Budapeste, Hungria), julho 2004.

[11] BILMES, J. A., *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, rel. tec., International Computer Science Institute - U. C. Berkeley, abril 1998. TR-97-021.

[12] HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J., **The Elements of Statistical Learning**. Springer, 2003.

[13] CASTELLI, V.; COVER, T., *On the Exponential Value of Labeled Samples*, **Pattern Recognition Letters**, v. 16, p. 105–111, 1995.

[14] CASTELLI, V.; COVER, T., *The Relative Value of Labeled and Unlabeled Samples in Pattern Recognition with an Unknown Mixing Parameter*, **IEEE Transactions on Information Theory**, v. 42, p. 2102–2117, 1996.

[15] CASTELLI, V., **The Relative Value of Labeled and Unlabeled Samples in Pattern Recognition**. PhD thesis, 1994, Stanford University.

[16] RATSABY, J.; VENKATESH, S., *Learning from a Mixture of Labeled and Unlabeled Examples with Parametric S Information*, em **Proceedings of the COLT 1995**, p. 412–417, 1995.

[17] COOPER, D.; FREEMAN, J., *On the Asymptotic Improvement in the Outcome of Supervised Learning Provided by Additional Nonsupervised Learning*, **IEEE Transactions on Computers**, v. C, p. 1055–1063, novembro 1970.

[18] O'NEILL, T., *Normal Discrimination with Unclassified Observations*, **Journal of the American Statistical Association**, v. 73, n. 364, p. 821–826, 1978.

[19] COZMAN, F. G.; COHEN, I., *Unlabeled Data Can Degrade Classification Performance of Generative Classifiers*, **Conference of the American Association for Artificial Intelligence**, 2002.

[20] COZMAN, F. G.; COHEN, I.; CIRELO, M. C., *Semi-Supervised Learning of Mixture Models*, em **Proceedings of the 20th International Conference on Machine Learning (ICML)**, (Washington DC, EUA), 2003.

[21] COHEN, I., **Semi-Supervised Learning of Classifiers with Application to Human Computer Interaction**. Tese de doutorado, 2003, University of Illinois at Urbana-Champaign.

[22] BENNETT, K. P.; DEMIRIZ, A., *Semi-Supervised Support Vector Machines*, em **Proceedings of the 1998 Neural Information Processing Systems**, (Denver, EUA), 1998.

[23] JOACHIMS, T., *Transductive Inference for Text Classification using Support Vector Machines*, em **Proceedings of the 16th International Conference on Machine Learning (ICML)**, 1999.

[24] ZHANG, T.; OLES, F., *A Probability Analysis on the Value of Unlabeled Data for Classification Problems*, em **Proceedings of the 17th International Conference on Machine Learning (ICML)**, 2000.

[25] BLUM, A.; MITCHELL, T., *Combining Labeled and Unlabeled Data with Co-Training From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering*, em **Proceedings**

of the 1998 Conference on Computational Learning Theory, julho 1998.

[26] NIGAM, K.; MCCALLUM, A.; THRUN, S.; MITCHELL, T., *Text Classification from Labeled and Unlabeled Documentes Using EM*, **Machine Learning**, v. 39, p. 103–134, 2000.

[27] SHAHSHAHANI, B.; LANDGREBE, D., *Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon*, **IEEE Transactions on Geoscience and Remote Sensing**, v. 32, n. 5, p. 1087–1095, 1994.

[28] BALUJA, S., *Probabilistic Modelling for Face Orientation Discrimination: Learning from Labeled and Unlabeled Data*, **Neural Information and Processing Systems**, 1998.

[29] BRUCE, R., *Semi-Supervised Learning Using Prior Probabilities and EM*, em **IJCAI-01 Workshop on Text Learning: Beyond Supervision**, agosto 2001.

[30] GHANI, R., *Combining Labeled and Unlabeled Data for Multiclass Text Categorization*, em **Proceedings of the 2002 International Conference on Machine Learning (ICML)**, 2002.

[31] SEEGER, M., *Learning with Labeled and Unlabeled Data*, rel. tec., Edinburgh University, 2001.

[32] BELKIN, M.; NIYOGI, P., *Semi-Supervised Learning on Riemannian Manifolds*, **Machine Learning**, n. 56, p. 209–239, 2004.

[33] ANDO, R. K.; ZHANG, T., *A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data*. IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA, 2005.

[34] OLIVEIRA, C. S.; COZMAN, F. G.; COHEN, I., *Splitting the Unsupervised and Supervised Components of Semi-Supervised Learning*, em **Proceedings**

of the 22nd International Conference on Machine Learning (ICML-2005) – Learning with Partially Classified Training Data Workshop, (Bonn, Germany), agosto 2005.

[35] SHLENS, J., *A Tutorial On Principal Component Analysis - Derivation, Discussion and Singular Value Decomposition*, rel. tec., UCSD, março 2003.

[36] SCHÖLKOPF, B.; SMOLA, A.; MÜLLER, K.-R., *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, **Neural Computation**, n. 10, p. 1299–1319, 1998.

[37] VAPNIK, V., **Estimation of Dependences Based on Empirical Data**. Springer-Verlag, 1982.

[38] CAWLEY, G. C., *MATLAB Support Vector Machine Toolbox (v0.50β)* [*http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox*]. University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000.

[39] BLAKE, C.; MERZ, C., *UCI Repository of machine learning databases* [*www.ics.uci.edu/∼mlearn/MLRepository.html*]. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[40] KOHAVI, R., *Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid*, em **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**, 1996.

[41] FANTY, M.; COLE, R., *Spoken letter recognition*, em **Advances in Neural Information Processing Systems 3** (LIPPMAN, R. P.; MOODY, J.; TOURETZKY, D. S., eds.), (San Mateo, CA), Morgan Kaufmann, 1991.

[42] MCCALLUM, A. K., *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering* [*http://www.cs.cmu.edu/~mccallum/bow*]. 1996.

[43] JAAKKOLA, T.; HAUSSLER, D., *Exploiting generative models in discriminative classifiers*, 1998.

[44] DEMPSTER, A. P.; LAIRD, N. M.; ; RUBIN, D. B., *Maximum-Likelihood from Incomplete Data via EM Algorithm*, **J. Royal Statist. Soc. Ser. B.**, n. 39, 1977.

[45] REDNER, R.; WALKER, H., *Mixture Densities, Maximum Likelihood and the EM Algorithm*, **SIAM Review**, n. 26(2), 1984.

[46] GHAHRAMAMI, Z.; JORDAN, M., *Learning from Incomplete Data*, rel. tec., MIT AI Lab, agosto 1995.

[47] PEARL, J., **Probabilistic Reasoning in Intelligent Systems**. Morgan-Kaufman, 1988.

[48] FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M., *Bayesian Network Classifiers*, **Machine Learning**, v. 29, n. 2, p. 131–163, 1997.

[49] PLATT, J., *Sequential minimal optimization: A fast algorithm for training support vector machines*, rel. tec., Microsoft Research, 1998.

[50] HYVÄRINEN, A., *Survey on Independent Component Analysis*, **Neural Computing Surveys**, v. 02, p. 94–128, 1999.

APÊNDICES

Apêndice A

Detalhamento dos resultados apresentados no Capítulo 5

Este apêndice lista os resultados do Capítulo 5 com maior detalhamento, incluindo respectivos desvios-padrões. Os resultados não-relevantes são desconsiderados deste detalhamento.

Note que os números aqui apresentados são absolutos, ou seja, não são mais relativos à "SVM trivial".

Tabela A.1. Detalhamento da Figura 5.1: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* RBF.

PCs/kPCA	Linear Lab	Linear LUL	Poly(2) Lab	Poly(2) LUL
2 PCs	$55,71 \pm 8,36$	$58,44 \pm 10,75$	$53,40 \pm 5,36$	$58,86 \pm 10,03$
6 PCs	$57,08 \pm 8,53$	$64,50 \pm 10,38$	$57,09 \pm 7,31$	$63,27 \pm 8,45$
10 PCs	$61,79 \pm 8,87$	$67,40 \pm 10,84$	$62,39 \pm 6,85$	$64,79 \pm 7,89$
14 PCs	$68,26 \pm 8,07$	$67,85 \pm 8,94$	$68,10 \pm 8,48$	$67,45 \pm 9,62$
18 PCs	$69,40 \pm 7,84$	$67,85 \pm 8,50$	$73,52 \pm 7,52$	$69,62 \pm 9,18$
22 PCs	$68,97 \pm 7,98$	$68,79 \pm 8,09$	$74,08 \pm 8,03$	$68,53 \pm 6,25$
26 PCs	$68,98 \pm 7,98$	$69,40 \pm 8,44$	$74,10 \pm 8,03$	$69,56 \pm 8,30$
30 PCs	$68,98 \pm 7,98$	$67,71 \pm 7,95$	$74,07 \pm 8,03$	$69,86 \pm 8,50$

Tabela A.2. Detalhamento da Figura 5.2: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* linear.

PCs/kPCA	Linear Lab	Linear LUL	Poly(2) Lab	Poly(2) LUL
2 PCs	$63,28 \pm 9,53$	$69,79 \pm 10,49$	$60,89 \pm 10,47$	$59,97 \pm 10,22$
6 PCs	$69,90 \pm 8,62$	$71,19 \pm 6,39$	$69,61 \pm 8,53$	$71,47 \pm 4,96$
10 PCs	$68,48 \pm 6,60$	$71,70 \pm 6,31$	$69,65 \pm 5,17$	$70,13 \pm 5,83$
14 PCs	$66,15 \pm 8,00$	$72,23 \pm 7,54$	$62,68 \pm 10,38$	$72,08 \pm 5,46$
18 PCs	$63,61 \pm 7,89$	$71,96 \pm 6,21$	$55,31 \pm 5,81$	$71,63 \pm 6,71$
22 PCs	$67,42 \pm 6,63$	$72,02 \pm 5,22$	$56,28 \pm 6,19$	$70,32 \pm 6,16$
26 PCs	$67,43 \pm 6,63$	$70,82 \pm 5,97$	$56,29 \pm 6,18$	$70,99 \pm 7,49$
30 PCs	$67,41 \pm 6,62$	$68,91 \pm 7,54$	$56,28 \pm 6,19$	$71,16 \pm 7,70$

Tabela A.3. Continuação do detalhamento da Figura 5.2: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* linear.

PCs/kPCA	RBF Lab	RBF LUL
2 PCs	$57,77 \pm 6,44$	$71,92 \pm 9,36$
6 PCs	$66,28 \pm 4,60$	$75,59 \pm 4,94$
10 PCs	$71,09 \pm 8,12$	$77,05 \pm 5,34$
14 PCs	$70,12 \pm 5,41$	$74,74 \pm 6,10$
18 PCs	$69,28 \pm 7,68$	$73,94 \pm 6,19$
22 PCs	$69,28 \pm 8,31$	$73,88 \pm 7,17$
26 PCs	$69,28 \pm 8,31$	$74,05 \pm 6,63$
30 PCs	$69,28 \pm 8,30$	$73,20 \pm 7,04$

Tabela A.4. Detalhamento da Figura 5.3: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *20 Newsgroups* e SVM de *kernel* polinomial 2.

PCs/kPCA	Linear Lab	Linear LUL	RBF Lab	RBF LUL
2 PCs	$64,01 \pm 8,91$	$66,54 \pm 10,21$	$58,85 \pm 6,72$	$72,58 \pm 7,16$
6 PCs	$64,53 \pm 8,38$	$63,86 \pm 5,85$	$63,93 \pm 6,69$	$73,78 \pm 6,24$
10 PCs	$61,76 \pm 7,90$	$66,31 \pm 6,28$	$68,81 \pm 8,36$	$73,18 \pm 8,34$
14 PCs	$59,14 \pm 8,82$	$65,21 \pm 8,29$	$68,19 \pm 6,27$	$71,29 \pm 8,66$
18 PCs	$58,45 \pm 7,27$	$65,72 \pm 9,46$	$68,97 \pm 8,17$	$71,81 \pm 7,94$
22 PCs	$60,12 \pm 8,07$	$65,64 \pm 9,54$	$68,76 \pm 8,57$	$72,34 \pm 8,22$
26 PCs	$60,12 \pm 8,07$	$64,80 \pm 9,25$	$68,75 \pm 8,56$	$72,60 \pm 7,46$
30 PCs	$60,12 \pm 8,06$	$64,11 \pm 9,28$	$68,75 \pm 8,56$	$71,74 \pm 7,46$

Tabela A.5. Detalhamento da Figura 5.4: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Spam* e SVM de *kernel* RBF.

PCs/kPCA	Linear Lab	Linear LUL
2 PCs	$75,90 \pm 7,71$	$79,45 \pm 5,73$
4 PCs	$80,55 \pm 5,98$	$81,86 \pm 4,83$
6 PCs	$79,75 \pm 7,58$	$82,75 \pm 4,03$
8 PCs	$79,52 \pm 6,49$	$83,07 \pm 3,43$
10 PCs	$79,58 \pm 7,28$	$81,15 \pm 4,94$
12 PCs	$79,12 \pm 7,31$	$80,68 \pm 5,12$

Tabela A.6. Detalhamento da Figura 5.5: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Spam* e SVM de *kernel* linear.

PCs/kPCA	Linear Lab	Linear LUL	RBF Lab	RBF LUL
2 PCs	$78,61 \pm 7,42$	$81,48 \pm 5,88$	$77,05 \pm 6,49$	$83,29 \pm 5,21$
4 PCs	$79,45 \pm 6,85$	$79,13 \pm 6,56$	$79,99 \pm 6,57$	$82,62 \pm 4,45$
6 PCs	$77,29 \pm 6,20$	$74,48 \pm 9,17$	$78,88 \pm 6,82$	$82,35 \pm 4,83$
8 PCs	$76,41 \pm 5,48$	$67,91 \pm 8,21$	$80,15 \pm 5,11$	$80,56 \pm 4,44$
10 PCs	$76,11 \pm 7,36$	$69,90 \pm 7,48$	$79,61 \pm 6,60$	$78,07 \pm 5,47$
12 PCs	$73,86 \pm 6,54$	$71,24 \pm 8,40$	$77,80 \pm 6,67$	$76,54 \pm 5,90$

Tabela A.7. Detalhamento da Figura 5.6: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Spam* e SVM de *kernel* polinomial 2.

PCs/kPCA	Linear Lab	Linear LUL	RBF Lab	RBF LUL
2 PCs	$70,98 \pm 9,28$	$71,94 \pm 9,09$	$76,67 \pm 7,14$	$82,92 \pm 5,96$
4 PCs	$63,56 \pm 6,07$	$65,25 \pm 9,71$	$80,21 \pm 6,72$	$82,54 \pm 4,38$
6 PCs	$63,42 \pm 5,70$	$64,01 \pm 9,16$	$78,56 \pm 5,94$	$81,19 \pm 5,10$
8 PCs	$62,57 \pm 4,47$	$64,02 \pm 8,80$	$79,35 \pm 5,48$	$77,59 \pm 4,80$
10 PCs	$61,96 \pm 6,72$	$62,83 \pm 9,72$	$79,23 \pm 6,54$	$75,22 \pm 5,59$
12 PCs	$61,51 \pm 5,10$	$62,22 \pm 7,70$	$77,04 \pm 6,50$	$75,47 \pm 5,82$

Tabela A.8. Detalhamento da Figura 5.7: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Adult* e SVM de *kernel* RBF.

PCs/kPCA	Linear Lab	Linear LUL
1 PCs	$58,28 \pm 8,86$	$62,13 \pm 9,79$
3 PCs	$61,54 \pm 8,15$	$60,34 \pm 7,05$
5 PCs	$61,07 \pm 7,22$	$58,80 \pm 7,21$
7 PCs	$62,98 \pm 7,31$	$60,85 \pm 7,55$
9 PCs	$65,81 \pm 4,65$	$62,04 \pm 6,56$

Tabela A.9. Detalhamento da Figura 5.8: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Adult* e SVM de *kernel* linear.

PCs/kPCA	Linear Lab	Linear LUL	RBF Lab	RBF LUL
1 PCs	$61,79 \pm 9,19$	$67,16 \pm 9,31$	$58,46 \pm 8,06$	$64,56 \pm 8,75$
3 PCs	$64,52 \pm 8,06$	$66,25 \pm 7,89$	$61,76 \pm 7,16$	$63,61 \pm 8,15$
5 PCs	$64,86 \pm 7,14$	$63,38 \pm 8,14$	$62,83 \pm 6,82$	$64,77 \pm 6,60$
7 PCs	$63,83 \pm 7,26$	$61,86 \pm 8,51$	$65,57 \pm 7,508$	$61,52 \pm 8,55$
9 PCs	$63,90 \pm 4,23$	$58,91 \pm 7,91$	$64,06 \pm 4,70$	$59,61 \pm 7,84$

Tabela A.10. Detalhamento da Figura 5.9: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Adult* e SVM de *kernel* polinomial 2.

PCs/kPCA	Linear Lab	Linear LUL	RBF Lab	RBF LUL
1 PCs	$60,03 \pm 8,28$	$65,39 \pm 7,71$	$58,96 \pm 8,74$	$65,48 \pm 8,96$
3 PCs	$60,79 \pm 5,89$	$54,37 \pm 5,87$	$62,60 \pm 7,20$	$64,42 \pm 6,83$
5 PCs	$58,12 \pm 7,71$	$55,30 \pm 6,98$	$63,34 \pm 7,36$	$63,14 \pm 7,16$
7 PCs	$58,30 \pm 6,92$	$56,62 \pm 7,42$	$65,87 \pm 7,66$	$61,26 \pm 9,00$
9 PCs	$57,49 \pm 7,06$	$57,69 \pm 7,83$	$63,32 \pm 4,76$	$60,16 \pm 7,85$

Tabela A.11. Detalhamento da Figura 5.10: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Isolet* e SVM de *kernel* RBF.

PCs/kPCA	Linear Lab	Linear LUL	Poly(2) Lab	Poly(2) LUL
2 PCs	$54,26 \pm 4,42$	$55,94 \pm 5,82$	$53,94 \pm 4,55$	$58,18 \pm 6,40$
6 PCs	$59,52 \pm 5,43$	$57,61 \pm 3,80$	$57,48 \pm 3,93$	$57,32 \pm 4,73$
10 PCs	$59,56 \pm 5,83$	$59,58 \pm 5,95$	$58,19 \pm 5,56$	$57,68 \pm 5,96$
14 PCs	$59,81 \pm 5,93$	$59,20 \pm 6,14$	$59,31 \pm 5,34$	$59,81 \pm 6,35$
18 PCs	$61,31 \pm 5,93$	$60,09 \pm 5,69$	$60,35 \pm 6,29$	$59,86 \pm 5,66$
22 PCs	$61,24 \pm 5,94$	$59,85 \pm 5,72$	$60,63 \pm 6,31$	$59,62 \pm 6,09$

Tabela A.12. Detalhamento da Figura 5.11: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Isolet* e SVM de *kernel* linear.

PCs/kPCA	Linear Lab	Linear LUL	Poly(2) Lab	Poly(2) LUL
2 PCs	$53,76 \pm 3,81$	$56,82 \pm 5,89$	$57,38 \pm 5,05$	$58,72 \pm 6,16$
6 PCs	$59,96 \pm 3,50$	$57,74 \pm 5,80$	$58,64 \pm 5,78$	$58,31 \pm 5,78$
10 PCs	$57,44 \pm 6,31$	$57,51 \pm 5,23$	$56,78 \pm 7,08$	$57,73 \pm 7,08$
14 PCs	$59,32 \pm 5,56$	$57,86 \pm 5,76$	$59,33 \pm 5,04$	$57,93 \pm 5,04$
18 PCs	$61,18 \pm 5,90$	$59,22 \pm 5,40$	$60,14 \pm 5,52$	$58,24 \pm 5,52$
22 PCs	$61,26 \pm 5,93$	$58,88 \pm 5,30$	$60,45 \pm 5,89$	$58,75 \pm 5,89$

Tabela A.13. Detalhamento da Figura 5.12: resultados dos testes a partir de *split learning* com SVM, utilizando a base de dados *Isolet* e SVM de *kernel* polinomial 2.

PCs/kPCA	RBF Lab	RBF LUL
2 PCs	$55,90 \pm 4,76$	$56,85 \pm 5,43$
6 PCs	$55,71 \pm 2,24$	$55,70 \pm 5,87$
10 PCs	$57,97 \pm 5,70$	$57,66 \pm 6,46$
14 PCs	$59,73 \pm 6,35$	$59,18 \pm 6,22$
18 PCs	$59,52 \pm 6,20$	$58,79 \pm 5,84$
22 PCs	$59,70 \pm 6,41$	$58,74 \pm 6,22$

Apêndice B

LUL-SVM: equacionamento, experimentação e resultados

O treinamento semi-supervisionado (Capítulo 2) permite o uso de bases semi-rotuladas para a aprendizagem de classificadores, mesmo considerando-se que em alguns casos a inclusão de dados não-rotulados pode prejudicar a acurácia do classificador (Seção 2.1). Algumas metodologias foram listadas como sendo eficientes em utilizar dados não-rotulados para tarefas de aprendizagem diretamente semi-supervisionada, como SVMs semi-supervisionadas que implementam transdução (TSVM e S^3VM).

Será visto que é possível utilizar bases semi-rotuladas em SVMs sem fazer uso da idéia de transdução, a partir de algum método *gerativo* que estime a distribuição $p(y, x)$ do modelo que se deseja aprender representado pela base semi-rotulada. O modelo de $p(y, x)$ será então posteriormente dado à SVM semi-supervisionada, que poderá utilizar $p(y|x)$ como "rótulos imprecisos" para os dados não-rotulados.

A Seção B.1 apresenta e formaliza uma das idéias originadas durante o período desse trabalho de mestrado: *LUL-SVM*. Posteriormente, a Seção B.2 listará os modelos probabilísticos utilizados para estimar $p(y, x)$, utilizado pela LUL-SVM, bem como o algoritmo de otimização utilizado para definir os parâmetros destes modelos: algoritmo *Expectation-Maximization*. A Seção B.3 apresenta o classificador que será considerado para comparações, em termos de acurácia de clas-

sificação, com a LUL-SVM: *classificador de Bayes*. Finalmente, as Seções B.4 e B.5 apresentam e discutem, respectivamente, os experimentos realizados e os resultados obtidos com a LUL-SVM e bases públicas da Internet.

B.1 SVM diretamente semi-supervisionada LUL-SVM

A formulação da LUL-SVM foi desenvolvida conjuntamente pelo autor e seu orientador.

A idéia contida na Equação (4.7) será utilizada para a formulação da LUL-SVM. Por isso, convém repetir a seguinte formulação de um SVC:

$$\min \|\beta\|^2/2 + \Phi \sum_{i=1}^{n} H(1 - y_i(x_i^T\beta + \beta_o)), \tag{B.1}$$

na qual $H(z) = max(1 - z, 0)$ (função perda ou *hinge loss*) e Φ é uma constante.

De outra forma, a principal idéia da Equação (B.1) é obter um classificador que minimize a seguinte *perda esperada*:

$$L = \int (H(1 - Y(X^T\beta + \beta_o)) + \Phi'\|\beta\|^2)p(X, Y)d(X, Y). \tag{B.2}$$

É possível reescrever a Equação (B.2) da seguinte maneira:

$$\begin{aligned} L &= \Phi'\|\beta\|^2 + \\ &+ \int H(1 - (X^T\beta + \beta_o))p(Y = +1|X)p(X)dX + \\ &+ \int H(1 + (X^T\beta + \beta_o))p(Y = -1|X)p(X)dX, \end{aligned} \tag{B.3}$$

e fazer a seguinte aproximação:

$$L \approx \Phi'\|\beta\|^2 + \frac{1}{n}\sum_{i=1}^{n} H(1-(x_i^T\beta+\beta_o))p(y_i = +1|x_i) + H(1+(x_i^T\beta+\beta_o))p(y_i = -1|x_i). \tag{B.4}$$

Minimizar a perda L é equivalente a otimizar o seguinte funcional:

$$\min \frac{\|\beta\|^2}{2} + \Phi'' \sum_{i=1}^{n} (\xi_i' p(y_i = +1|x_i) + \xi_i'' p(y_i = -1|x_i)), \tag{B.5}$$

tal que,

$$1 - (x_i^T \beta + \beta_o) \leq \xi'_i, \quad \xi'_i \geq 0$$
$$1 + (x_i^T \beta + \beta_o) \leq \xi''_i, \quad \xi''_i \geq 0.$$

Entretanto, os funcionais de otimização derivados estão escritos para construir SVCs. Para uma SVM, o seguinte funcional pode ser escrito, ainda com o mesmo objetivo de se minimizar a perda L dada pela Equação (B.6):

$$\max \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \lambda_i \lambda_{i'} x_i^T x_{i'} + \sum_{i=1}^{n} \lambda_i, \tag{B.6}$$

tal que,

$$0 \leq \lambda'_i \leq \gamma, \quad 0 \leq \lambda''_i \leq \gamma,$$

na qual $\lambda_i = \lambda'_i p(y_i = +1|x_i) - \lambda''_i p(y_i = -1|x_i)$.

Com isso, usa-se um gerador de modelos probabilísticos, como MoG (mistura de gaussianas)[1] (Seção B.2.2) ou redes bayesianas (Seção B.2.3), para se obter valores aproximados para $p(y_i = +1|x_{ul,i})$ e $p(y_i = -1|x_{ul,i})$. Dessa forma, dados previamente não-rotulados podem ser fornecidos à LUL-SVM (Equação B.6), de acordo com os parâmetros $p(y_i|x_i)$. Essa SVM então está apta a ser treinada de forma diretamente *semi-supervisionada*.

A Figura B.1 ilustra o algoritmo proposto para a LUL-SVM.

B.2 Estimadores semi-supervisionados de modelos probabilísticos

Como visto na Seção B.1, a LUL-SVM precisa da distribuição $p(y|x)$ para implementar treinamento semi-supervisionado, sendo que $p(y|x)$ pode ser visto como "rótulos imprecisos". Assim, para o treinamento da LUL-SVM foram utilizados dois modelos para $p(y|x)$: ora dizia-se que $p(y|x)$ era formado por uma *mistura de gaussianas*, ora por uma *rede bayesiana*. Os parâmetros desses supostos modelos

[1] Em inglês, *mixture of gaussians*.

Algoritmo 2: *LUL-SVM*

Entrada: base de dados rotulada $\{X_l, Y_l\}$, base de dados não-rotulada X_{ul}, modelo probabilístico $P(X, Y)$ da base de dados completa $B_{lul} = \{\{X_l, Y_l\}, X_{ul}\}$

Saída: classificador $c_{dir}(.)$ aprendido de forma semi-supervisionada

01. Encontrar modelo probabilístico de B_{lul}

02. Calcular $p(y_i|x_{ul,i})$ para todos os vetores não-rotulados

03. Aprender o classificador $c_{dir}(.)$ (SVM diretamente semi-supervisionada) de acordo com a Equação (B.6) e a partir da base B_{lul}, sendo que os dados não-rotulados de B_{lul} estarão probabilisticamente rotulados com $p(y_i|x_{ul,i})$

Figura B.1. Algoritmo para treinamento da LUL-SVM.

para $p(y|x)$ eram aprendidos através do algoritmo de otimização *Expectation-Maximization.*

Nas próximas seções, serão apresentados o algoritmo de otimização *Expectation-Maximization* e os modelos probabilísticos mistura de gaussianas e rede bayesiana.

B.2.1 Algoritmo *Expectation-Maximization* (EM)

Suponha uma densidade de probabilidades dada pela distribuição $p(x|\Theta)$, parametrizada pelos parâmetros Θ [2]. Juntamente, tem-se um conjunto de vetores $X = \{x_1, \ldots, x_n\}$, amostrados de forma *i.i.d. (independente e identicamente distribuída)* da distribuição $p(.)$. Dessa forma, a densidade de probabilidades pode ser dada por:

$$p(X|\Theta) = \prod_{i=1}^{n} p(x_i|\Theta) = L(\Theta|X). \tag{B.7}$$

A função $L(\Theta|X)$ é chamada *likelihood* (verossimilhança) dos parâmetros Θ dado X. No problema de máxima verossimilhança, o objetivo é encontrar Θ que maximize $L(.)$. Em outras palavras,

$$\Theta^* = \arg\max_{\Theta} L(\Theta|X). \tag{B.8}$$

A partir desse ponto é que se considera o algoritmo *Expectation-Maximization* (EM)[3]. O EM [44, 45, 46] é um algoritmo de otimização utilizado para encontrar a estimação de máxima verossimilhança dos parâmetros de uma suposta distribuição (Equação B.8). Como era requerido um modelo de $p(y|x)$ na aprendizagem da LUL-SVM, o EM foi utilizado para encontrar os parâmetros desse modelo: ou seja, o EM estimava $p(y|x, \Theta)$, sendo que Θ ora representava os parâmetros de uma mistura de gaussianas, ora representava os parâmetros de uma rede bayesiana.

[2] Por exemplo, dado que $p(.)$ é um conjunto de gaussianas, Θ seria o conjunto de médias e matrizes de covariância dessas gaussianas.

[3] Em português, maximização de expectância.

B.2.2 MoG: mistura de gaussianas

Originalmente, o MoG[4] é um método não-supervisionado de aprendizagem (como citado na Seção 2.1). Entretanto, o equacionamento apresentado a seguir, derivado pelo autor e seu orientador, adapta o algoritmo para considerar também dados rotulados.

Considere uma tarefa de classificação binária, $Y = \{+1, -1\}$, um conjunto rotulado $\{X_l, Y_l\} = \{x_{l,1}, \ldots, x_{l,n_l}, y_{l,1}, \ldots, y_{l,n_l}\}$ de vetores-coluna $x_{l,i}$ e rótulos $y_{l,i}$, e um conjunto não-rotulado $X_{ul} = \{x_{ul,1}, \ldots, x_{ul,n_{ul}}\}$ de vetores-coluna $x_{ul,i'}$. Assume-se também que a densidade de probabilidades $p(x|y)$ é dada por uma mistura de gaussianas:

$$p(x|y=+1) = \sum_{l_1=1}^{M_1} \alpha_{l_1} p_{l_1}(x|y=+1), \tag{B.9}$$

e

$$p(x|y=-1) = \sum_{l_2=1}^{M_2} \alpha_{l_2} p_{l_2}(x|y=-1), \tag{B.10}$$

nas quais $p_l(.)$ é dada por uma distribuição gaussiana $\mathcal{G}_l(\mu_l, Cov_l)$ de parâmetros μ_l (média) e Cov_l (matriz de covariância), e $\alpha_l = p(p_l(.) = \mathcal{G}_l(\mu_l, Cov_l))$.

Dessa forma, por regra de Bayes:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x|y=+1)p(y=+1) + p(x|y=-1)p(y=-1)}. \tag{B.11}$$

O modelo probabilístico é chamado de mistura de gaussianas por ser formado pela "mistura" de gaussianas que formam $p(x|y=+1)$ e $p(x|y=-1)$.

A mistura de gaussianas é então otimizada através do algoritmo EM (Seção B.2.1), sendo que o vetor Θ é composto pelos parâmetros do modelo, ou seja, $\Theta = (\beta_1, \beta_2, \alpha_{l_1}, \alpha_{l_2}, \mu_{l_1}, \mu_{l_2}, Cov_{l_1}, Cov_{l_2}),\ \forall l_1,\ \forall l_2$, sendo que β_1 e β_2 representam $p(y=+1)$ e $p(y=-1)$, respectivamente.

O aprendizado desse modelo é *diretamente semi-supervisionado*, pois são considerados os conjuntos $\{X_l, Y_l\}$ e X_{ul} concomitantemente: caso $x = x_{l,i}$, $p(y = y_{l,i}|x_{l,i}) = 1$; caso $x = x_{ul,i'}$, $p(y = y_{ul,i'}|x_{ul,i'})$ é estimado pela Equação B.11. Assim, os "rótulos imprecisos" para $x_{ul,i'}$ que serão utilizados pela LUL-SVM podem ser estimados.

[4]Sigla para *mixture of gaussians*.

B.2.3 Redes bayesianas

Redes bayesianas são grafos direcionados acíclicos (DAG, do inglês *directed acyclic graph*) associados a um conjunto de distribuições de probabilidade [47].

Considere um DAG, onde cada nó é associado a uma variável X_k. Denote por $\text{pa}(X_k)$ o conjunto de variáveis que são pais diretos da variável X_k na rede. Cada variável X_k é associada a uma distribuição $p(X_k|\text{pa}(X_k))$ (uma variável sem pais é associada à distribuição marginal $p(X_k)$).

A propriedade básica em redes bayesianas é que toda variável X_k é independente de todos os antecessores de X_k, condicional nos pais de X_k. Isso garante que a distribuição conjunta seja definida por [47]:

$$p(\mathbf{X}) = \prod_i p(X_k|\text{pa}(X_k))\,, \tag{B.12}$$

na qual $\mathbf{X}$ é o conjunto de todas as variáveis. Essa expressão é importante por garantir que uma distribuição conjunta, potencialmente complexa, possa ser representada através do produto de blocos menores.

Foram utilizados principalmente dois tipos específicos de redes bayesianas: *naïve Bayes (NB)* [21] e *TAN (tree-augmented naïve Bayes)* [48]. *Naïve Bayes* são classificadores bayesianos nos quais o nó associado à variável "rótulo" (ou classe) é associado a todas as variáveis X_k, e nenhuma variável é associada a outra; ou seja, as variáveis são consideradas *independentes*. No caso do TAN, as variáveis são consideradas *dependentes*.

As redes bayesianas não são construídas originalmente para lidar com dados semi-rotulados. Porém, existem algoritmos de redes bayesianas que podem ser treinados a partir de bases semi-rotuladas, e que são otimizados a partir do algoritmo EM (Seção B.2.1). Dessa forma, foram utilizados TAN e NB como estimadores semi-supervisionados para o modelo probabilístico de $p(y|x)$ a ser utilizado na LUL-SVM. Especificamente, foram considerados os algoritmos citados em [21].

Maiores detalhes e aplicações interessantes de redes bayesianas podem ser encontradas em [3].

B.3 Classificador de Bayes

Após a estimação de qualquer modelo probabilístico, pode-se estimar finalmente $p(y|x)$, considerando-se que y é a variável que fornece o rótulo de x e $y \in Y = \{+1, -1\}$, e que x é uma amostra de X. A partir desta informação, a amostra x é rotulada pelo *classificador de Bayes*:

$$y^* = \arg\max_y p(y|x). \tag{B.13}$$

Dado o modelo probabilístico correto do qual a base de dados X foi amostrada, o classificador de Bayes é comprovadamente ótimo, ou seja, o classificador que possui a menor taxa de erros de classificação. Como é esperado que o modelo estimado por MoG ou por rede bayesiana aproxime o modelo gerador de X, o classificador de Bayes pode então ser utilizado para tarefas de classificação. Dessa forma, dado um modelo probabilístico aproximado, a acurácia da LUL-SVM será comparado ao do classificador de Bayes na seção seguinte.

B.4 Experimentos com a SVM semi-supervisionada LUL-SVM

Para os experimentos com a LUL-SVM, foram escolhidas algumas bases reais do repositório de dados público *UCI Database Repository* [39] e geradas algumas bases artificiais.

Estimava-se o modelo probabilístico de $p(y|x)$ para certa base, e esse era utilizado pela LUL-SVM. No entanto, para comparações com a SVM, esse modelo era utilizado também por um classificador de Bayes para classificar os dados de teste.

As peculiaridades dos procedimentos experimentais e resultados obtidos serão apresentados a seguir.

Tabela B.1. Resultados obtidos para os experimentos com MoG e LUL-SVM: base de dados *Adult*.

Base *Adult*/Caso	**L40**	**L40 UL100**	**L40 UL1000**	**L40 UL3000**
LUL-SVM (radial)	$66,88 \pm 2,74$	$72,39 \pm 1,53$	$63,34 \pm 7,17$	$69,04 \pm 8,40$
classif. de Bayes	$70,24 \pm 1,31$	$72,69 \pm 1,50$	$63,64 \pm 6,75$	$68,87 \pm 8,48$
LUL-SVM (linear)	$71,09 \pm 3,75$	$71,62 \pm 1,71$	$74,47 \pm 0,62$	$70,81 \pm 6,59$
classif. de Bayes	$71,49 \pm 1,08$	$71,15 \pm 1,57$	$72,87 \pm 0,47$	$70,73 \pm 4,54$

B.4.1 Estimando modelos probabilísticos por MoG

Neste caso, os modelos probabilísticos de $p(y|x)$ eram aprendidos por mistura de gaussianas (MoG)[5], a partir de um certo número de gaussianas desejadas.

A *porcentagem de acertos* é dada pela média entre 5 diferentes tentativas (*5-fold cross validation*, ou validação cruzada de 5 setores).

A letra 'L' representa dados rotulados, e 'UL' representa uma certa quantidade de dados não-rotulados. Assim, por exemplo, a sigla 'L20 UL1000' representa 20 dados rotulados mais 1000 dados não-rotulados

Foram utilizadas 3 bases do *UCI Repository*: *Adult*, *Spam* e *Musk*. As funções *kernel* selecionadas para a LUL-SVM foram função de base radial (RBF) e linear.

Os resultados obtidos, em porcentagem (%), estão listados nas Tabelas B.1, B.2 e B.3[6].

B.4.2 Estimando modelos probabilísticos por redes bayesianas

Estes experimentos foram feitos de maneira similar aos testes feitos na Seção B.4.1. Entretanto, a base de dados foi amostrada (artificialmente) de uma rede bayesiana TAN. Em seguida, o modelo de $p(y|x)$ era estimado a partir de um

[5] O algoritmo de MoG utilizado para os experimentos foi escrito em Matlab.

[6] Os experimentos cujos resultados foram substituídos por '——' não foram realizados devido aos tempos computacionais demasiadamente longos apresentados para o treinamento da SVM.

Tabela B.2. Resultados obtidos para os experimentos com MoG e LUL-SVM: base de dados *Spam*.

Base *Spam*/Caso	**L30**	**L30 UL100**	**L30 UL1000**	**L30 UL2850**
LUL-SVM (radial)	$77,35 \pm 0,62$	$69,56 \pm 7,52$	$70,64 \pm 6,73$	$70,92 \pm 13,82$
classif. de Bayes	$78,55 \pm 1,50$	$69,87 \pm 7,32$	$71,17 \pm 6,51$	$70,88 \pm 14,11$
LUL-SVM (linear)	$81,00 \pm 1,41$	$72,60 \pm 4,90$	$71,60 \pm 6,20$	——
classif. de Bayes	$83,10 \pm 1,55$	$68,60 \pm 4,30$	$70,40 \pm 4,80$	——

Tabela B.3. Resultados obtidos para os experimentos com MoG e LUL-SVM: base de dados *Musk*.

Base *Musk*/Caso	**L30**	**L30 UL100**	**L30 UL500**	**L30 UL1500**
LUL-SVM (radial)	$58,85 \pm 1,84$	$63,60 \pm 3,19$	$61,92 \pm 2,18$	$60,89 \pm 7,00$
classif. de Bayes	$63,43 \pm 0,61$	$63,60 \pm 3,44$	$61,68 \pm 1,94$	$61,03 \pm 6,82$
LUL-SVM (linear)	$61,05 \pm 0,77$	$57,09 \pm 6,84$	$53,94 \pm 8,02$	$55,47 \pm 8,26$
classif. de Bayes	$60,31 \pm 2,17$	$57,64 \pm 7,39$	$53,55 \pm 8,18$	$54,78 \pm 8,96$

Tabela B.4. Resultados obtidos para os experimentos com redes bayesianas (TAN e NB): modelo de $p(y|x)$ estimado por um TAN (modelo *correto).*

Modelo correto/Caso	**L100**	**L100 UL250**	**L100 UL1000**
LUL-SVM (radial ($\sigma = 1,0$))	$55,10 \pm 2,08$	$60,72 \pm 2,05$	$69,35 \pm 0,54$
LUL-SVM (radial ($\sigma = 0,5$))	$56,42 \pm 1,21$	$61,45 \pm 1,58$	$69,08 \pm 0,29$
LUL-SVM (radial ($\sigma = 0,25$))	$57,08 \pm 0,54$	$61,27 \pm 1,25$	$67,45 \pm 0,68$
LUL-SVM (linear)	$50,62 \pm 2,04$	——	$54,55 \pm 2,07$
LUL-SVM (polinomial $d = 2$)	$56,50 \pm 0,89$	$59,10 \pm 0,84$	——
classificador de Bayes	$64,78 \pm 0,46$	$76,07 \pm 2,04$	$83,64 \pm 0,98$

TAN (modelo correto), e a partir de um NB (modelo errado). Além disso, as *porcentagens de acertos* são médias entre 4 diferentes tentativas, através de 4 amostragens diferentes do TAN "gerador".

As rotinas de redes bayesianas (TAN e NB) utilizadas para estes experimentos foram as utilizadas em [21].

Os resultados[7] obtidos, em porcentagem (%), estão compilados nas Tabelas B.4 e B.5.

B.5 Discussão e conclusão dos experimentos realizados com a LUL-SVM

Os resultados obtidos na Seção B.4.2 demonstram a validade das idéias previamente apresentadas na Seção 2.1: quando o modelo de $p(y|x)$ é estimado de forma correta, a inclusão de dados não-rotulados beneficia a acurácia final do classificador; quando o modelo está errado, a adição destes dados pode beneficiar ou prejudicar o classificador. Verifica-se que a LUL-SVM consegue então fazer uso

[7] Os casos cujos resultados foram substituídos por '——', não foram realizados devido aos tempos computacionais demasiadamente longos apresentados por estes casos durante os treinamentos das SVMs.

Tabela B.5. Resultados obtidos para os experimentos com redes bayesianas (TAN e NB): modelo de $p(y|x)$ estimado por um NB (modelo *errado).*

Modelo errado/Caso	**L100**	**L100 UL250**	**L100 UL1000**
LUL-SVM (radial ($\sigma = 1,0$))	$55,10 \pm 2,08$	$57,88 \pm 1,24$	$60,10 \pm 0,74$
LUL-SVM (radial ($\sigma = 0,5$))	$56,42 \pm 1,21$	$59,72 \pm 0,91$	$59,85 \pm 0,81$
LUL-SVM (radial ($\sigma = 0,25$))	$57,08 \pm 0,54$	$60,15 \pm 1,00$	$59,52 \pm 1,52$
LUL-SVM (linear)	$50,62 \pm 2,04$	——	$51,12 \pm 1,21$
LUL-SVM (polinomial $d = 2$)	$56,50 \pm 0,89$	$58,92 \pm 0,83$	——
classificador de Bayes	$66,66 \pm 0,38$	$65,62 \pm 2,83$	$61,70 \pm 1,07$

dos dados não-rotulados em seu benefício, dado o modelo correto de $p(y|x)$. Isso demonstra sua validade como classificador de aprendizagem semi-supervisionada. Entretanto, conforme o esperado, a utilização do modelo errado trouxe prejuízos à acurácia do classificador.

De acordo com os resultados, o melhor *kernel* para a LUL-SVM é a função de base radial, cuja escolha permitiu à LUL-SVM os melhores índices de acurácia. Além disso, esse *kernel* também se mostrou a melhor opção em termos de tempo de treinamento, já que esse permitiu os menores tempos de aprendizagem para a LUL-SVM. Portanto, o *kernel* de função de base radial é o que melhor *reescreveu* a base de dados em um novo espaço de variáveis, facilitando assim a aprendizagem da LUL-SVM. Esse fato suporta a idéia de que *a maneira como a base de dados é escrita (ou reescrita) em seu espaço de variáveis é fundamental para a acurácia do classificador a aprender essa base.*

Ao se comparar a acurácia do classificador de Bayes com a LUL-SVM, observa-se que a LUL-SVM tende a manter a média de acertos do classificador de Bayes, na grande maioria dos casos, não importando se o modelo probabilístico de $p(x|y)$ dado está correto ou não. Esse fato originou uma questão importante sobre a validade da LUL-SVM:

> *Será mesmo vantajoso estimar um modelo para $p(y|x)$, e utilizá-lo no aprendizado de uma LUL-SVM, sendo que a escolha de um classificador de Bayes após a estimação do modelo resulta em acurácia semelhante ao da LUL-SVM?*

Uma análise de custo-benefício entre a LUL-SVM e o classificador de Bayes aponta para uma escolha definitiva desta última opção de classificador. SVMs são algoritmos computacionalmente custosos, mesmo com a utilização de rotinas otimizadas, como *sequential minimal optimization (SMO)* [49], que diminuem consideravelmente o tempo de aprendizagem destes classificadores. Já o classificador de Bayes é apenas um limiar de decisão ótimo, que necessita de um modelo probabilístico mas não necessita de qualquer tipo de aprendizagem prévia sobre o problema. Assim, a escolha de uma LUL-SVM para uso de dados não-rotulados

gerará o custo de estimação de um modelo probabilístico *mais* o custo de treinamento da LUL-SVM. No caso de um classificador de Bayes, somente o primeiro custo existirá. Dessa forma, o classificador de Bayes é menos custoso e apresenta acurácia semelhante ao da LUL-SVM, sendo então mais vantajoso.

A utilização de um classificador de Bayes basta para tomar decisões a partir do modelo de $p(y|x)$, obtido previamente de forma diretamente semi-supervisionada a partir de MoG ou redes bayesianas. A LUL-SVM, mesmo tendo sua validade comprovada, possui então uma relação custo/benefício desfavorável frente àquele classificador, e dessa forma não apresenta nenhuma evolução em termos de utilização de bases semi-rotuladas.

Porém, o tema desse trabalho é a utilização de bases semi-rotuladas para o aprendizado de SVMs; isso foi mantido na busca de novas idéias. Entretanto, a essência da busca foi modificada: em vez de se idealizar novas formas de utilização *direta* de dados não-rotulados para o treinamento de SVMs (como no treinamento semi-supervisionado da LUL-SVM), buscaram-se metodologias que agregassem um aprendizado fundamental adquirido durante a pesquisa realizada para a implementação da LUL-SVM:

> *A forma como a base de dados é escrita em seu espaço de variáveis influenciará fundamentalmente a acurácia final do classificador (escolha do* kernel*).*

Esse aprendizado foi utilizado para a idealização da metodologia de treinamento semi-supervisionado indireto *split learning* que, conforme visto no Capítulo 3, pode ser utilizada para o treinamento de SVMs a partir de bases semi-rotuladas.

Apêndice C

Artigo publicado na 22^{nd} *International Conference on Machine Learning* (ICML-2005)

O artigo anexado neste apêndice, denominado *Splitting the Unsupervised and Supervised Components of Semi-Supervised Learning*, foi publicado na 22^{nd} *International Conference on Machine Learning* (ICML-2005) – *Learning with Partially Classified Training Data Workshop* pelo autor, seu orientador (prof. Dr. Fabio Cozman) e pelo pesquisador do HP Labs (EUA) Dr. Ira Cohen. Essa conferência foi realizada em Bonn, Alemanha.

A publicação disserta sobre alguns tópicos de treinamento semi-supervisionado que embasam a conceituação da metodologia *split learning* (Capítulo 3), que é o foco do artigo. O trabalho também lista e discute resultados obtidos a partir de experimentos realizados com essa metodologia.

Vale ressaltar alguns pontos relacionados ao artigo e à dissertação de mestrado:

1. Nos experimentos do artigo, o *split learning* foi executado a partir dos métodos PCA (Seção 3.2.2) e ICA (*independent component analysis*[1]) [50], como métodos para a implementação da função $g(.)$, e SVM, como método para o classificador $c_{spl}(.)$. Nessa dissertação, o PCA foi substituído pelo kPCA

[1]Em português, análise de componentes independentes.

(método mais geral do que o PCA), e o ICA não foi mencionado já que não foram obtidos resultados expressivos quando de seu uso para execução de *split learning*.

2. A discussão do artigo que tenta relacionar *gaussianidade* dos dados e *acurácia do split learning* foi substituída pela discussão da dissertação entre *esparsidade* dos dados e *acurácia do split learning*. A relação esparsidade-acurácia se mostrou mais coerente tanto com os resultados obtidos do artigo quanto com os resultados descritos nessa dissertação.

O artigo se encontra na página seguinte[2].

[2]As margens originais do artigo foram ligeiramente modificadas devido ao formato do texto da dissertação.