Data Mining

LABDATA

Coordenação:

Prof. Dr. Adolpho Walter Pimazzi Canton

Profa. Dra. Alessandra de Ávila Montini

Disciplina: Machine Learning

Tema da Aula: Algoritmos em Python

Prof. Carlos Eduardo Martins Relvas

Currículo

- Bacharel em Estatística, Universidade de São Paulo.
- Mestre em Estatística, Universidade de São Paulo.
- Itaú, 2010-2015. Principais atividades:
  - Consultoria estatística para várias áreas do banco com foco principal em melhorias no processo de modelagem de risco de crédito.
  - De 2013 a 2015, participação do projeto Big Data do banco usando tecnologia Hadoop e diversas técnicas de machine learning. Desenvolvemos diversos algoritmos em MapReduce usando R e Hadoop streaming, criando uma plataforma de modelagem estatística no Hadoop.
- Nubank, dede 2015. Principais atividades:
  - Equipe de Data Science, responsável por toda a parte de modelagem da empresa, desde modelos de crédito a identificar motivos de atendimento.

Conteúdo da Aula

- Boosting
- Redes Neurais
- SVM

Boosting

Boosting

- Uma das ideias mais bem aceitas dos últimos anos. Consisite em criar e combinar classificadores ”fracos” para criar um classificador ”forte”.

- Utilizado tanto para classificação quanto para regressão.

- Um classificador fraco é um classificador cuja acurácia é um pouco melhor do que o chute aleatório. A ideia do boosting é criar classificadores fracos em diferentes versões dos dados, produzindo uma sequência de classificadores $G_m(x), m = 1, \ldots, M$.

Boosting

- Assim, o modelo final será dado por uma média ponderada dos votos de todos os classificadores:

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

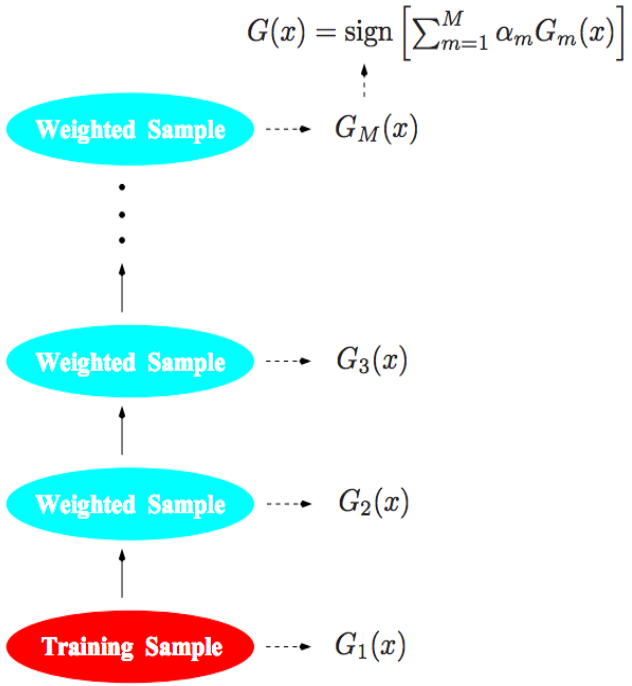


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Boosting

- Para o cálculo do peso da cada classificador no modelo final $(\alpha_1, \dots, \alpha_m)$, a ideia é que classificadores melhores tenham mais peso e classificadores piores pouco peso. Assim, dependem da acurácia do seu respectivo classificador.

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

- O boosting pode ser utilizado com vários classificadores diferentes, tendo sido mais utilizado com árvores de decisão.

Boosting

- E como modificamos os dados para cada classificador produzir um modelo diferente?
  A ideia é sempre usar todas as observações, mas atribuir pesos diferentes para observação nos diferentes classificadores.
  Inicialmente, para o primeiro modelo, fazemos $\omega_1, \dots, \omega_N$ iguais a $^1/_N$.

- Para os próximos classificadores $m = 2, 3, \dots, M$, olhamos o peso do modelo anterior e aumentamos o peso das observações que foram classificadas incorretamente. Da mesma forma, reduzimos o peso das observações classificadas corretamente.
  Assim, forçamos o classificador a acertar as observações que estão sendo consistemente mal classificadas.

Boosting

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Boosting - Exemplo

- $X_1, X_2, \ldots, X_{10}$ são variáveis aleatórias normalmente distribuídas.
- Simulamos a variável resposta como: $$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi_{10}^2(0.5), \\ -1 & \text{otherwise.} \end{cases}$$
- 2.000 (1.000 positivas e 1.000 negativas) observações na base de treino e 10.000 observações na base de teste.
- Como classificador para ser utilizado nesta simulação, usaremos um Stump (árvore de decisão com apenas dois nós finais) com boosting. Este classificador simples apresenta um erro de 45.8% na base de teste (pouco melhor que o aleatório).

Boosting - Exemplo

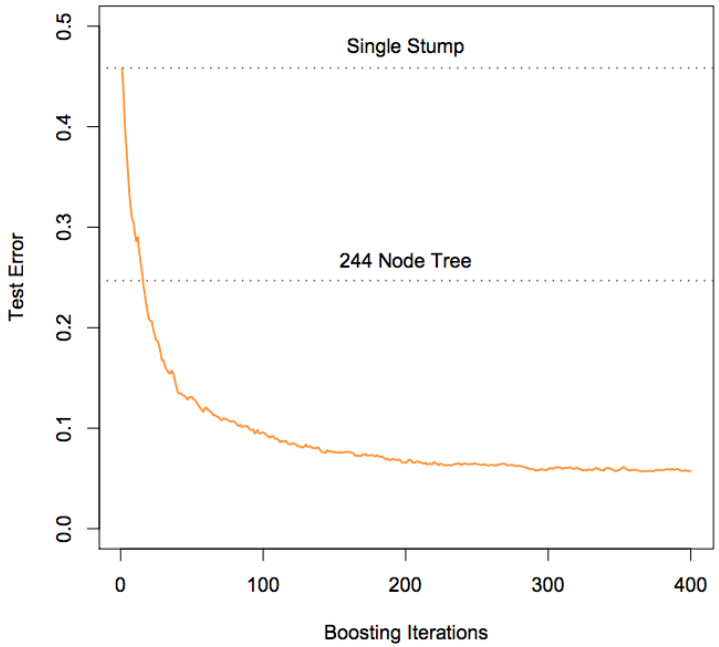


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

Por que Boosting funciona tão bem?

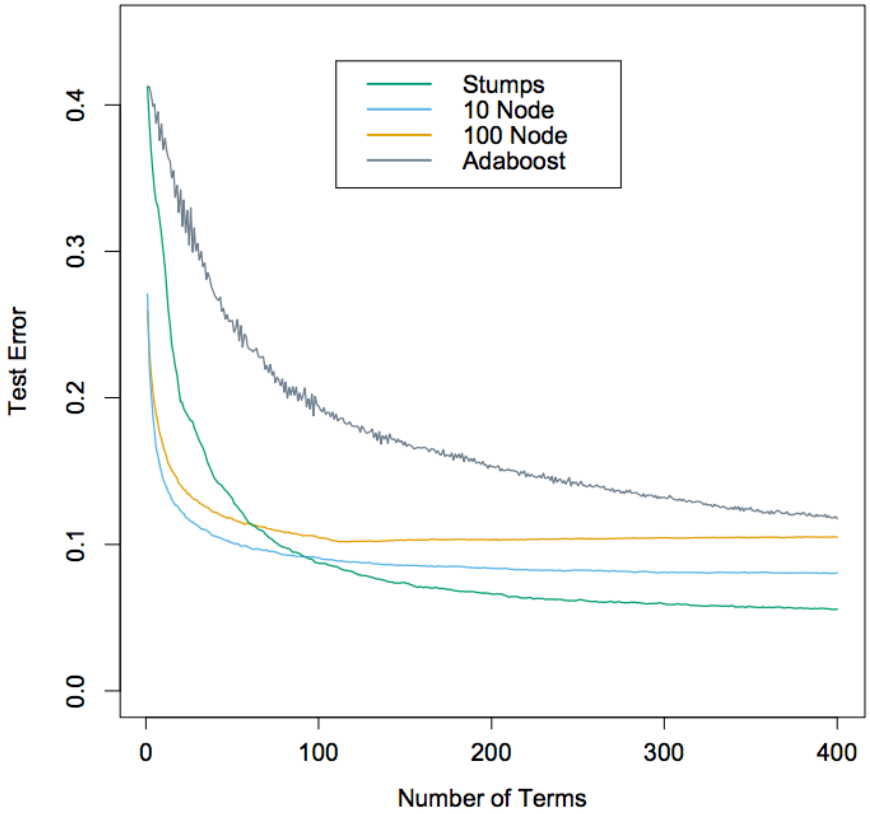
- O Boosting pode ser visto como uma expansão aditiva em um conjunto de funções bases.

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m),$$

- O que permite criar ajustes não lineares e é a mesma base de outros algoritmos, como a rede neural.

- A função de custo minimizada é a exponential.

Boosting


Stumps
10 Node
100 Node
Adaboost
Test Error
0.0
0.1
0.2
0.3
0.4
0
100
200
300
400
Number of Terms

Boosting

- Parâmetros para otimizar em boosting trees:

  - Complexidade da árvore (número de nós finais): J
  - M, o número de iterações do algoritmo. Na base de treino sempre é melhor aumentar o valor de M, o que causa overfitting.

- É recomendável otimizar estes valores por meio de cross validation ou utilizando uma base de validação.

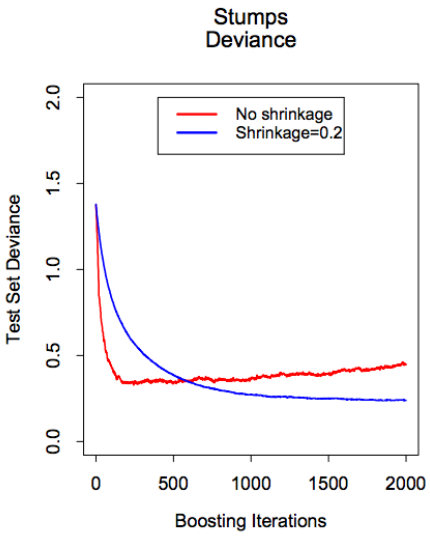
Boosting - Regularização

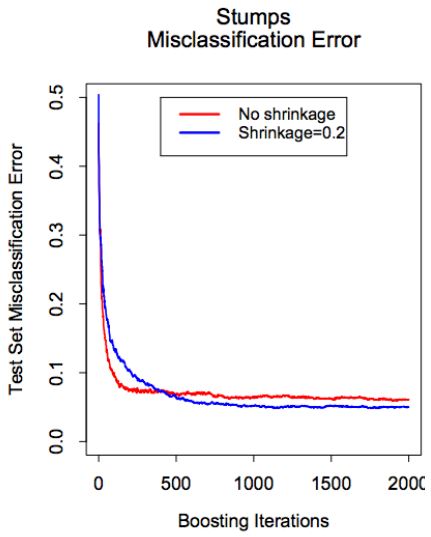
- Outra estratégia para evitar overfitting é usar um parâmetro v diminuindo o peso de cada árvore.

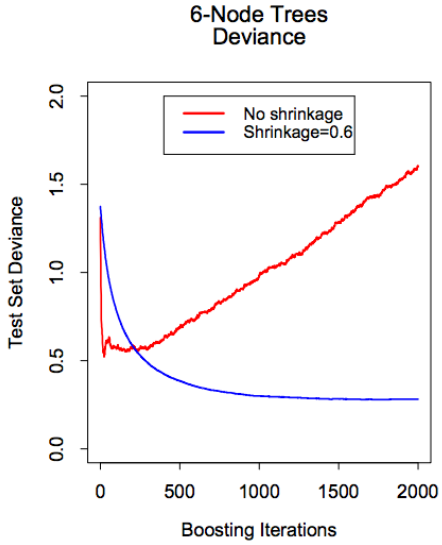
$$f_m(x) = f_{m-1}(x) + \nu \cdot \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm}).$$

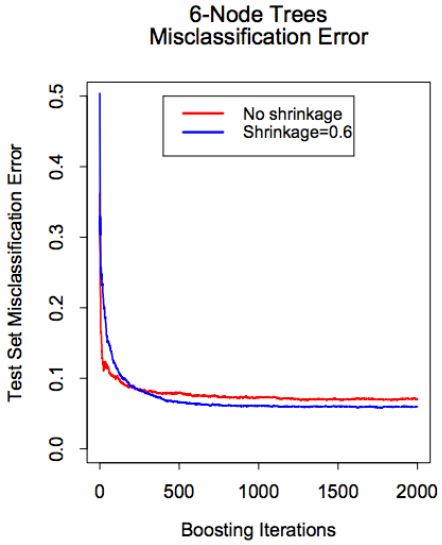
- Resultados empíricos mostram que é melhor utilizar v pequeno (< 0.1) e escolher M acompanhando os resultados em uma base de validação. Isso irá necessitar de mais processamento computacional, visto que M tende a ser maior.

Boosting - Regularização







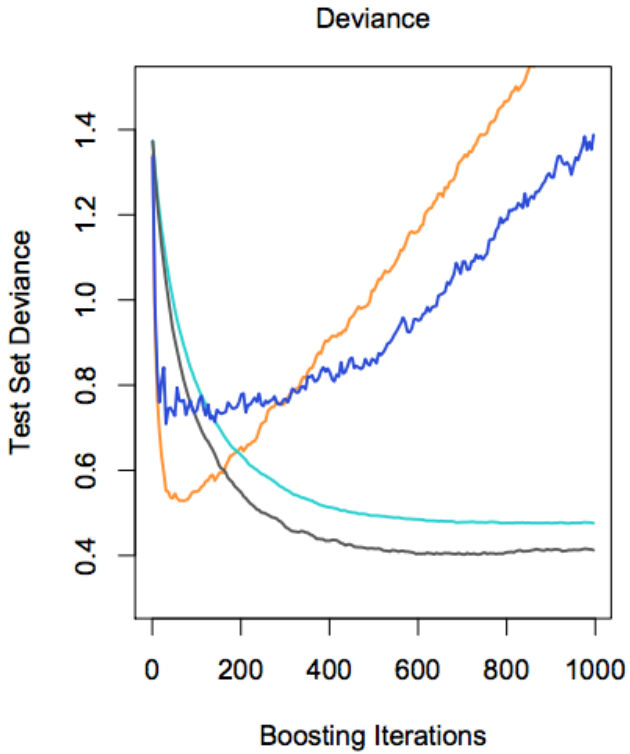


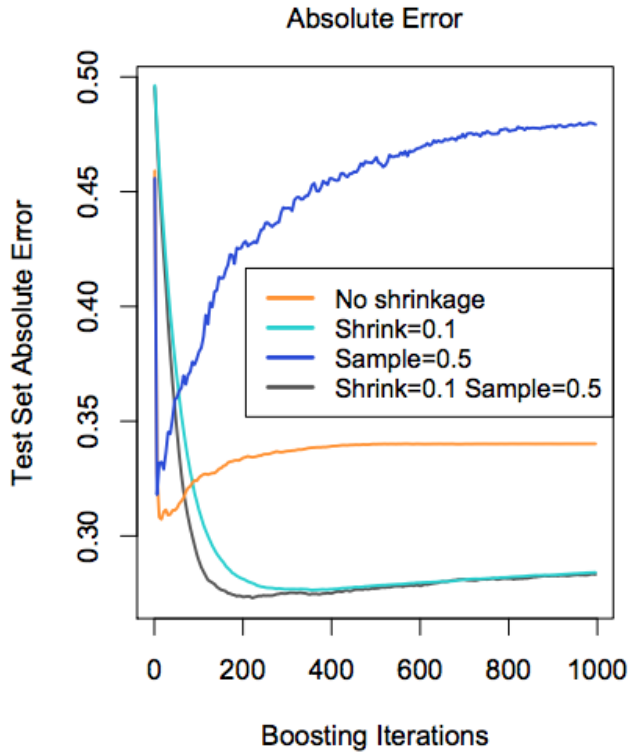
Boosting - Subsample

- Outra estratégia para otimizar o algoritmo de boosting é usar o conceito de subsample, que é muito parecido a ideia do bagging.

- Para cada classificador, ao invés de utilizarmos todas as observações, utilizamos apenas uma proporção η dos dados, escolhida aleatoriamente sem reposição.

Boosting - Subsample





Boosting - Laboratório – Regressão

Base simulada com 150 observações e 5 variáveis.

- Gastos no cartão em reais
- Idade
- Renda
- Pagamento de impostos
- Segmento

Objetivo:
Prever os Gastos no cartão com base nas outras informações.

```
> head(dados)
  Gastos_Cartao Idade Renda Impostos Segmento
1           510    35  1120       60        C
2           490    30  1120       60        C
3           470    32  1040       60        C
4           460    31  1200       60        C
5           500    36  1120       60        C
6           540    39  1360      120        C
```

Boosting - Laboratório – Regressão

Notebook de análise: Gastos_cartao.ipynb

Boosting - Laboratório – Classificação

Laboratório R - Base de Spam

- Base com 4.601 e-mails. Porcentual em que 54 palavras ou pontuações aparecem em cada e-mail. Além disso, temos o tamanho médio das palavras, tamanho da maior palavra e quantidade de palavras.

Objetivo:
Criar um detector automático de SPAM que verificará cada novo e-mail.

- Disponível em: https://archive.ics.uci.edu/ml/datasets/Spambase

Boosting - Laboratório – Classificação

Notebook de análise: Spam.ipynb

Boosting - Laboratório – Exercício

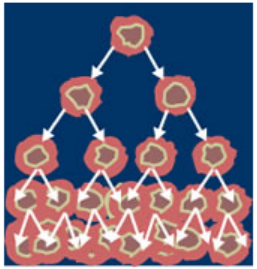
http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	390512

Boosting - Laboratório – Exercício

Base de dados (“cancer.data”) com 699 observações e 10 variáveis de pacientes com tumores. O objetivo é detectar com base em algumas informações dos tumores se é benigno ou maligno.

Variáveis:

- 1. Sample code number id number
- 2. Clump Thickness 1 - 10
- 3. Uniformity of Cell Size 1 - 10
- 4. Uniformity of Cell Shape 1 – 10
- 5. Marginal Adhesion 1 - 10
- 6. Single Epithelial Cell Size 1 - 10
- 7. Bare Nuclei 1 - 10
- 8. Bland Chromatin 1 - 10
- 9. Normal Nucleoli 1 - 10
- 10. Mitoses 1 - 10
- 11. Class: (2 for benign, 4 for malignant)

Boosting - Laboratório – Exercício

1.) Crie bases de desenvolvimento (80%) e teste (20%). Utilize seed de 42.

2.) Cheque e corriga possíveis problemas de missing.

3.) Ajuste um modelo simple de boosting com n_estimators igual a 50 e max_depth igual a 3

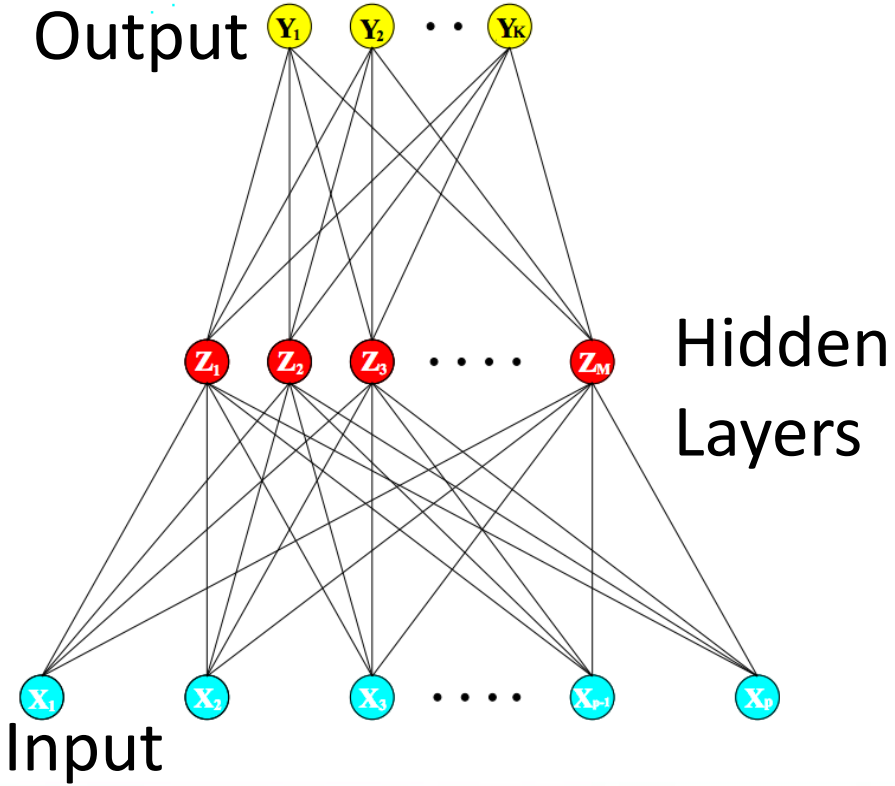
4.) Otimize os hiper-parâmetros.

5.) Avalie os resultados.

Rede Neural

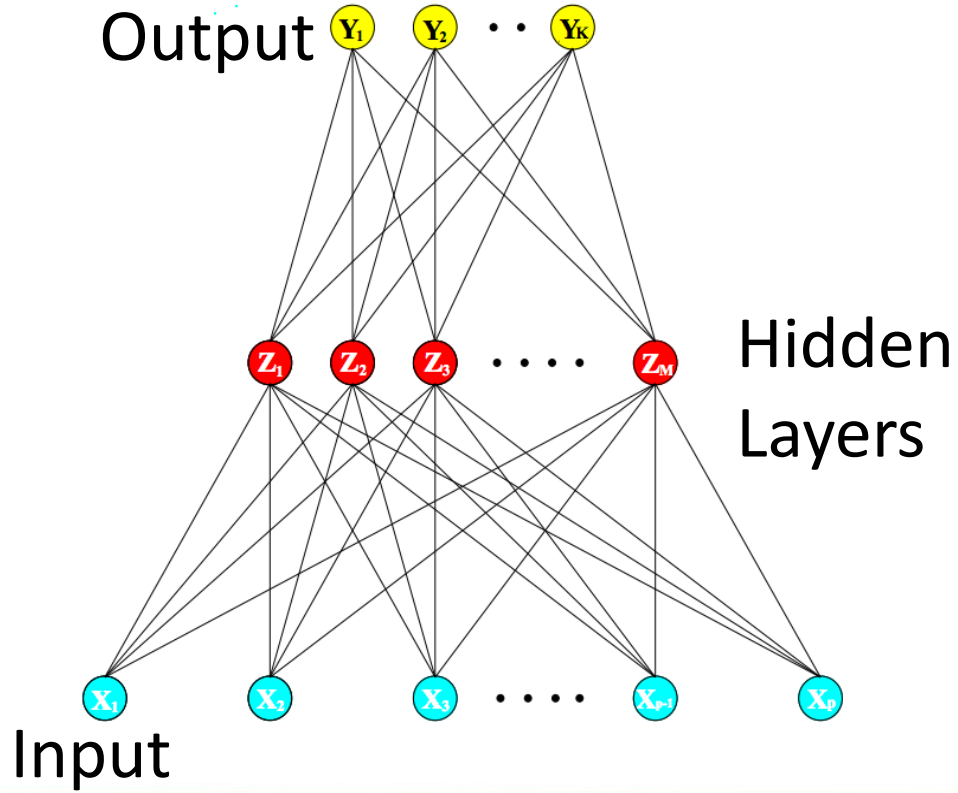
Rede Neural

- Rede neural artificial é uma técnica de aprendizado que utiliza uma estrutura matemática baseada em uma rede de funcionamento do cerébro para produzir relações não lineares.



Rede Neural

- Neste exemplo temos p variáveis explicativas (inputs), uma camada escondida e K neurônios na camada de output. Geralmente, para problemas de regressão ou classificação binária, temos K igual a 1. Em problemas com R classes, temos que K = R, e cada neurônio assim representa a probabilidade de cada classe.



Rede Neural

- Cada neurônio da camada escondida é uma combinação linear dos inputs (variáveis explicativas) e o output da rede é uma combinação linear dos neurônios da camada escondida. Assim, temos:

$$Z_i = f\left(\alpha_{0i} + \sum_{j=1}^{p} \alpha_{ji} X_j\right), i = 1, \dots, M \qquad Y_k = g\left(\beta_{0k} + \sum_{j=1}^{M} \beta_{jk} Z_j\right), k = 1, \dots, K$$

- Geralmente é utilizado a função sigmóide para f e a identidade para g, isto é:
- $f(x) = \frac{1}{(1+e^{-x})}$; $g(x) = x$.

- Para problemas de classificação, geralmente utilizamos a função softmax para g. $g_k(Y) = \frac{e^{Y_k}}{\sum_i e^{Y_i}}$.

  Estas funções são denominadas como funções de ativação.

Rede Neural

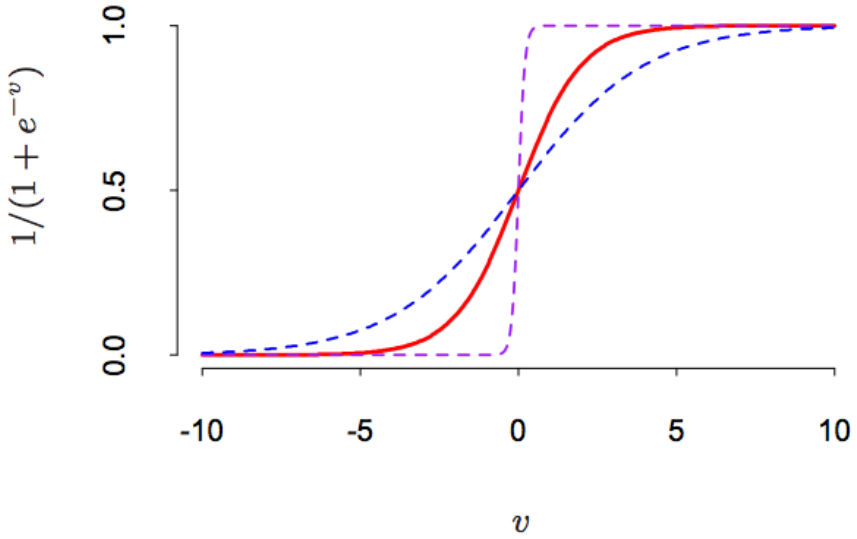


FIGURE 11.3. *Plot of the sigmoid function* $\sigma(v) = 1/(1+\exp(-v))$ *(red curve), commonly used in the hidden layer of a neural network. Included are* $\sigma(sv)$ *for* $s = \frac{1}{2}$ *(blue curve) and* $s = 10$ *(purple curve). The scale parameter* s *controls the activation rate, and we can see that large* s *amounts to a hard activation at* $v = 0$*. Note that* $\sigma(s(v - v_0))$ *shifts the activation threshold from* 0 *to* v_0*.*

Rede Neural – Estimação

- Como estimamos os seguintes parâmetros?
  - $\alpha_{0m}, \alpha_m; m = 1, 2, \ldots, M.$ M(p+1) parâmetros.
  - $\beta_{0k}, \beta_k; k = 1, 2, \ldots, K.$ K(M+1) parâmetros.

- Para regressão minimizamos a seguinte função objetivo:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$$ Soma de erros ao quadrado

- Já para classificação, minimizamos:

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i),$$ Entropia cruzada

Rede Neural – Estimação

- A minimização das funções objetivos diretamente não é recomendada, visto que é provável o *overfitting*. Por isso, utilizamos regularização, seja por um termo de penalidade ou por uma estratégia de ”early stopping”.

- O algoritmo utilizado para a minimização da função objetivo é denominado *backpropagation.*

Rede Neural – Otimizações

- **Valores iniciais:**

  É recomendado utilizar como valores iniciais valores aleatórios próximos do 0.
  O uso de valor 0 para todos faz com que o modelo não saia do lugar e o uso de valores grandes leva a soluções pobres.

- **Escala:**

  As redes neurais são bastante sensíveis a diferentes escalas nas variáveis explicativas, portanto, é recomendado padronizar todas as variáveis para média 0 e variância 1, assim todas variáveis terão o mesmo peso no processo.

Rede Neural – Otimizações

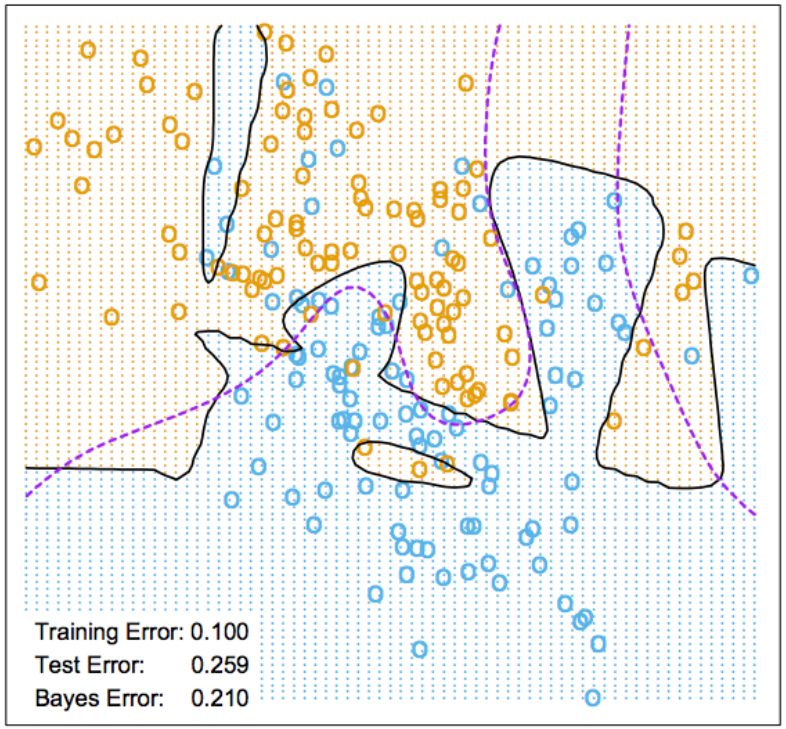
- **Overfitting:**
- Quando se usa muitos neurônios ou camadas escondidas, o *overfitting* é muito comum. Para contornar isso, podemos utilizar a técnica de *early stopping*, que consiste em parar a otimização antes de encontrar o mínimo. Podemos parar quando o resultado em uma base de validação a performance parar de melhorar.
- Outra abordagem é utilizar o chamamos de *weight decay* que consiste em minimizar $R(\theta) + \lambda J(\theta)$:

$$J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{m\ell} \alpha_{m\ell}^2$$

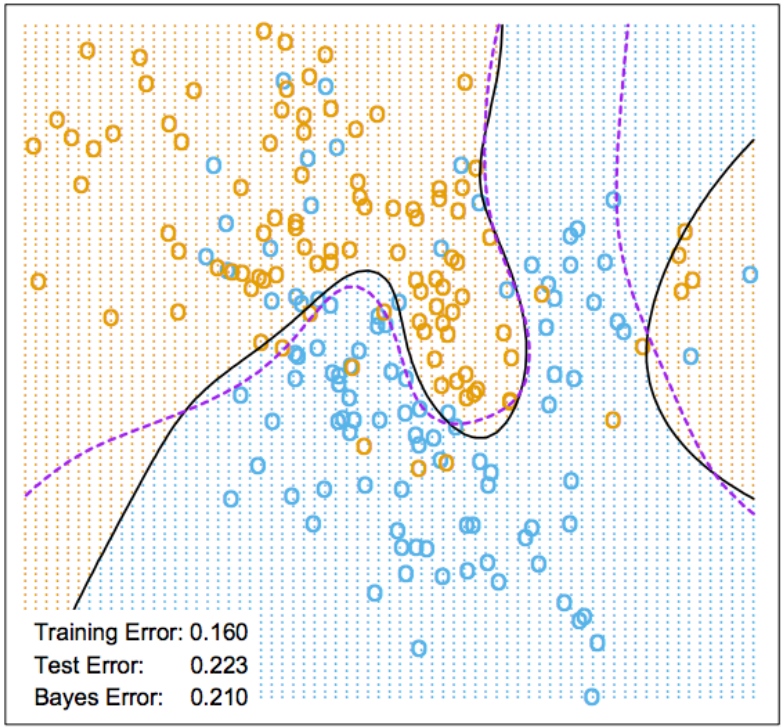
- Em que lambda é um parâmetro que controla a penalização, estimado geralmente por validação cruzada.

Rede Neural – Otimizações

Neural Network - 10 Units, No Weight Decay



Neural Network - 10 Units, Weight Decay=0.02



Rede Neural – Otimizações

- **Número de neurônios e camadas escondidas:**
- Geralmente é melhor mais neurônios do que poucos. Com poucos, o modelo pode não ter flexibilidade suficiente para capturar todas as não linearidades nos dados. Se for muitos neurônios, o weight decay irá restringir para 0 os que não são úteis. Geralmente utiliza-se entre 5 e 100 neurônios.
- Já utilizar mais camadas escondidas permite a construção de variáveis hierárquicas em diferentes levels de resolução.

- **Múltiplos mínimos:**
- A função objetivo é não convexa e geralmente apresenta vários mínimos locais. Sugere-se testar diferentes valores iniciais para os pesos e utilizar o melhor modelo. Outra alternativa é utilizar a média de todas as redes construídas.

Rede Neural – Otimizações

- Note que se f e g são a função identidade, então o modelo inteiro é composto por funções lineares. Então uma rede neural é pode ser vista como uma generalização não linear de uma função linear.

- Nas primeiras versões de redes neurais os neurônios da camada escondida disparavam apenas quando o sinal total passado passava de um limitante, o que seria o mesmo ao usarmos uma ”step function”.

- Para classificação e utilizando entropia cruzada a função softmax, a rede neural é exatamente uma regressão logística nas camadas escondidas.

Rede Neural – Exemplo 1

$$\text{Sum of sigmoids: } Y = \sigma(a_1^T X) + \sigma(a_2^T X) + \varepsilon_1;$$

$$\text{Radial: } Y = \prod_{m=1}^{10} \phi(X_m) + \varepsilon_2.$$

$X_1, X_2, \ldots, X_{10}$ são variáveis aleatórias normalmente distribuídas
ε_1 e ε_2 são erros aleatórios normalmente distribuídos.

100 observaçõe foram simuladas para cada caso treinar o modelo e 10.000 observações de teste.

Erro quadrático foi utilizado como medida de performance.

Rede Neural – Exemplo 1

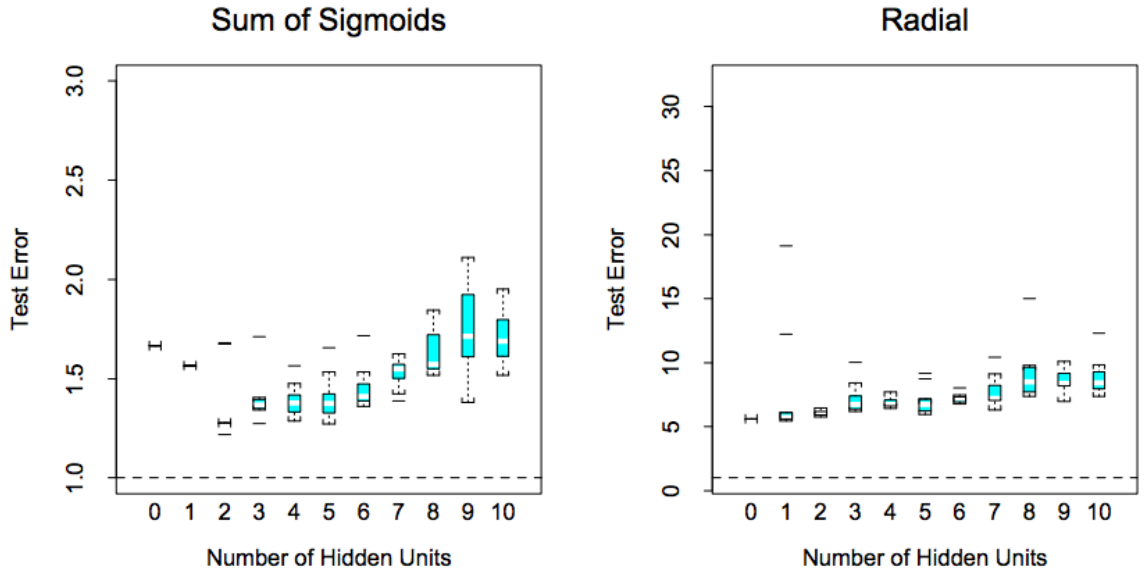


FIGURE 11.6. *Boxplots of test error, for simulated data example, relative to the Bayes error (broken horizontal line). True function is a sum of two sigmoids on the left, and a radial function is on the right. The test error is displayed for* 10 *different starting weights, for a single hidden layer neural network with the number of units as indicated.*

Rede Neural – Exemplo 1

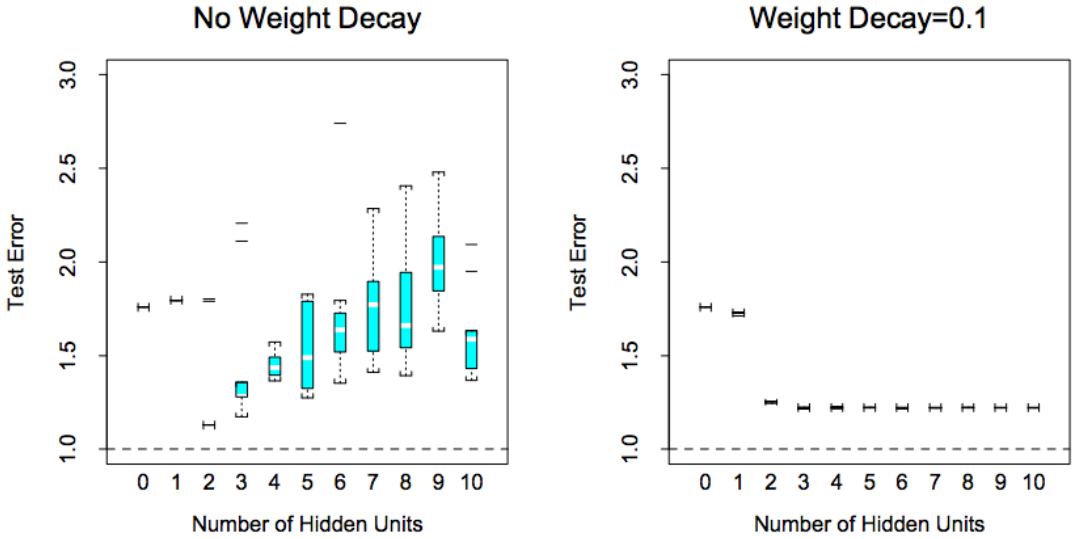


FIGURE 11.7. *Boxplots of test error, for simulated data example, relative to the Bayes error. True function is a sum of two sigmoids. The test error is displayed for ten different starting weights, for a single hidden layer neural network with the number units as indicated. The two panels represent no weight decay (left) and strong weight decay* $\lambda = 0.1$ *(right).*

Rede Neural – Exemplo 1

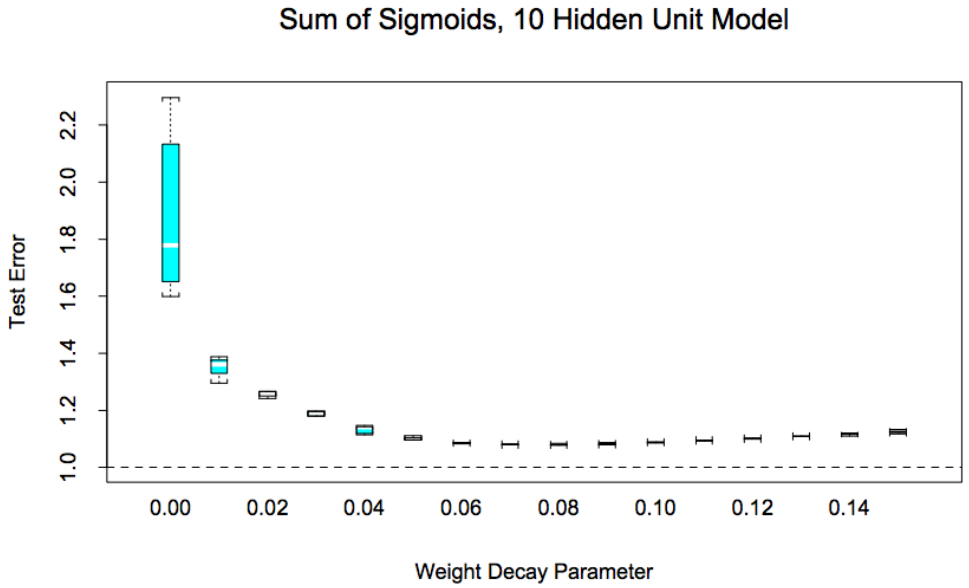


FIGURE 11.8. *Boxplots of test error, for simulated data example. True function is a sum of two sigmoids. The test error is displayed for ten different starting weights, for a single hidden layer neural network with ten hidden units and weight decay parameter value as indicated.*

Rede Neural – Exemplo 2

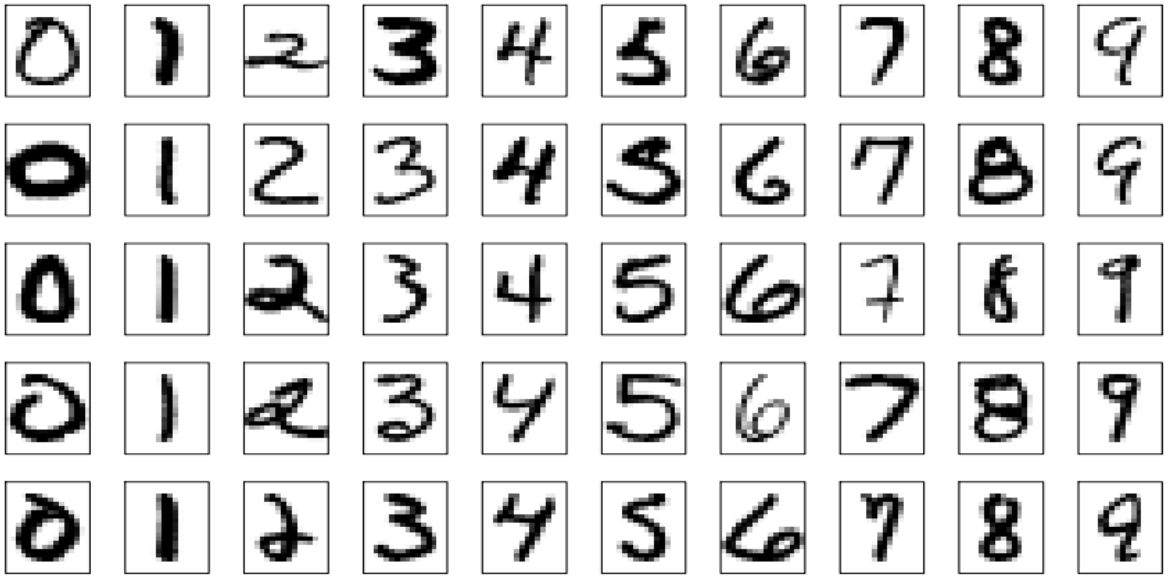


FIGURE 11.9. *Examples of training cases from ZIP code data. Each image is a 16 × 16 8-bit grayscale representation of a handwritten digit.*

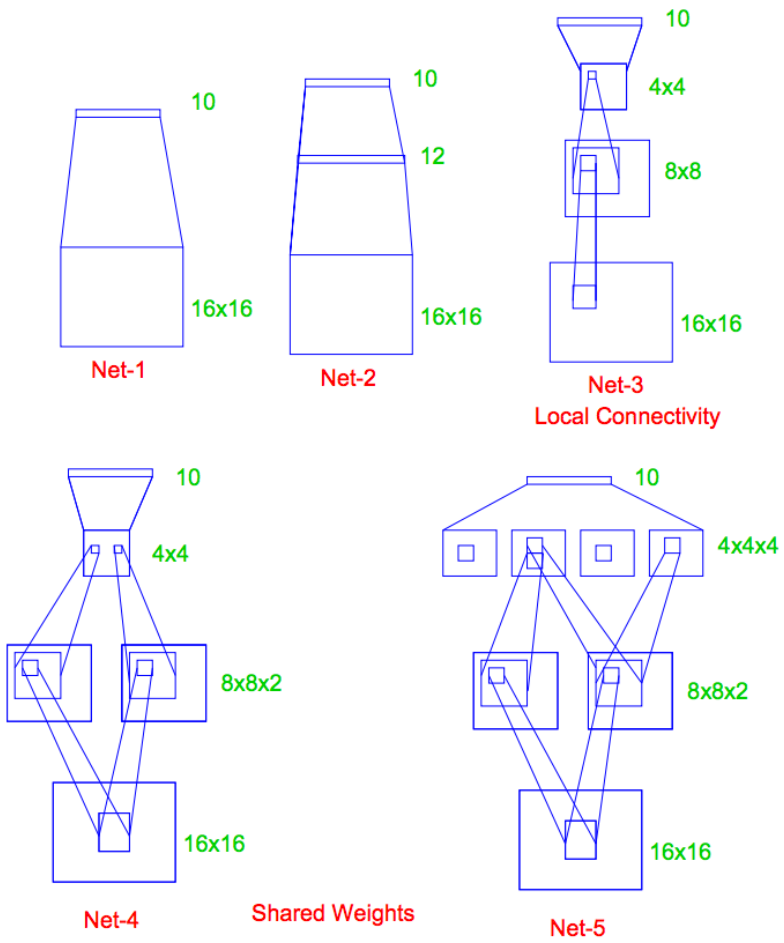
Dígitos de CEP americanos escaneados de envelopes do U.S. Postal Office.

320 dígitos na base de treino e 160 na base de teste.

Rede Neural – Exemplo 2

- **5 tipos diferentes de redes:**
- Rede 1: Sem camada escondida, equivalente a regressão logística multinomial.
- Rede 2: Uma camada escondida, 12 neurônios totalmente conectados.
- Rede 3: Duas camadas escondidas localmente conectadas.
- Rede 4: Duas camadas escondidas localmente conectadas com parâmetros compartilhados
- Rede 5: Duas camadas escondidas localmente conectadas, dois níveis de parâmetros compartilhados

Rede Neural – Exemplo 2



Rede Neural – Exemplo 2

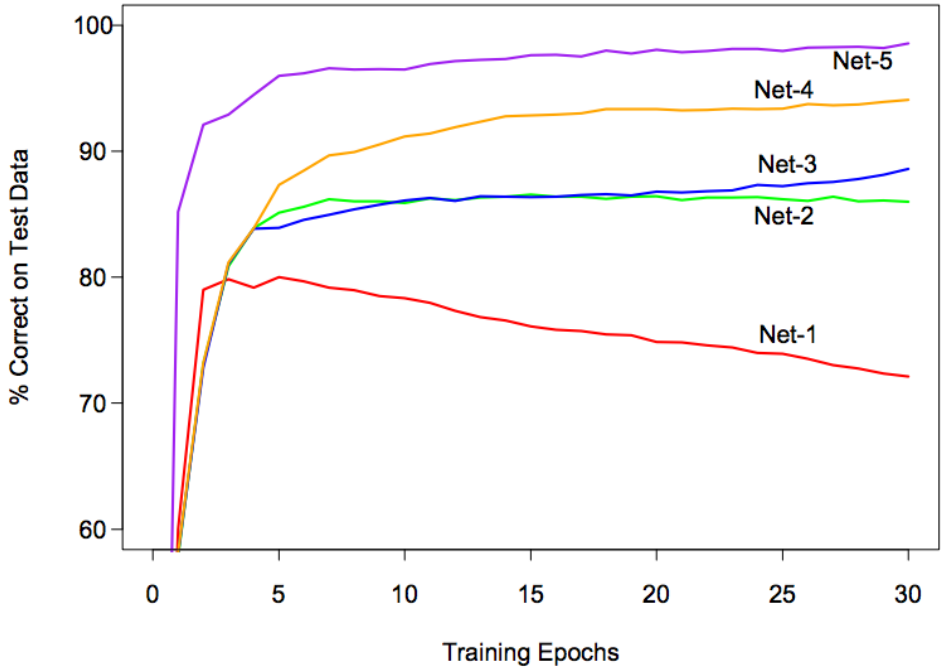


FIGURE 11.11. *Test performance curves, as a function of the number of training epochs, for the five networks of Table 11.1 applied to the ZIP code data. (Le Cun, 1989)*

Rede Neural – Exemplo 2

TABLE 11.1. *Test set performance of five different neural networks on a handwritten digit classification example (Le Cun, 1989).*

	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

Rede Neural - Laboratório – Regressão

Base simulada com 150 observações e 5 variáveis.

- Gastos no cartão em reais
- Idade
- Renda
- Pagamento de impostos
- Segmento

Objetivo:
Prever os Gastos no cartão com base nas outras informações.

```
> head(dados)
  Gastos_Cartao Idade Renda Impostos Segmento
1           510    35  1120       60        C
2           490    30  1120       60        C
3           470    32  1040       60        C
4           460    31  1200       60        C
5           500    36  1120       60        C
6           540    39  1360      120        C
```

Rede Neural - Laboratório – Regressão

Notebook de análise: Gastos_cartao.ipynb

Rede Neural - Laboratório – Classificação

Laboratório R - Base de Spam

- Base com 4.601 e-mails. Porcentual em que 54 palavras ou pontuações aparecem em cada e-mail. Além disso, temos o tamanho médio das palavras, tamanho da maior palavra e quantidade de palavras.

Objetivo:
Criar um detector automático de SPAM que verificará cada novo e-mail.

- Disponível em: https://archive.ics.uci.edu/ml/datasets/Spambase

Rede Neural - Laboratório – Classificação

Notebook de análise: Spam.ipynb

Rede Neural - Laboratório – Exercício

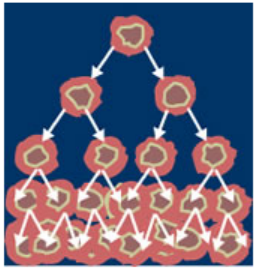
http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	390512

Rede Neural - Laboratório – Exercício

Base de dados (“cancer.data”) com 699 observações e 10 variáveis de pacientes com tumores. O objetivo é detectar com base em algumas informações dos tumores se é benigno ou maligno.

Variáveis:

- 1. Sample code number id number
- 2. Clump Thickness 1 - 10
- 3. Uniformity of Cell Size 1 - 10
- 4. Uniformity of Cell Shape 1 – 10
- 5. Marginal Adhesion 1 - 10
- 6. Single Epithelial Cell Size 1 - 10
- 7. Bare Nuclei 1 - 10
- 8. Bland Chromatin 1 - 10
- 9. Normal Nucleoli 1 - 10
- 10. Mitoses 1 - 10
- 11. Class: (2 for benign, 4 for malignant)

Rede Neural - Laboratório – Exercício

1.) Crie bases de desenvolvimento (80%) e teste (20%). Utilize seed de 42.

2.) Cheque e corriga possíveis problemas de missing.

3.) Ajuste uma rede neural simples com 5 neurônios na camada escondida.

4.) Otimize os hiper-parâmetros.

5.) Avalie os resultados.

SVM (Support Vector Machines)

SVM

- Vamos considerar que temos N observações (x_1, y_1), (x_2, y_2), …, (x_N, y_N). Vamos considerar que $y_i \in \{-1, 1\}$.

- Assumindo que as classes são totalmente separáveis, podemos construir um hiperplano que classifica corretamente todas as observações.

Hiperplano: $\{x : \ f(x) = x^T\beta + \beta_0 = 0\}, \ \|\beta\| = 1.$

Classificador: $G(x) = \text{sign}[x^T\beta + \beta_0].$

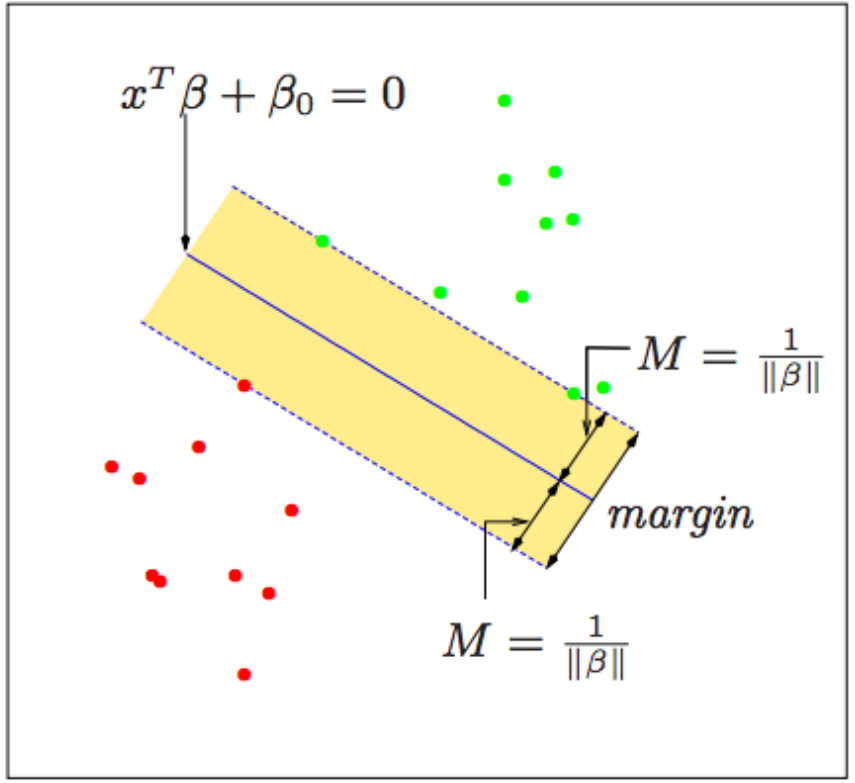
SVM

- A margem é a distância mínima entre os pontos positivos e os pontos negativos do hiperplano.

- O objetivo do método é classificar todos os pontos corretamente e maximizar a margem. Isto equivale a:

$$\max_{\beta,\beta_0,\|\beta\|=1} M$$

$$\text{subject to } y_i(x_i^T\beta+\beta_0) \geq M,\ i=1,\ldots,N,$$

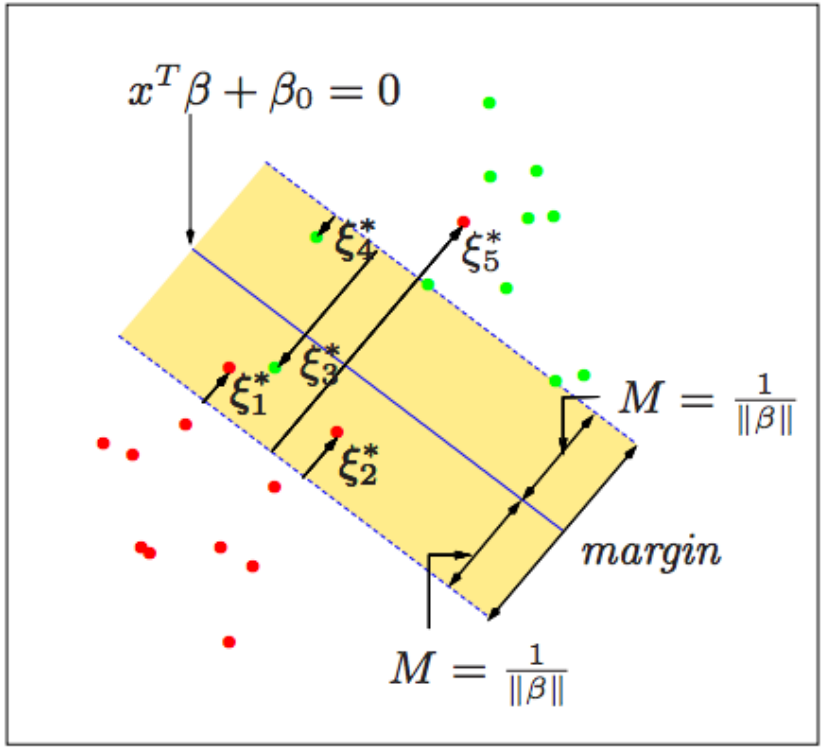


SVM

- E no caso que as classes não são perfeitamente separáveis?
- Neste caso, podemos permitir na otimização que algumas observações sejam mal classificadas.
- Para tal, definimos as variáveis $\xi_1, \ldots, \xi_N$ e mudamos a restrição da otimização para:

$$y_i(x_i^T\beta + \beta_0) \quad \geq \quad M(1 - \xi_i),$$

- ξ_i representa a proporção que a predição é feita erroneamente.



SVM

- Para controlar a proporção de observações mal classificadas, colocamos uma restrição em $\sum \xi_i$. Por exemplo, impondo a restrição que $\sum \xi_i < K$, impomos que o número total de amostras mal classificadas é no máximo K.

- Portanto a função objetivo fica como:

$$\min \|\beta\| \quad \text{subject to} \begin{cases} y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \ \forall i, \\ \xi_i \geq 0, \ \sum \xi_i \leq \text{constant}. \end{cases}$$

- Repare que pontos afastados do hiperplano, acabam não tendo um peso grande em encontrá-la.

SVM

- Computacionalmente é conveniente escrever a otimização como:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\ \forall i,$$

- O parâmetro C é um parâmetro de custo que otimizamos para obtermos a melhor performance. Este parâmetro controla o tradeoff entre complexidade e overfitting.

SVM

- Até o momento, só vimos o SVM formando uma região linear de fronteira entre as classes.

- A ideia para criar regiões não lineares é transformar o espaço das variáveis explicativas por meio de kernels/splines, ou seja, alterar x_i para $h(x_i) = \left(h_1(x_i), h_2(x_i), \dots, h_M(x_i)\right)$

- Neste novo espaço, a fronteira é criado novamente supondo linearidade. Ao retornarmos ao espaço original, fronteiras totalmente não lineares foram criadas.

SVM

- Aumentando a dimensão e complexidade do espaço criado, é muito fácil ocorrer overfitting.

- Otimização:

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

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0. \end{aligned}$$

Kernel

$$K(x, x') = \langle h(x), h(x') \rangle$$

SVM

- 3 Kernels mais populares:

$$
\begin{aligned}
d\text{th-Degree polynomial: } & K(x, x') = (1 + \langle x, x' \rangle)^d, \\
\text{Radial basis: } & K(x, x') = \exp(-\gamma \|x - x'\|^2), \\
\text{Neural network: } & K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2).
\end{aligned}
$$

- Exemplo: X_1 e X_2 e um polinomial kernel de grau 2.

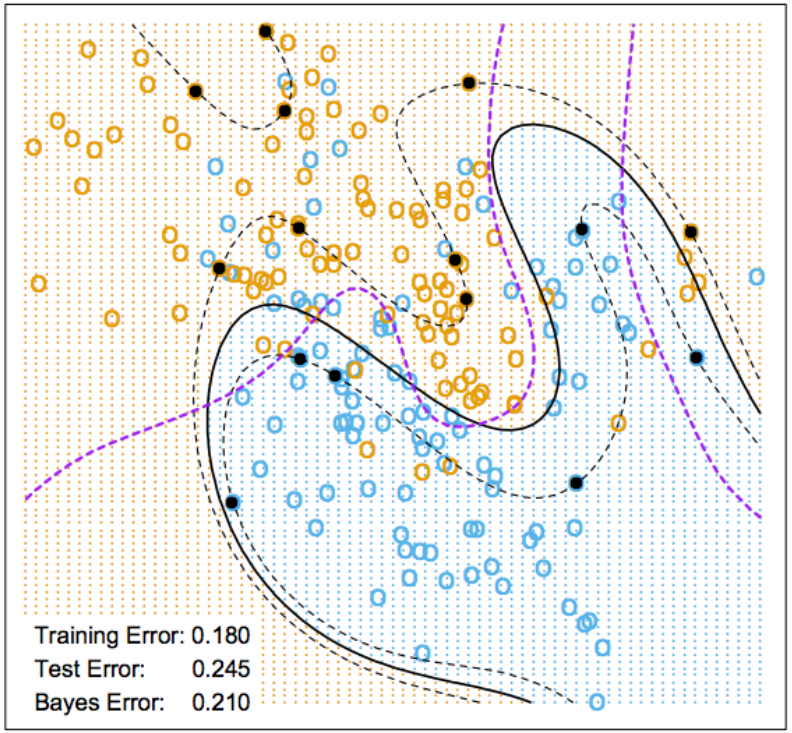
$$
\begin{aligned}
K(X, X') &= (1 + \langle X, X' \rangle)^2 \\
&= (1 + X_1 X_1' + X_2 X_2')^2 \\
&= 1 + 2X_1 X_1' + 2X_2 X_2' + (X_1 X_1')^2 + (X_2 X_2')^2 + 2X_1 X_1' X_2 X_2'.
\end{aligned}
$$

SVM

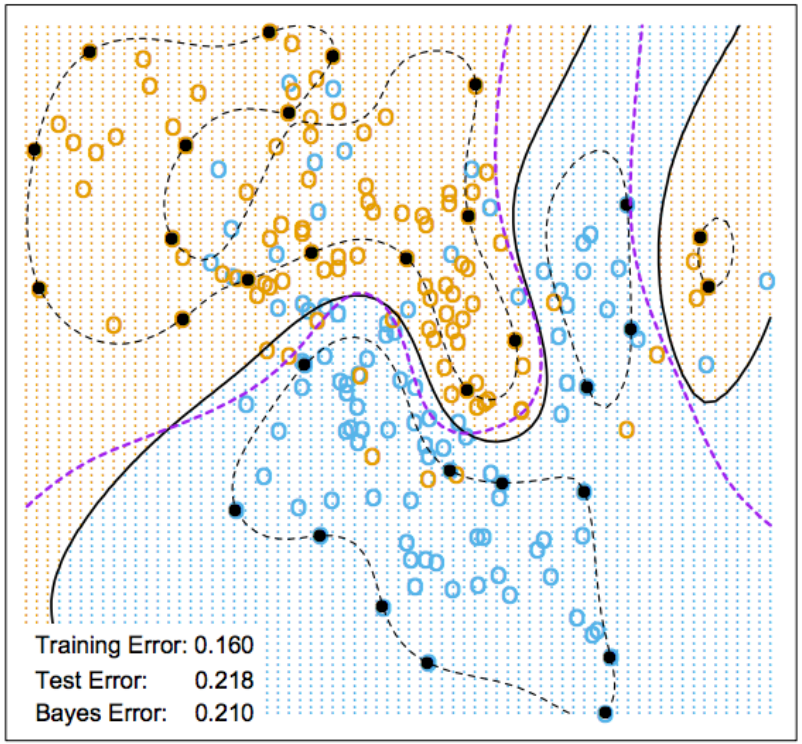
- Repare que facilmente podemos aumentar a dimensão do espaço consideravelmente. Logo, o parâmetro C é muito importante neste contexto.
- Um valor muito alto de C não irá permitir nenhum ξ_i positivo e overfitting irá ocorrer.
- Um valor muito baixo de C pode levar uma regra muito simples e má performance do modelo.
- Podemos escolher C por cross validation.

SVM

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



SVM

- SVM também pode ser adaptador para problemas de regressão de forma similar.

- SVM é bastante utilizado para problemas de classificação com mais de 2 classes, em que um classificador para cada classe pode ser construído.

SVM - Laboratório – Regressão

Base simulada com 150 observações e 5 variáveis.

- Gastos no cartão em reais
- Idade
- Renda
- Pagamento de impostos
- Segmento

Objetivo:
Prever os Gastos no cartão com base nas outras informações.

```
> head(dados)
  Gastos_Cartao Idade Renda Impostos Segmento
1           510    35  1120       60        C
2           490    30  1120       60        C
3           470    32  1040       60        C
4           460    31  1200       60        C
5           500    36  1120       60        C
6           540    39  1360      120        C
```

SVM - Laboratório – Regressão

Notebook de análise: Gastos_cartao.ipynb

SVM - Laboratório – Classificação

Laboratório R - Base de Spam

- Base com 4.601 e-mails. Porcentual em que 54 palavras ou pontuações aparecem em cada e-mail. Além disso, temos o tamanho médio das palavras, tamanho da maior palavra e quantidade de palavras.

Objetivo:
Criar um detector automático de SPAM que verificará cada novo e-mail.

- Disponível em: https://archive.ics.uci.edu/ml/datasets/Spambase

SVM - Laboratório – Classificação

Notebook de análise: Spam.ipynb

SVM - Laboratório – Exercício

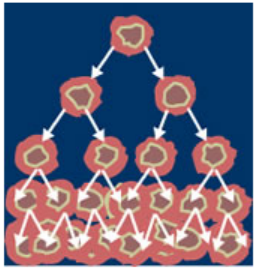
http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	390512

SVM - Laboratório – Exercício

Base de dados (“cancer.data”) com 699 observações e 10 variáveis de pacientes com tumores. O objetivo é detectar com base em algumas informações dos tumores se é benigno ou maligno.

Variáveis:

- 1. Sample code number id number
- 2. Clump Thickness 1 - 10
- 3. Uniformity of Cell Size 1 - 10
- 4. Uniformity of Cell Shape 1 – 10
- 5. Marginal Adhesion 1 - 10
- 6. Single Epithelial Cell Size 1 - 10
- 7. Bare Nuclei 1 - 10
- 8. Bland Chromatin 1 - 10
- 9. Normal Nucleoli 1 - 10
- 10. Mitoses 1 - 10
- 11. Class: (2 for benign, 4 for malignant)

SVM - Laboratório – Exercício

1.) Crie bases de desenvolvimento (80%) e teste (20%). Utilize seed de 42.

2.) Cheque e corriga possíveis problemas de missing.

3.) Ajuste uma SVM com C igual a 1.

4.) Otimize os hiper-parâmetros.

5.) Avalie os resultados.

Referências Bibliográficas

- Hastie, T., Tibshirani, R. & Friedman, J.H. (2001) “The Elements of Statistical Learning”
- Bishop, C.M. (2007) “Pattern Recognition and Machine Learning”
- Mitchell, T.M. (1997) “Machine Learning”
- Abu-Mostafa, Y., Magdon-Ismail, M., Lin, H.T (2012) “Learning from data”
- Theodoridis, S., Koutroumbas, K., (2008) “Pattern Recognition”
- Kuhn, M., Johnson, K., (2013) “Applied Predictive Modeling”