APRENDIZADO DE MAQUINA

Definição de aprendizado: mudança adaptativa no comportamento provocado por dinâmica da experiência

Todos os animais com um sistema nervoso e capaz de aprender.

* Nem todos os problemas podem ser resolvidos por um programa, nesse caso é necessario Generalizar

* Em ML Experiência = Dado

+ dado, melhora a experiencia e resultado senão infringe regra da TA

Aprendizado de maquina produz um programa a partir do dado do mundo do problema.

APRENDIZADO SUPERVISIONADO: com exemplos de um problema, associa-se o valor da entrada (X) a saida (Y)

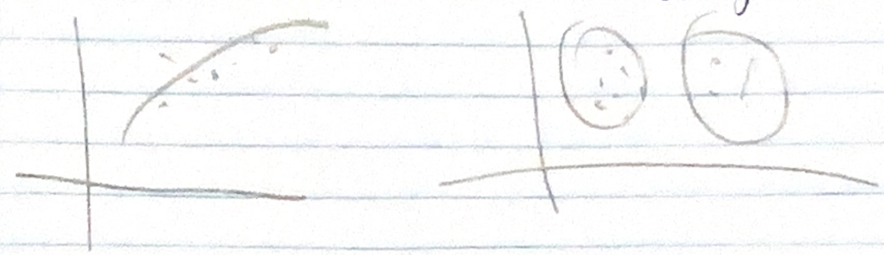
APRENDIZADO POR REFORÇO: regras que definem um ambiente, busca-se simular comportamentos que respeitem essas regras e recebem recompensas com resultados positivos. Busca padrões com recompensas boas p/ gerar [illegible]

APRENDIZADO NÃO SUPERVISIONADO: Busca padrões, normalmente agrupando os dados: discover clusters, manifolds, embedding.

CHAMADO DE: Estimação estatística
Reconhecimento de padrão
Rede Neural
Data Mining

APRENDIZADO SUPERVISIONADO: realiza uma regressão para traçar uma curva que representa o dado

Classifica o dado em categorias



APRENDIZADO NÃO SUPERVISIONADO separa por agrupamentos, mas ñ se sabe porque foi feito (agrupado) desta forma

QUANDO NECESSARIO AULAS 8 e 9 GRAVADAS ENSINADO CALCULAR MLP COM MATRIZES e BACK PROPAGATION.

Funções / Formulas - Aprendizagem de Maquina

Minimizar o erro/perda

$$\min_{W} \sum_{i=1}^{m} \left(\overbrace{f_W(x^{(i)})}^{\text{predito}} - \overbrace{(y^{(i)})}^{\text{Real}} \right)^2$$

→ ignora direção ⊕ ⊖

→ mais simples

muito usada

(0,1) saída

X → entrada

W → Pesos ou parametros

Y → entrada

m → nº de amostras

i → indexador entrada

f_W → função composta por pesos ou parametros W

Treinamento ou sintonia → Ajuste do W

→ Função alvo desconhecida $f: X \to Y$ (oráculo)

→ Hipotese (função de aproximação) $g: X \to Y$ (desejada)

→ Dados: $(X_1, y_1) .. (X_N, y_N)$ → Traing set dado de treino ou amostra

ORÁCULO

UNKNOWN TARGET FUNCTION
$T: X \to Y$

AMOSTRA

TRAINING EXAMPLES
$(X_1, y_1) .. (X_n, y_n)$

* Parametros são descobertos pelo algoritmo

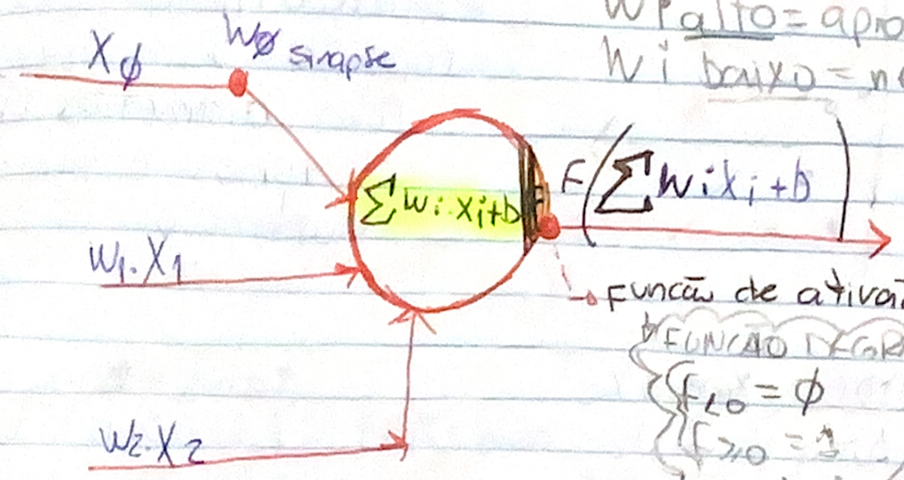
Hiperparametros

Ex: altura da arvore, normalização, taxa de aprendizado

LEARNING ALGORITHM A → FINAL HYPOTHESIS $g \cong f$

HYPOTHESIS SET H

Perceptron



W_i alto = aprovação

W_i baixo = negação

Aprova: $\sum_{i=1}^{d} w_i \cdot x_i > \text{threshold}$

Denn: $\sum_{i=1}^{d} w_i \cdot x_i < \text{threshold}$

Junção das duas formulas

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i \cdot x_i\right) - \text{threshold}\right)$$

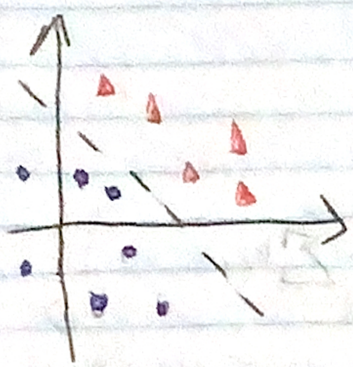
threshold → w_ϕ

retorna $\geq \phi$ +1 aprova

retorna $< \phi$ −1 nega

Formulação vetorial $h(x) = \text{sign}(w^T x)$

$$h(x) = \text{sign}\left(\sum_{i=\phi}^{d} w_i \cdot x_i\right)$$



=> Perceptron realiza separação linear do hiperplano, iniciando com parametros (w_i) aleatorios e vai ajustando

NAND, X_1 and X_2

Inputs V_0, X_1, X_2 com $X_0 = 1$

Threshold $T = 0,5$

Bias $= b = \emptyset$

Learning rate $= r = \emptyset.1$

Training set

$((1,0,0),1)$
$((1,0,1),1)$
$((1,1,0),1)$
$((1,1,1),0)$

X_n y

$X = (\underset{x_0}{1}, \underset{x_1}{0.5}, \underset{x_2}{-0.5})$ => $y = -1$

$W = (\underset{w_0}{0.5}, \underset{w_1}{2}, \underset{w_2}{-1})$ => (+1) diminuir pesos das entradas

$x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2$

$\underbrace{0.5 + 1 + 0.5}_{2 \quad 2 > 0.5}$ (linear athoc)

$\cdot (w,b) \leftarrow (w,b) - (x,1) =$
$(0.5, 2, -1) - (1, 0.5, -0.5) = (\underset{x_0}{-0.5}, \underset{x_1}{1.5}, \underset{x_2}{-0.5})$

Se haver separação linear (para os dados), o algoritmo do perceptron irá encontrala.

Você para de iterar dentro de um limite de epocas que você informar HIPERPARAMETRO ou até parar de errar PIOR CENARIO ou taxa de erro aceitavel PODE TB FICAR LOOP

FUNÇÕES DE ATIVAÇÃO

Degrau

$$\phi(v) = \begin{cases} 1, & \text{if } v \geq \emptyset \\ \emptyset, & \text{otherwise.} \end{cases}$$

Piecewise Linear

$$\phi(v) = \begin{cases} 1, & \text{if } v \geq 0.5 \\ v, & \text{if } 0.5 > v > -0.5 \\ \emptyset, & \text{if } v \leq -0.5 \end{cases}$$

Sigmoid

$$\phi(v) = \frac{1}{1 + \exp(-av)}$$

SUAVIZADA

$\frac{1}{1+e^x}$

CALCULO DE NOVOS PESOS

REGRA DELTA

$$\underset{\text{peso atual}}{w} \leftarrow \underset{\text{peso atual}}{w} + \underset{\text{taxa aprendizado}}{r}[\underset{\text{real}}{d} - \underset{\text{predição}}{y}]\underset{\text{entrada}}{X}$$

recebe

r x [erro] x entrada

PAC => PROBABILY APROXIMATE CORRECTLY

POSSIBLE x PROBABLE

PAC learning

Evento ruim

$$P[|v - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

(Hoeffding's Inequality)

Margem erro

auxilia no entendimento de overfitting aprendizagem supervisionada

$\mu = v$ is P.A.C.

↑ N grande e ↓ ϵ pequeno vice versa

NOTATION

μ e ν depende da escolha de h

$N_i \to \nu$ $E_{in} \Rightarrow$ erro na amostra ou erro empírico (que pode ser)

$E_{in}(h) \Rightarrow$ Baixo, boa hipótese (se amostra grande)

$M_i \to \mu$ $E_{out} \Rightarrow$ erro fora da amostra (todo o restante) erro esperado

$E_{out}(h) \Rightarrow$ não mesmo, se for baixo a hipótese é boa porque tem muito dado

testa-se varias hipóteses

$$P[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

$\to$ H complexo errar menos e mais

- overfitting

nº de hipóteses

Conjunto de mais (caixas) testados (HIPÓTESES)

$E_{in}(h)$ e $E_{out}(h)$ $\to$ começa do mais simples vai até o mais complexo

(H)

Dilema - H simples ou H complexos

Complexidade tem relação com priorizar (amostra) (mundo) E_{in} ou E_{out} ou equilibrar os dois

THE LINEAR MODEL

* FEATURE ENGINEAR

x_1: intensidade
x_2: simetria

} Ajuda a criar características para ter HS mais ou menos complexos.

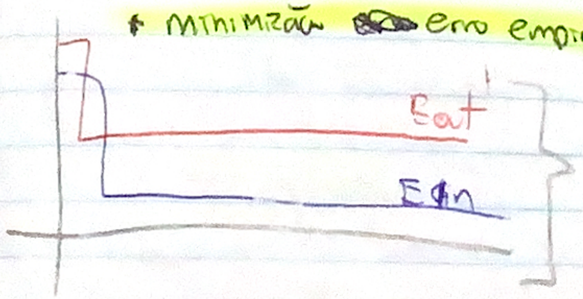
* Viés na amostra $\to$ aprendizado ativo, colocar um viés p/ escolher a amostra, se o viés for adequado, pode-se aumentar o nº de hipóteses (ou seja ser mais complexo em relação as hipóteses)

Curva, erro empírico E_{in} x erro esperado E_{out}

* Milésima época

Erro empírico é uma aproximação do erro esperado. Dado de treino, parte usado p/ E_{in} e o restante para E_{out}

* Minimização erro empírico melhor + com limitações



Minimização erro empírico, perceptron de bolso

Busca pela melhor hipótese

ALGORITMOS

apenas problemas bem definidos linearmente

Detectado baixa capacidade do perceptron, buscou-se outros algoritmos a evolução do perceptron:

(1) ÁRVORE DE DECISÃO

(2) REDES NEURONAIS
(POUCAS CAMADAS)
perceptrons encadeados nas saídas

(3) PERCEPTON ÓTIMO (SVM)
modelo estatístico entre (1) e (2)
(mais ainda problemas linearmente)

KERNEL
TRANSFORMA O DADO PI QUE SEJA LINEARMENTE SEPARAVEL
+
SVM

(4) ENSEMBLE
Combinações de vários MODELOS
ADABOOST - BUSTING
RANDOM FOREST - BAGIN

(5) REDES NEURAIS DE (MULTIPLAS CAMADAS) +

(6) DEEP LEARNING
TRABALHA BEM COM MUITOS DADOS
→ aprendizado profundo
→ under especification (paper)

ÁRVORES DE DECISÃO

A melhor separação é um problema NP completo, usa-se uma Heuristica
[os positivos / os negativos]

$$H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log\frac{p}{p+n} - \frac{n}{p+n}\log\frac{n}{p+n}$$

→ Entropia: impureza ou incerteza associada aos conjuntos de dados ou seja, medida pl entender quão confuso e misturado está o conjunto de dados: alto (misturado), baixo (homogeneo). Mede o grau previsível e o dado

→ Entropia antes da distribuição dos dados

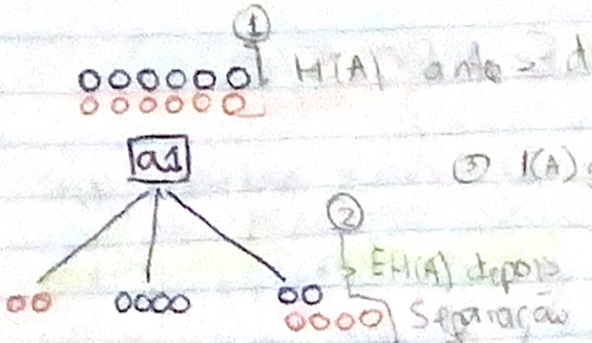
$$EH(A) = \sum_{i=1}^{k} \frac{p_i+n_i}{p+n} H\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

→ Entropia após separação dos dados
→ Entropia esperada

$$I(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(A)$$

→ Ganho de informação
→ Processo recursivo

Heuristica: calcula I(A) pl todos atributos e escolhe o que oferece maior ganho de Informação



$\log_a \left(\frac{b}{c}\right) = \log_a b - \log_a c$

6 $\boxed{O = P}$ positivos

6 $\boxed{O = N}$ negativos

1°) $H(A) = \left(\frac{P}{P+N}, \frac{N}{P+N}\right) = \left[\frac{-P}{P+N} \log \frac{P}{P+N}\right] + \left[\frac{-N}{P+N} \log \frac{N}{P+N}\right]$

$P = 6$

$N = 6$ $H(A) = \left(\frac{6}{12}, \frac{6}{12}\right) = \left[-\frac{6}{12} \log \frac{6}{12}\right] + \left[-\frac{6}{12} \log \frac{6}{12}\right]$

$H(A) = \left(\frac{6}{12}, \frac{6}{12}\right) = \left[-\frac{1}{2} \cdot (-1)\right] + \left[-\frac{1}{2} \cdot (-1)\right] = \boxed{1}$

Entropia da distribuição máxima e $\boxed{1}$

2°) $EH(A) = \sum_{i=1}^{k} \frac{P_i + N_i}{P+N} \cdot H\left(\frac{P}{P+N}, \frac{N}{P+N}\right)$ total conjunto

$H(A) = \left[\frac{0}{0+2}, \frac{2}{0+2}\right], \left[\frac{4}{4+0}, \frac{0}{4+0}\right], \left[\frac{2}{2+4}, \frac{4}{2+4}\right]$

$H(A) = (0, 1), (1, 0), \left(\frac{2}{6}, \frac{4}{6}\right)$

$EH(A) = \sum_{i=1}^{3} \left[\frac{0+2}{12}\right] \cdot H(1), \left[\frac{4+0}{12}\right] \cdot H(1), \left[\frac{6}{12}\right] \cdot H(3)$

$EH(A) = 0 + 0 + \left[\frac{6}{12}\right] \cdot H(3)$

$\frac{6}{12} \cdot H(A) \left(\frac{2}{6}, \frac{4}{6}\right)$

$\frac{6}{12} \cdot H(A) \left[-\frac{2}{6} \cdot \log \frac{2}{6}\right] + \left[-\frac{4}{6} \cdot \log \frac{4}{6}\right]$

$\frac{6}{12} \cdot H(A) \left[+0.5283\right] + \left[+0.3819\right]$

$0.5 \times H(A) \left[0.9182\right]$

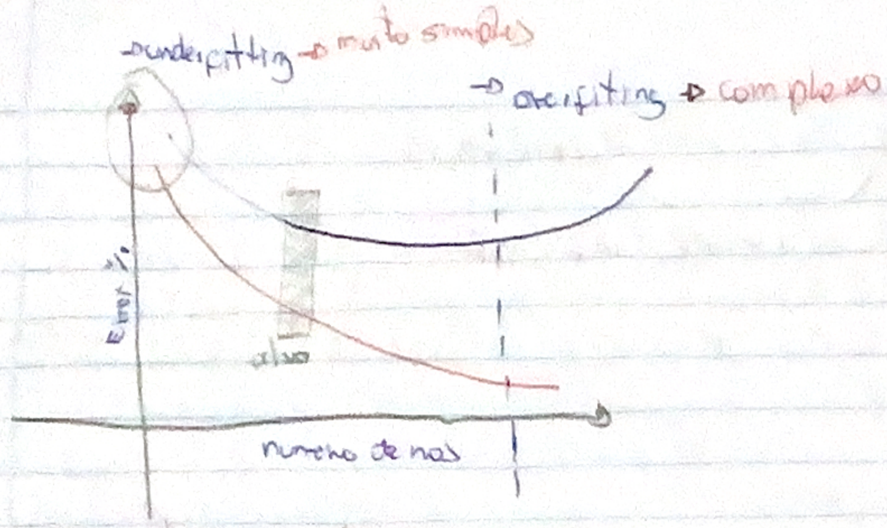
$EH(A) = 0.4591$

3°) $I(A) = H\left(\frac{P}{P+N}, \frac{N}{P+N}\right) - EH(A)$

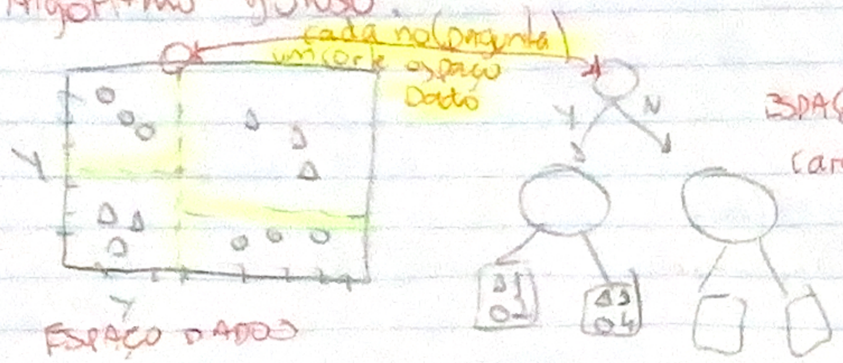
$I(A) = 1 - 0.4591$

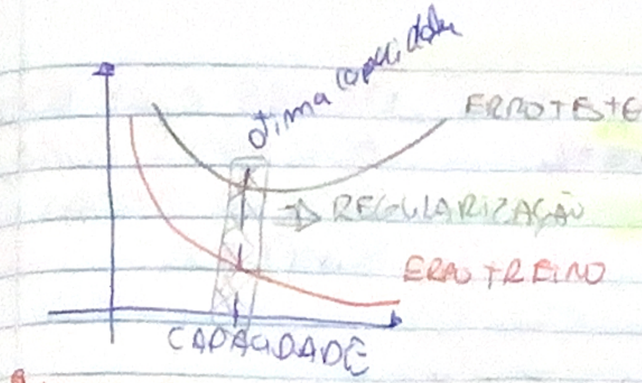
$I(A) = 0,5409$ reduziu a entropia

- A arvore cresce até todos nos serem puros ou ate usar todos os atributos
- Ter somente um no em cada folha produz overfitting → baixa generalização
- Dados discretos ou discretizados
- Para com alvo de pureza aceitavel (Como empirico aceitavel)
- Cresce a arvore toda e de baixo pra cima, corta até alvo aceitavel.



Algoritmo guloso.



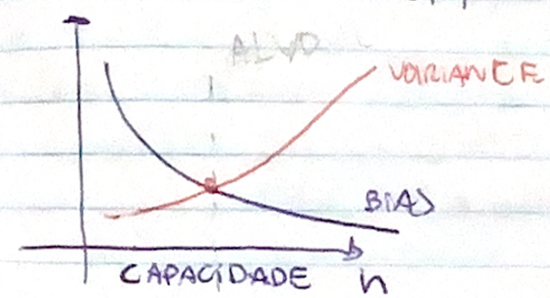


* capacidade é o dado que definem under/over fitting.

* VIES (BIAS) DECORA DADOS DE TREINO — VIES ALTO, MUITOS ERROS DEVIDO AO VIES, MODELO CAPACIDADE ALTA POUCO ERRO DEVIDO AO VIES (mitigado), CAPACIDADE BAIXA, POUCO ERRO DEVIDO AO VIES
Pouco Dado → simplifica muito as coisas

* VARIANCIA — CAPACIDADE BAIXA POUCO ERRO DEVIDO
↓ n conseguimos VARIANCIA, CAPACIDADE ALTA MUITO chegar $f(x) \cong g(x)$ ERRO DEVIDO A VARIANCIA.

VIES E VARIANCIA



PROCURAMOS ESPAÇO DE HIPOTESES PEQUENO QUE CONTENHA f

O APRENDIZADO É COMO UMA BUSCA NO ESPAÇO DE HIPOTESES H

* RISCO EMPIRICO — MODELO TREINADO E APLICADO MODELO NO DADO DE TREINO

* RISCO ESPERADO — MODELO APLICADO AO DADO ESPERADO FORA DA AMOSTRA

* CAPACIDADE
$y = wx + b$
capacidade dos em d dimensões
→ Encontro de erro empirico x erro esperado

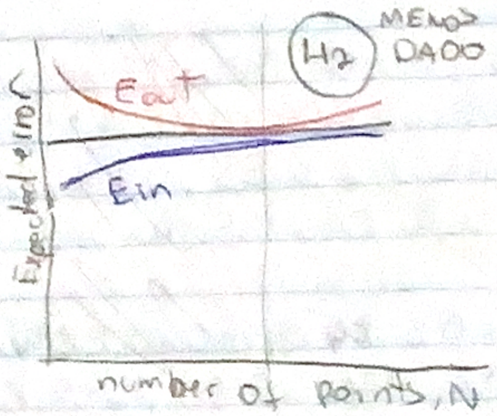
PARA TODAS AS POSSIVEIS ROTULAÇÕES, ENCONTRAR ERRO EMPIRICO ZERO, OU SEJA P/ MUDANÇA DE TODOS OS NÓS DA ARVORE LIMITADO POR N.
+ CAPACIDADE − ERRO EMPIRICO

REGULARIZAÇÃO ou FATOR DE REGULARIZAÇÃO: COMPROMISSO ENTRE ERRO E CAPACIDADE, Ñ QUEREMOS CRESCER H INDISCRIMINADAMENTE. MINIMIZAR O ERRO EMPIRICO SEM PASSAR DO PONTO QUE ELE SEJA BOA APROXIMAÇÃO DO ERRO ESPERADO.

NAVALHA DE OCCAM: DENTRE DUAS ESCOLHAS IGUALMENTE BOAS, ESCOLHA A MAIS SIMPLES.

ERRO EMPÍRICO IGUAL

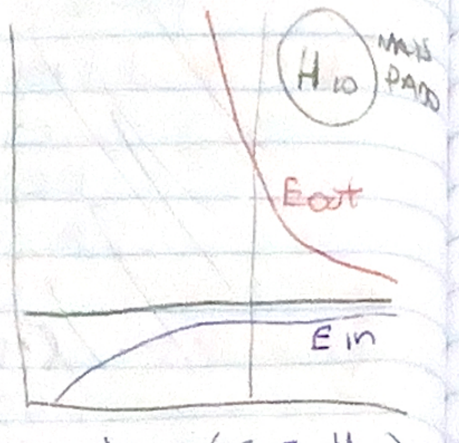
ESCOLHA O COM MENOR CAPACIDADE.



PRECISA DE POUCO DADO P/ APROXIMAR

H2 MAIS SIMPLES H10

+ O ERRO E ALTO



Overfitting (f e H_{10})

PRECISA DE MUITO DADO P/ APROXIMAR

Ein = Eout + o Erro Empírico é NZ NOR

* Necessário ter senso crítico em relação capacidade e volume de dados

+ folhas +CAPACIDADE A ÁRVORE TEM, OU SEJA, ELA É MAIS COMPLEXA, MAIS HIPÓTESES

$\Omega(T)$ = nº de folhas da árvore → Penaliza modelos + complexos do que deveriam ser

$E_{in}(T) + \lambda\Omega(T)$

Minimizar erro empírico sem crescer muito a árvore senão é penalizado por $\lambda\Omega(T)$

λ entre 0 e 1 é a importância do termo de regularização

REGULARIZAÇÃO (PENALIZAÇÃO)

Nesse caso crescemos a árvore até $E_{in} = 0$, e vamos de baixo p/ cima cortando a árvore e recalculando

$E_{in}(T) + \lambda\Omega(T)$ → PENALIZAÇÃO P/

até termos um valor satisfatório em

$E_{in}(T) + \lambda\Omega(T)$ regularização

$$L = \frac{1}{n}\left(\sum_{i=1}^{i=n} L(W, y^i, X_i) + \lambda C(W)\right)$$

↳ minimizar função de erro é o alvo

MINIMIZAR RISCO ESTRUTURAL REGULARIZAÇÃO COM PENALIZAÇÃO DE MODELOS COMPLEXOS (REGULARIZAÇÃO + REDUÇÃO RISCO EMPÍRICO)

SIGMOID

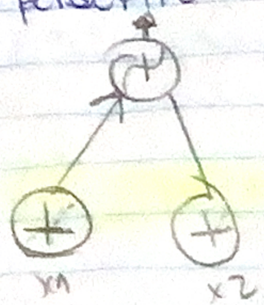
PARA UTILIZAR ~~CALCULAR~~ A PROBABILIDADE DA SIGMOIDE NO MODELO, ELA PRECISA SER CALIBRADA

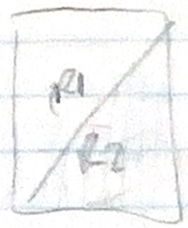
PARA USAR A SIGMOID P/ CALCULAR O ERRO, ELA NÃO PRECISA SER CALIBRADA.

REDES NEURAIS

$P(y=1|x)$ 1 neurônio

Modelo é o perceptron com sigmoid na saída





Regressão logística = rede neural com 1 camada
1 neurônio

$f(x) = \sigma(W^T X + b)$

↳ sigmoid aplicada a saída do perceptron

Perceptron = funciona bem p/ problemas convexos ou linearmente separáveis

RN = funciona p/ não linearmente separáveis.

$L(W) = \frac{1}{2}\sum_{i}^{m} (\sigma(W^T X^{(i)}) - y^{(i)})^2$

↳ minimizar função erro quadrático por meio do gradiente

gradiente — $\nabla_W L$ (minimizar $L(W)$ em relação a W)

$$\nabla_W L = \sum_{i}^{m} [\sigma(W^T X^{(i)}) - y^{(i)}]\, \sigma'(W^T X^{(i)}) X^{(i)}$$

OBTENÇÃO DO GRADIENTE PELA REGRA DA CADEIA

Como treinar uma rede neural

GD - Decidado gradiente atualiza ~~cada época~~ depois passar por todos os registros do treino

① • Inicializa W aleatório

② • Calcula $\nabla_W L$

③ • $W \leftarrow W - r \times (\nabla_W L) = \sum_i \text{Error}(i)$

↳ $\times\, \sigma'(W^T X^{(i)}) \times X^{(i)}$

④ • Repetir 2 e 3

influenciado pela taxa de aprendizagem

Boa tática é plotar a função de perda p/ saber se ela está convergindo ou divergindo. Todas as features devem estar na mesma escala. $X_i' = \frac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)}$

FEATURE SCALING

r → taxa de aprendizado ~~tem~~ que provoca a divergência nesse modelo

SGD - Decida do gradiente estocástico rápido atualiza os pesos a cada iteração → para cada registro do treino

EMBARALHA ORDEM REGISTROS NO TREINO

Recomendado reduzir a taxa de aprendizado a cada iteração.

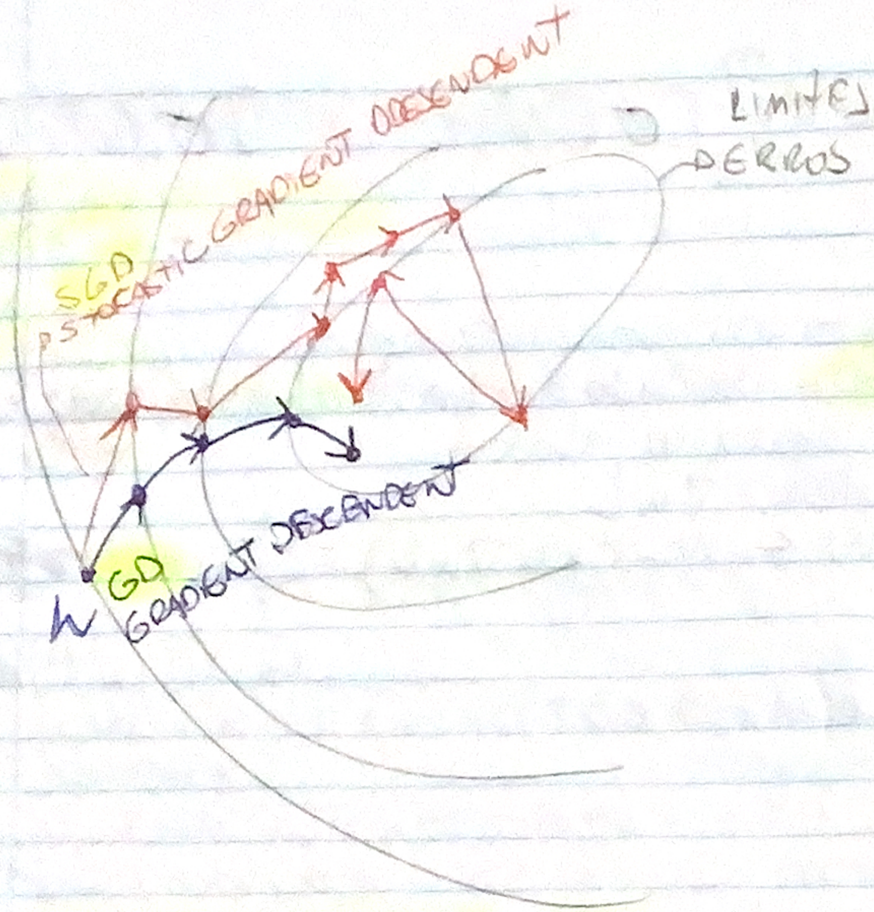
PORQUE SE REDUZ QUANTIDADE DADOS

Angulo da sigmoid de me diz o quanto certo estou

DERIVADA DA SIGMOID: $\sigma(z) = \frac{1}{1+\exp(-z)}$

SLOPE: Inclinação da sigmoide no ponto

DERIVADA DA SIGMOID NO PONTO (Z) $\sigma'(z) = \sigma(z)(1-\sigma(z))$



Mini-batch SGD: junta os erros e atualiza em determinado tempo (batch) tipicamente 10 exemplos p/ recalcular o erro.
(10 Registros)

Em menos epocas converge p/ ~~[illegible]~~ um erro menor que SGD

$$w \leftarrow w - r \times \frac{1}{b}\sum_{i}^{b} Error(i) \times \sigma'(w^T x^{(i)}) \times x^{(i)}$$

Regularização pela normas dos pesos $E_{in}(w) + \lambda \|w\|$

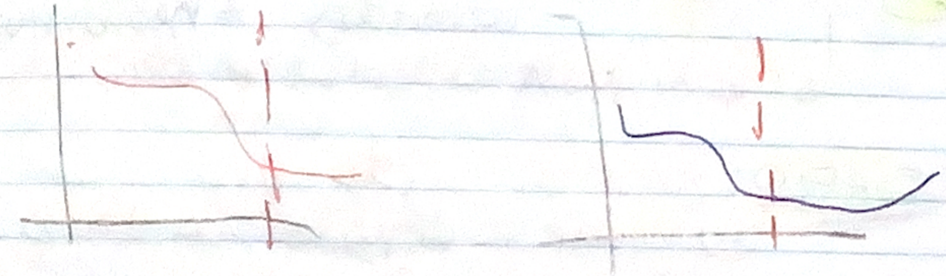
REGULARIZAÇÃO:

Minimizar $L(w) = \frac{1}{2}\sum_{i} (\sigma w^T x^{(i)} - y^{(i)})^2$

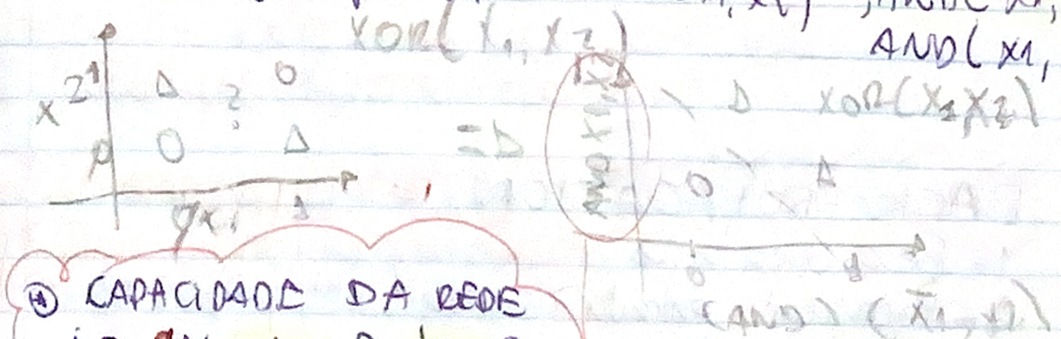
Regularização $L(w) = \frac{1}{2}\sum_{i} (\sigma(w^T x^{(i)}) - y^{(i)})^2 + \|w\|$

DEZESAO PESOS PEQUENOS P/ ~~[illegible]~~ REDUZIR ~~[illegible]~~ ERRO EM MIN + ~~[illegible]~~ POR overfitar.

Parada antecipa (cross-validation): separa amostra em 2 partes: dado de treino e dado de teste, rodamos o modelo treino na parte do teste e realizamos corte



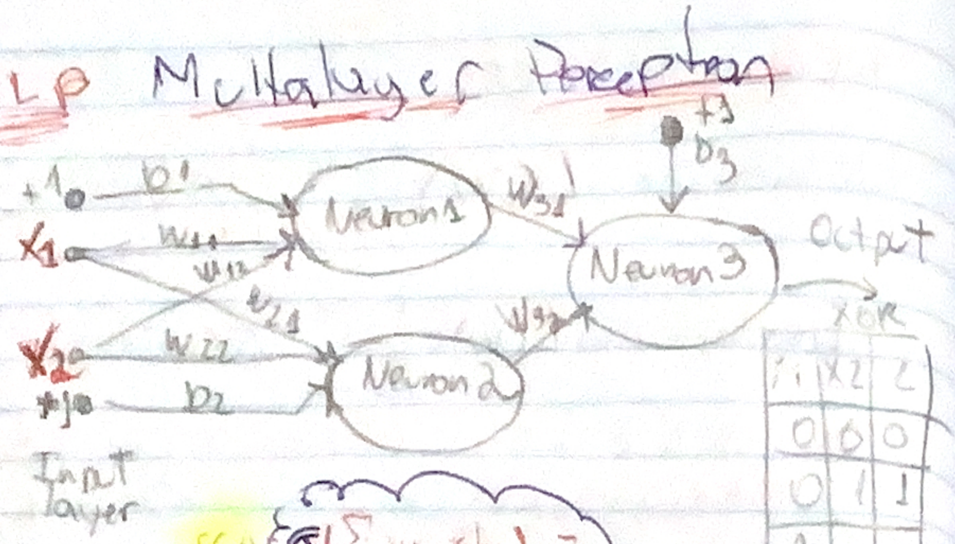
Linearmente separável - OR(x_1, x_2) ; AND$(\bar{x}_1, x_2)$
AND$(x_1, \bar{x}_2)$
XOR(x_1, x_2)



CAPACIDADE DA REDE NEURAL AUMENTA COM MAIS CAMADAS

ADIÇÃO DE CAMADA P/ MUDAR REPRESENTAÇÃO

MLP Multilayer Perceptron



XOR		
x1	x2	z
0	0	0
0	1	1
1	0	1
1	1	0

$f(x) = \sigma(\sum_j w_j \times h_j) =$

$\sigma(\sum_j w_j \times \sigma(\sum_i w_{ij} \times x_i))$

1 camada - Linear
2 camadas - aproximador universal
≥3 camadas - + camadas - neurônios
- camadas + neurônios
↳ rápido porém contribui p/ overfitting

DESAFIO: encontrar pesos p/:
1. Mudar a disposição dos dados tornando o problema ñ linear em linear [representação linear
2. Pesos que permitam encontrar encontrar uma separação

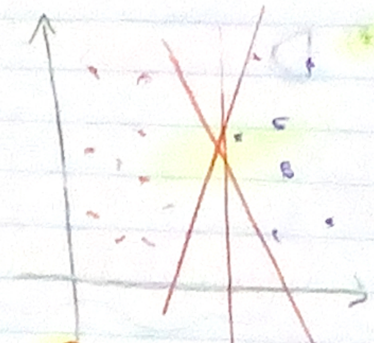
BACK-PROPAGATION retropropagação de erro
1- Pesos aleatórios (podem ser heurísticas p/ ñ ter mínimo local)
2- Para cada entrada + saída $(x^{(i)}, y^{(i)})$
3- calcula o erro (forward pass)
4- calcula gradiente local p/ cada nó — derivada
5- atualiza os pesos (backward pass)

SUPPORT VECTOR MACHINES SVM

Como maximizar a generalização do perceptron p/ uma separação linear?

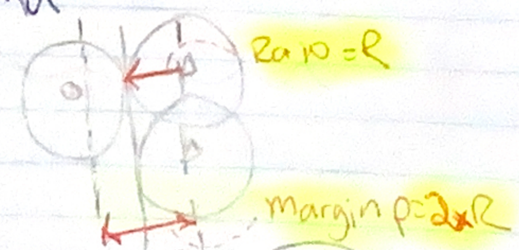
Utilizando a maior margem possível p/ cobrir uma variabilidade dos dados.

Melhor separador
Separador (ou perceptron) de margem máxima



Varias possibilidades

Margem $\rho = 2 \times R$



$$L(h)_{test} \leq L(h)_{train} + O\left(\sqrt{\frac{VC(h)}{n}}\right)$$

Capacidade

Dimensão VC é a forma de calcular a capacidade de um separador SVM

m pontos, 2 cores = 2^m separações

similar a capacidade

no treino

número maximo de pontos que seu modelo h (separador) consegue despedaçar → Dimensão VC

PERCEPTRON NÃO CONTROLA MARGEM

- 3 pontos não lineares são separáveis pelo perceptron em 2D espaço
- dimensão 2D do perceptron é 3
- não consegue 4 pontos em 2D
- Consegue separar 4 pontos em 3D
- D dimensões + 1 = Quantidade de pontos separáveis pelo perceptron ótimo.

$VC(h) = d + 1$

minimizar erro treino # minimizar dimensão

- $VC(h) \leq \min\left(d, \left\lceil \frac{D^2}{\rho^2} \right\rceil\right)$

Margem relativa: $\frac{\rho}{D}$

→ ao ampliar ρ^2, pode-se ter $VC(h)$ otimizado sendo independente da dimensionalidade do dado

→ ao controlar ρ^2 (margem), controlo a capacidade $VC(h)$

* Existe apenas um separador SVM ou em casos raros, podemos ter 2 iguais

Dimensão VC é a capacidade do SVM
Reduz erro treino / sem aumentar dimensão VC (minimizando)

CALCULO: $\vec{w} \cdot \vec{u} \geq c$, onde $c = -b$

taxa aprendizagem

OBJETIVO: achar W que minimiza o erro

$\vec{w} \cdot \vec{u} + b \geq \phi \rightarrow +$

- $\vec{w} \cdot \vec{x}_+ + b \geq +1$
- $\vec{w} \cdot \vec{x}_- + b \leq -1$

- $a \times (-3) + b \leq -1 \Rightarrow b \leq 3 \times a - 1$
- $a \times (-1) + b \leq -1 \Rightarrow b \leq a - 1$
- $a \times (2) + b \geq +1 \Rightarrow b \geq -2 \times a + 1$

- 0.66×-0.33 → não satisfaz como se minimiza

→ toda essa regra poderia ser substituída por $y_i \cdot (\vec{w} \cdot \vec{x} + b) - 1 = \phi$

classe separador

* Como calcular os pesos (W) e o viés (b) maximizar a margem p/ alcançar o vetor W que leva a maior margem.

$$\rho = \|r \times \vec{w}\| = 2\frac{\|\vec{w}\|}{\|\vec{w}\|^2} = \frac{2}{\|\vec{w}\|}$$

para maximizar a margem basta maximizar $\frac{2}{\|\vec{w}\|}$ que é a margem

* No M.L.P. nos regularizamos pela normal dos pesos e aqui no SVM também!!

Regularizamos pela margem

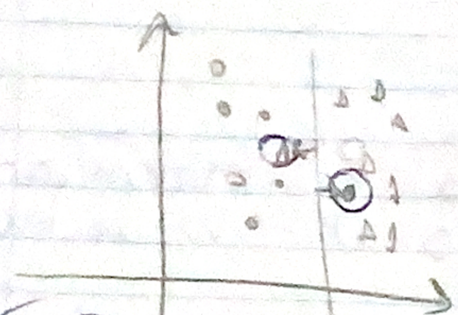
$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum_i \alpha_i \times [y_i \times (\vec{w}\cdot\vec{x}_i + b) - 1]$$

Penaliza

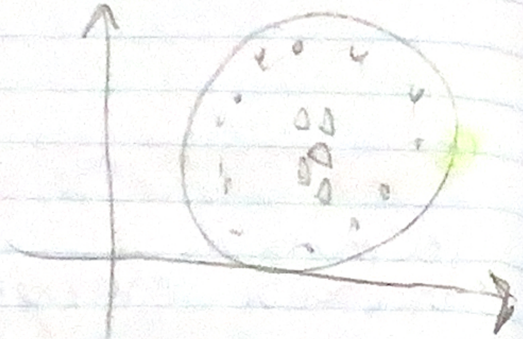
Erro Empírico

* Utiliza-se multiplicadores Lagrangianos p/ calcular o erro.

SVM é um separador ótimo quando o dado for linearmente separáveis.
Sem viés, variância máxima
erro empírico ZERO



Linearmente separável com ~~[illegible]~~ até com Anomalias



Não é linearmente separável
SOFT MARGIN

$y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \epsilon_i$

Variável folga p/ PUNIÇÃO

Busca-se minimizar o ϵ_i

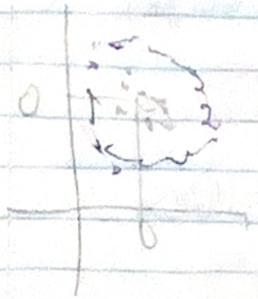
$$\frac{1}{2}\|\vec{w}\|^2 + C \times \sum \epsilon_i$$

RISCO ESTRUTURAL

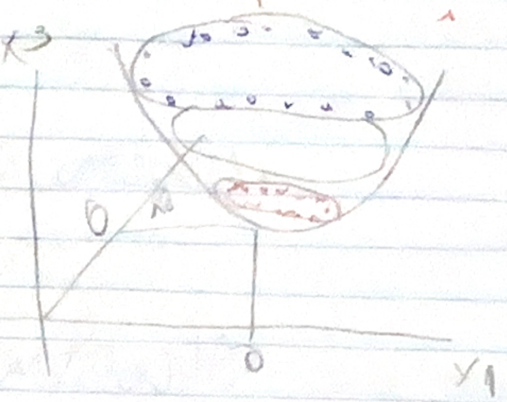
C → controla erros vs margem

C menor considera mais a margem.

KERNELS Função (inversível) que aumenta a dimensionalidade do dado para que o dado possa ser linearmente separável)



$x_3 = x_1^2 + x_2^2$



APLICAÇÃO SVM

1. kernel +
2. Erro empírico por margem
   • depois você aplica a SOFT MARGIN

TRUQUE DO KERNEL → Aplica-se função aos pontos x_1, x_2 sem ter que transformar o dado $(\vec{x}_1 \cdot \vec{x}_2)^2$

Kernel polinomial Grau 2

SVM kernel é comparável a rede neural de 2 camadas

SELEÇÃO DE MODELOS

Regularização

- ALGORITMO ARVORE DECISÃO

$$E_{in}(T) + \lambda \omega(T)$$

- ALGORITMO REDES NEURAIS

$$E_{in}(W) + \lambda \|W\|$$

- ALGORITMO SVM

$$\frac{1}{2}\|W\|^2 + C\sum \xi_i$$

SELEÇÃO DE MODELOS é encontrar o conjunto de hiperparametros ótimos por validação

$E_{val} \simeq E_{out}$ [Separamos parte dos dados de treino e T e V]

E_{out} pequeno

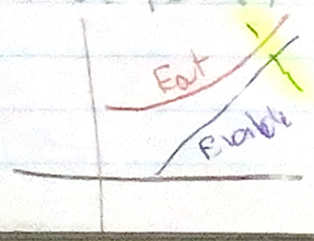
* Precisamos de um V grande

Como separar T de V p/ construir meu modelo?

LEAVE-ONE-OUT (Deixe-o de fora)

$T = n-1 \quad V = 1$

normalmente cai em overfitting



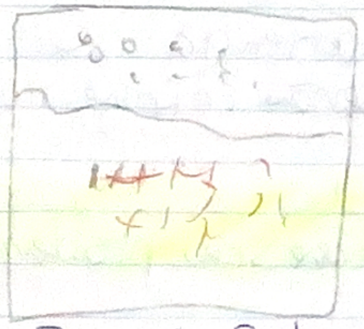
Para evitar, usa-se Cross-validation

error:

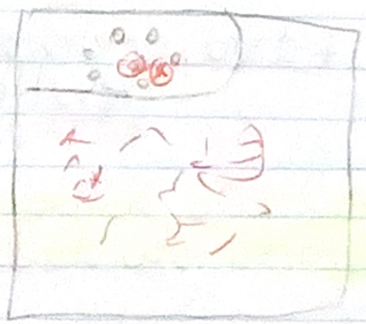
$$E_{cv} = \frac{1}{k}\sum_{i=1}^{k} \epsilon_i$$

$$E_{cv} = \frac{1}{3} \times (e_1^{T,V} + e_2^{T,V} + e_3^{T,V})$$ vamos folds

Leak vazamento de dados



$E_{in} = 0\%$ E_{out} 2,5%



$E_{in} = 0,8\%$ E_{out} 1,5%

K repetições em $\frac{n}{K}$

10-folds

$CS = K = \frac{n}{10}$

(1) Errado dizer usei 50 folds p/ diminuir E_{in}

* Correto dizer usei uma + camadas a menos p/ 50 folds pq isso diminui na arq

Treino com todos os DS

Valido de DS

Repete + media