Laboratório 6: logistic and softmax classifiers

O objetivo deste trabalho é estudar uma implementação dos algoritmos dos classificadores logisticos e softmax para avaliar a sua eficiência. 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 $\mathcal{C} = \{c_1, \cdots, c_J\}$ é constituido de J itens que não são necessariamente valores (podem ser nominativos). Caracterizamos o output pela sua probabilidade $p = (p_1, \cdots, p_J) \in \mathcal{K}^J$, o simplex é dado por

$$\mathcal{K}^J = \Big\{ p_j \in [0,1], j = 1, \cdots, J, \text{ e } \sum_{j=1}^{J} p_j = 1 \Big\}.$$

O caso particular de output "certo" corresponde a uma distribuição p tal como $p_k = 1$, e $p_j = 0$ se $j \neq k$, o que indica que o valor de classe é o objeto c_k.

Consideramos o conjunto de dados, $D = (x^n, p^n)_{n=1,\cdots,N}$ onde $x^n \in \mathbb{R}^I$ e $p^n \in \mathcal{K}^J$. Quando se trata de eventos "certos", podemos utilizar a notação mais tradicional $D = (x^n, y^n)_{n=1,\cdots,N}$ onde $x^n \in \mathbb{R}^I$ e $y^n \in \mathcal{C}$. Neste caso identificamos a probabilidade p^n associada a sua classe. Para cada x, definimos a sua extensão $\widetilde{x} \in \mathbb{R}^{I+1}$ tal como $\widetilde{x}^T = (1, x^T)$.

1.1 Logistic classifier

No caso onde $J = 2$, temos $p_1 = 1 - p_2$ logo apenas o valor $p_1 \in [0,1]$ é necessario para classificar. Por outro lado, a classe $\mathcal{C}$ é reduzida ao sistema binário $\{0,1\}$ onde p_1 representa a probabilidade de ter o valor 1 (presente, `true`). Na prática faz-se uma identificação (abusiva) entre o valor de probabilidade $p_1 \in [0,1]$ e o valor $y \in \{0,1\}$. Neste subsecção, vamos adoptar esta identificação para simplificar as notações.
O classificador logístico é caracterizado pela função de classificação

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

onde $\widetilde{w} \in \mathbb{R}^{I+1}$ são os parâmetros do classificador e $\sigma(s)$ a função logística (ou sigmoid) dada por

$$\sigma(s) = \frac{1}{1 + e^{-s}}.$$

Rigorosamente, o classificador deveria ser uma função $\widehat{p}(x; \widetilde{w})$ mas aceitamos este abuso de notação porque tratamos de eventos certos onde $p_1^n = 0$ ou $p_1^n = 1$.

Um evento (x^m, y^m) é bem classificado com os parâmetros $\widetilde{w}$ se

$$\widehat{y}(x^m; \widetilde{w}) = \widehat{y}^m \approx y^m.$$

Para avaliar a qualidade do classificador $\widehat{y}(x; \widetilde{w})$ com a base de dados D, introduzimos a função custo baseada na entropia cruzada

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

Recordamos que y^n é a classe certa dada pela base de dados enquato $\widehat{y}^n$ representa a predição da classe. Procuramos o argumento mínimo desta funcional seja

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

Finalmente, recordamos o gradiente da função custo:

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

O cálculo do gradiente pode ser reduzido usando o método estocástico onde calculamos $(y^m - \widehat{y}^m)\widetilde{x}^m$ apenas para um evento m escolhido ao acaso.

1.2 Softmax classifier

Se $J > 2$, a extensão do logistic classifier é o softmax classifier. Desta vez, devemos usar uma notação mais rigorosa utilizando as probabilidades p. Por outro lado, os parâmetros são mais numerosos tendo em conta que os output são multinomiais. Se $\widetilde{x} \in \mathbb{R}^{I+1}$, notamos porque

$$s_j = \widetilde{w_{\bullet j}}^T \widetilde{x}, \quad \text{com} \quad \widetilde{w_{\bullet j}}^T = [w_{0j}, w_{1j}, \cdots, w_{Ij}]$$

o produto interno para a saida j. Se $W \in \mathbb{R}^{I+1\times J}$ representa a matriz cujas colunas são $\widetilde{w_{\bullet j}}$, temos

$$s = [s_1, \cdots, s_j, \cdots, s_J]^T = W^T x \in \mathbb{R}^J.$$

A partir dos s_j, calculamos as probabilidades com a função softmax

$$p_j = \widehat{p}_j(\widetilde{x}; W) = \frac{\exp(s_j)}{\sum_{k=1}^{J}\exp(s_k)}.$$

Agrupamos as probabilidades num vetor $\widehat{p}(\widetilde{x}; W)$.

Seja $D = (x^n, p^n)_{n=1,\cdots,N}$ uma base de dados, queremos procurar W tal como

$$\widehat{p}(\widetilde{x}^n; W) \approx p^n.$$

Do mesmo modo que no caso do logistic classifier, introduzimos a função custo

$$E(W; D) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{J} p_j^n \ln(\widehat{p}_j^n) \qquad \widehat{p}_j^n = \widehat{p}_j(x^n; W).$$

O gradiente para cada vector coluna $w_{\bullet j}$ é dado por

$$\nabla_{\widetilde{w_{\bullet j}}} E(W; D) = \frac{1}{N} \sum_{n=1}^{N} (p_j^n - \widehat{p}_j^n) \widetilde{x}^n$$

logo deduzimos o algoritmo de gradiente

$$\widetilde{w_{\bullet j}}(t+1) = \widetilde{w_{\bullet j}}(t) - \eta \frac{1}{N} \sum_{n=1}^{N} (p_j - \widehat{p}_j^n) \widetilde{x}^n.$$

Podemos reduzir o custo computacional substituindo o gradiente pelo gradiente estocástico usando apenas um evento.

2 Implementação do logistic classifier

2.1 Funções

Seguimos os procedimentos semelhantes ao classificador perceptron. A construção do script é baseada em 3 funções

- `predictor(x,ew)` que calcula o predictor $\widehat{y}(x; \widetilde{w})$ e retorna a probabilidade $\widehat{y}$;
- `cost(X,Y,N,ew)` que calcula o custo $E(\widetilde{w}; D)$ onde `X` e `Y` agrupam respetivemente os dados (x^n) e (y^n), $n = 1, \cdots, N$;
- `update(x,y,eta,ew)` que retorna $w(t+1)$ calculado com $w(t)$, x e y. O parâmetro η representa a taxa de aprendizagem. Introduzimos uma alteração importante que ajuda bastante a convergência. Multiplicamos o passo η com uma correção temporal `exp(-it/800)` onde `it` é o número de iterações realizadas assim que uma correção posicional `1/(1+3*r*r)` com `r=s-1/2`.

Implementar as 3 funções e experimentar com a base de dados correspondente ao AND, o OR e o XOR.

Utilizar a função adicional `C =confusion(X,Y,N,ew)` para determinar a matriz de confusão (verdareido ou falso positivo ou negativo) com os dados do *training* e os dados para a *evaluation*.

2.2 Método estocástico

A aprendizagem será realizada com uma técnica estocástica.

1) Implementar uma função `run_stocastic(X,Y,N,eta,MAX_ITER,ew,err)` que vais escolher até `MAX_ITER` elementos D da base de dados aleatoriamente. O ciclo de aprendizagem acaba quando o custo $E(\widetilde{w}; D)$ é nulo ou se percorremos o número máximo de iterações. A função retorna o último $\widetilde{w}$ calculado.

2) Experimentar várias aprendizagens quer com os ficheiros `AND.txt` ou `XOR.txt` e depois `line600.txt`.

2.3 *In-samples* e *Out-samples* erros

Carregamos a base de dados `line1500.txt` para a classificação de linhas verticais e horizontais e dividir a base de dados numa base para o *training* (50%) e uma base para a *validation*.

Experimentar diferentes opções de aprendizagens estocásticas. Determinar os valores que permitem reduzir os erros usando a matriz de confusão. Podem-se introduzimos as métricas seguintes baseadas na matriz de confusão:

$$\text{Recall} = \frac{TP}{TP+FN},\ \text{accuracy} = \frac{TP+TN}{N},\ \text{precision} = \frac{TP}{TP+FP}.$$

3 Implementação do softmax classifier

A implementação do classificador é semelhante ao caso logistic mas requer um pouco de tecnicidade devido à utilização de uma matriz para os parâmetros. Em particular, temos de trabalhar com as probabilidades e não com a classe. Substituimos os dados da classe `P` pelas suas probabilidades `P` de dimensão $N \times J$

3.1 Implementação das funções

Como no caso logístico, precisamos definir três funções fundamentais

- `predictor(x,eW)` que calcula o predictor $\widehat{y}(x;W)$ e retorna as J probabilidades associadas.
- `cost(X,P,N,W)` que calcula o custo $E(W;D)$ onde `X` e `P` agrupam respetivemente os dados (x^n) e (p^n), $n = 1, \cdots, N$.
- `update(x,p,eta,W)` que retorna $w(t+1)$ calculado com $w(t)$, x e p. O parâmetro η representa a taxa de aprendizagem. Introduzimos de novo a alteração. Multiplicamos o passo η com uma correção temporal `exp(-it/800)` onde `it` é o número de iterações realizadas. A correção posicional deve ser realizada `1/(1+3*r_j*r_j)` com `r_j=s_j-1/2` é s_j a probabilidade associada ao item c_j.

Utilizamos o metodo estocástico para realizar a aprendizagem. A definição dos parametros `epsi` assim que o controlo dos r_j é muito sensível.

3.2 *In-samples* e *Out-samples* erros

Realizamos as primeiras experiencias com os dados do logistic classifier. Por exemplo coma base de dados `line600.txt` temos as tabelas de confusão seguintes

```
logistic
in-samples error=0.904817
confusion matrix
[[615.   0.]
```

```
softmax
in-samples error=0.914635
confusion matrix
[[387.   0.]
```

```
 [ 76. 509.]]                          [ 57. 306.]]
out-samples error=0.881045            out-samples error=1.017876
[[161.   0.]                          [[377.   0.]
 [ 18. 121.]]                          [ 63. 310.]]
```

A segunda experiência corresponde a classificar imagens de dígitos contidos no ficheiros `my_digit.txt`. Temos $J = 10$ valores diferentes logo uma tabela de confusão de 10×10 cuja a diagonal apresenta os eventos corretamente classificados. Por exemplo, consegui ter a tabela seguinte

```
in-samples error=0.720342
[[195.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [ 12. 188.   0.   0.   0.   3.   0.   0.   4.   6.]
 [  4.   1. 208.   0.   0.   0.   0.   0.   0.   1.]
 [  8.   3.   0. 163.   0.   1.   0.   0.   0.   0.]
 [  0.  11.   0.   0. 174.   0.   0.   1.   0.   8.]
 [  6.   0.   1.   3.   0. 167.   0.   0.   0.   5.]
 [  1.   2.   0.   0.   0.   0. 170.   0.   0.   0.]
 [  2.   0.   1.   0.   2.   0.   0. 169.   0.   1.]
 [ 14.   3.   0.   1.   2.   0.   0.   1. 159.   0.]
 [ 21.   6.   0.   0.   8.   5.   0.   1.   0. 169.]]
out-samples error=0.861596
[[189.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [ 22. 161.   0.   0.   0.   0.   0.   1.   2.   0.]
 [  6.   4. 168.   1.   0.   0.   0.   0.   4.   1.]
 [ 10.   4.   0. 182.   0.   2.   0.   0.   0.   1.]
 [  4.   6.   0.   0. 167.   0.   1.   7.   2.   7.]
 [ 17.   0.   0.   3.   1. 165.   0.   0.   0.   3.]
 [  7.   8.   0.   0.   3.   0. 177.   0.   0.   0.]
 [  2.   0.   0.   0.   2.   0.   0. 203.   0.   1.]
 [ 20.   5.   0.   0.   0.   0.   0.   1. 139.   0.]
 [ 17.   7.   0.   9.   5.   7.   0.   4.   3. 151.]]
```

Nota a dificuldade em classificar corretemente o digito 0. Porquê?