Laboratório 6: Fully and sparsy classifiers

O objetivo deste trabalho é de estudar uma implementação dos algoritmos do classificadore perceptron com uma estrutura de rede de tipo *sparse* e comparar com a sua versão *full*. Um template de script acompanhado com funções de visualização é disponibilizado.

1 Definições e notações

O espaço dos atributos é $\mathcal{A} = \mathbb{R}^I$ onde notamos $x = (x_1, \cdots, x_I)^T$ as componentes do vetor x. O espaço da classe para o Perceptron é $\mathcal{C} = \{-1, 1\}$. Consideramos o conjunto de dados, $D = (x^n, y^n)_{n=1,\cdots,N}$ onde $x^n \in \mathbb{R}^I$ e $y^n \in C$. Recordamos que a função $s \to \text{sign}(s)$ representa a função sinal.

1.1 Fully Connected Perceptron (FCP)

O classificador *Percetron* é caracterizado pela arquitectura

$$x \in \mathbb{R}^I \to \widehat{y}(x; \widetilde{w}) = \text{sign}(\widetilde{w}^T \widetilde{x})$$

onde $\widetilde{w} \in \mathbb{R}^{I+1}$ são os parâmetros do classificador. Para avaliar a qualidade do classificador $\widehat{y}(x; \widetilde{w})$ com a base de dados D, introduzimos a função custo baseada no erro quadrado

$$E(\widetilde{w}; D) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} (y^n - \widehat{y}^n)^2, \qquad \widehat{y}^n = \widehat{y}(x^n; \widetilde{w}).$$

Recordamos que y^n é o valor real, dada pela base de dados enquato $\widehat{y}^n$ representa a predição da classe. O objetivo é de determinar os argumentos $\widetilde{w}$ que mínimisam a funcional:

$$\widetilde{w}^\star = \arg \min_{\widetilde{w} \in \mathbb{R}^{I+1}} E(\widetilde{w}; D).$$

1.2 Sparsy Connected Perceptron (SCP)

Alem do vetor $\widetilde{w}$, introduzimos o conjunto de indicios $S \subset [\![1, I]\!]$ correspondente aos coeficientes ativos. Caracterizamos S com o vetor $\widetilde{s} = (s_i)_{i=0}^I$ tal como $s_0 = 1$, $s_i = 1$ se $i \in S$, senão $s_i = 0$. Por este modo, temos

$$w_0 + \sum_{i \in S} w_i x_i = w_0 s_0 x_0 + \sum_{i=1}^{I} w_i s_i x_i = \sum_{i=0}^{I} w_i s_i x_i.$$

Definimos os *sparsy connected Perceptron* como

$$x \in \mathbb{R}^I \to \widehat{y}(x; \widetilde{w}, \widetilde{s}) = \text{sign}\Big(\sum_{i=0}^{I} s_i w_i x_i \Big).$$

A função custa associada é dada por

$$E(\widetilde{w}, \widetilde{s}; D) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} (y^n - \widehat{y}^n)^2, \qquad \widehat{y}^n = \widehat{y}(x^n; \widetilde{w}, \widetilde{s}).$$

Nota que, se $S = [\![1, I]\!]$, obtemos de novo o *Perceptron* usual onde todos os coeficientes de $\widetilde{w}$ são ativos.
O objetivo é de determinar os argumentos $\widetilde{w}$ que mínimisam as funcionais:

$$\widetilde{w}^\star, \widetilde{s}^\star = \arg \min_{\substack{\widetilde{w} \in \mathbb{R}^{I+1} \\ |\widetilde{s}|_0 = K}} E(\widetilde{w}, \widetilde{s}; D),$$

onde K é um número inteiro dado, $K \leq I$ e $|\widetilde{s}|_0$ representa a soma das componentes de $\widetilde{s}$. Por outro lado, notamos por $S^\star$ the index subset associado to vector $\widetilde{s}^\star$.

2 Implementação do sparsy Perceptron

Recordamos o algoritmo (primal) do Perceptron. Construir uma sequência $\widetilde{w}(t)$ tal como

$$\widetilde{w}(t+1) = \widetilde{w}(t) + \eta \frac{y^m - \widehat{y}^m}{2} \widetilde{x}^m$$

onde $m = m(t)$ é escolhido aleatoriamente para cada t e $\eta > 0$ a taxa de aprendizagem. A versão *sparsy* é baseada na construção do vetor $\widetilde{s}$ para desativar algumas celulas. A avaliação da qualidade é realizada pelo calculo das matrizes de confusão usando o *in-samples* e *out-samples error*.

2.1 Algoritmo S-minimization

Seja S, um conjunto de indices ativos caracterisados pelo vetor $\widetilde{s}$. Definimos o problema de minimização

$$\widetilde{w}^S = \arg \min_{\widetilde{w} \in \mathbb{R}^{S+1}} E(\widetilde{w}, \widetilde{s}; D)$$

onde o minimizante depende da escolha de S. Para este efeito, criamos uma sequência $\widetilde{w}(t) \to \widetilde{w}^S$ em $\mathbb{R}^{S+1}$ com o algoritmo seguinte.

1. Escolher (x^m, y^m), $m = m(t)$ da base de dados;

2. Calcular um predictor

$$\widetilde{\mathrm{w}}(t+1) = \widetilde{w}(t) + \eta \frac{y^m - \widehat{y}^m}{2} \widetilde{x}^m, \qquad \widehat{y}^m = \widehat{y}(x^m; \widetilde{w}(t), \widetilde{s}).$$

3. Projectar no espaço $\mathbb{R}^{S+1}$: $\mathrm{w}(t+1) \in \mathbb{R}^{I+1} \to w(t+1) \in \mathbb{R}^{S+1}$

$$w_i(t+1) = \mathrm{w}_i(t+1) s_i, \quad i = 0, \cdots, I.$$

Quando convergir, notamos por $\widetilde{w}^S \in \mathbb{R}^{S+1}$ a solução. Os trabalhos para realizar são:

- implementar o algoritmo;
- Usando a base de dados ..., experimentar diversos conjuntos S;
- Produzir as tabelas de confusões usando um *training set* e um *test set*;
- Experimentar também com a base de dados ...

2.2 Algoritmo estocastico

Seja $K \leq I$. O métoto estocástico ou de Monte-Carlo consiste em experimentar vários conjuntos S escolhidos aleatoriamentes tal como $|S| \leq K$ e criar uma sequência $\widetilde{s}^\ell$ caracterizando o conjunto S^ℓ e $\widetilde{w}^\ell \in \mathbb{R}^{S^\ell}$ tal como

$$E(\widetilde{w}^{\ell+1}, \widetilde{s}^{\ell+1}; D) \leq E(\widetilde{w}^\ell, \widetilde{s}^\ell; D).$$

O algoritmo é o seguinte.

1. Escolher um primeiro conjunto $|S^0| \leq K$ e determinar $\widetilde{w}^0$.
2. Seja S tal como $|S| \leq K$. Determinar $\widetilde{w}^S$ usando o algortimo de S-minimização. Se $E(\widetilde{w}^S, \widetilde{s}; D) \leq E(\widetilde{w}^\ell . \widetilde{s}^\ell; D)$ então $\widetilde{s}^{\ell+1} = \widetilde{s}$ e $\widetilde{w}^{\ell+1} = \widetilde{w}^S$.
3. Até convergência.

O trabalho para realizar consiste em implementar o algoritmo e experimenta-lo usando a base de dados ....

- cria um conjunto S aleatoriarmente tal como $|S| \leq K$;
- determinar $\widetilde{w}^S$ e compara-o para realizar a atualização como indicado no ponto (2).
- Produzir as tabelas de confusões usando um *training set* e um *test set*;
- Experimentar com a outra de base de dados ...

O algoritmo será utilizado para realizar a classificação da base de dados ...

2.3 Algoritmo Hard Thresholding

Seja $K \leq I$. Definimos a sequência $\widetilde{w}(t), \widetilde{s}(t)$ como

1. Escoler $(x^m, y^m) \in D$. Define

   $$\widetilde{\mathrm{w}}(t+1) = \widetilde{w}(t) + \eta \frac{y^m - \widehat{y}^m}{2} \widetilde{x}^m$$

   com $\widehat{y}^m = \widehat{y}(x^m; \widetilde{w}(t), s(t))$;
2. Definir $\widetilde{s}(t+1)$ com $s_0(t+1) = 1$ e $s_i(t+1) = 1$ para as K maiores entradas $|\mathrm{w}_i(t+1)|$, $i = 1, \cdots, I$.

3. Calcular $\widetilde{w}(t+1)$ como $w_i(t+1) = s_i(t+1)\mathrm{w}_i(t+1)$.

O algoritmo acaba quando o erro $E(\widetilde{w}(t), \widetilde{s}(t); D)$ fica bloqueado por um valor limite por baixo.

- implementar o algoritmo de Hard Thresholding.
- Usando a base de dados ..., experimentar diverses valores de K. Produzir as tabelas de confusões usando um *training set* e um *test set*. Conclusões?
- Experimentar com a outra de base de dados ...

2.4 Algoritmo Iterative Hard Thresholding (IHT)

Este algoritmo é uma alternativa ao algoritmo anterior onde calculamos de novo o minimizante para cada iteração.

1. Escoler $(x^m, y^m) \in D$. Define

$$\widetilde{\mathrm{w}}(t+1) = \widetilde{w}(t) + \eta \frac{y^m - \widehat{y}^m}{2} \widetilde{x}^m$$

com $\widehat{y}^m = \widehat{y}(x^m; \widetilde{w}(t), \widetilde{s}(t))$;

2. Definir $\widetilde{s}(t+1)$ com $s_0(t+1) = 1$ e $s_i(t+1) = 1$ para as K maiores entradas $|\mathrm{w}_i(t+1)|$, $i = 1, \cdots, I$.

3. Determinar $\widetilde{w}(t+1)$ como minimizante

$$\widetilde{w}(t+1) = \arg \min_{\widetilde{w} \in \mathbb{R}^{S(t+1)}} E(\widetilde{w}, \widetilde{s}(t+1); D)$$

calculado com o algoritmo de S-minimização.

Usando a base de dados ..., os trabalhos para realizar são de

- implementar o algoritmo de IHT;
- experimentar diverses valores de K e produzir as tabelas de confusões usando um *training set* e um *test set*;
- experimentar com a outra de base de dados ...