Corrige IFT6266 - TP 3

26 octobre 2006

Q1. (a) Le critère à minimiser est (en supposant Σ fixe) :

$$\begin{aligned} C &= -\sum_{t=1}^{n} \log p(y_t|x_t) \\ &= -\sum_{t=1}^{n} \log \left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(y_t - f(x_t))'\Sigma^{-1}(y_t - f(x_t))} \right) \\ &= D + \frac{1}{2}\sum_{t=1}^{n} (y_t - f(x_t))'\Sigma^{-1}(y_t - f(x_t)) \end{aligned}$$

où D est une constante qui ne dépend pas de f. Il suffit donc de minimiser

$$C' = \frac{1}{2}\sum_{t=1}^{n} (y_t - f(x_t))'\Sigma^{-1}(y_t - f(x_t)).$$

(b) Si la matrice de covariance Σ doit également être optimisée, alors le critère devient :

$$C'' = -\frac{n}{2}\log|\Sigma^{-1}| + C'.$$

Pour une fonction f fixée, on peut alors calculer analytiquement la matrice Σ minimisant C'' en annulant la dérivée de C'' par rapport à Σ, ou, de manière équivalente, en annulant la dérivée de C'' par rapport à Σ^{-1} (ce qui est plus simple). On va utiliser les relations

$$\frac{\partial \log(|\Sigma^{-1}|)}{\partial \Sigma^{-1}} = \Sigma'$$

et (ici $A = \Sigma^{-1}$)

$$\left(\frac{\partial v'Av}{\partial A}\right)_{ij} = \frac{\partial}{\partial A_{ij}}\left(\sum_{k,l} v_k A_{kl} v_l\right) = v_i v_j = (vv')_{ij}\,.$$

Annuler la dérivée de C'' par rapport à Σ^{-1} donne alors l'équation

$$-\frac{n}{2}\Sigma' + \frac{1}{2}\sum_{t=1}^{n}(y_t - f(x_t))(y_t - f(x_t))' = 0$$

d'où

$$\Sigma = \frac{1}{n}\sum_{t=1}^{n}(y_t - f(x_t))(y_t - f(x_t))' \tag{1}$$

qui est bien une matrice de covariance (symétrique définie positive, aux cas pathologiques près, du moment que $n \geq d$).

Il reste à vérifier que c'est bien un minimum pour avoir des points boni, et pour cela il suffit de montrer que C'' est strictement convexe en Σ^{-1}. Si on note le produit scalaire entre deux matrices de mêmes dimensions par $A \cdot B = \sum_{i,j} A_{ij}B_{ij}$, cela revient à montrer que pour toute matrice $A \neq 0$, on a

$$A \cdot \left(\frac{\partial^2 C''}{\partial(\Sigma^{-1})^2} A\right) > 0.$$

En utilisant que la différentielle de l'inversion matricielle en M est l'application $N \mapsto -M^{-1}NM^{-1}$, on a alors que

$$\frac{\partial^2 C''}{\partial(\Sigma^{-1})^2} A = \frac{n}{2}\Sigma A \Sigma$$

et il suffit donc de vérifier que $A \cdot \Sigma A\Sigma > 0$. Si Σ est diagonale (avec éléments $\sigma_1, \ldots, \sigma_d$ strictement positifs sur la diagonale), alors cette quantité est simplement égale à $\sum_{i,j} A_{ij}^2 \sigma_i \sigma_j > 0$. Dans le cas général, on peut diagonaliser Σ dans une base orthonormale puisqu'elle est symétrique, c'est-à-dire $\Sigma = P'DP$ avec $PP' = P'P = I$ et D diagonale, et on peut réécrire $A = P'BP$ avec $B = PAP' \neq 0$. Si on peut montrer que pour toutes matrices M, N, on a $(P'MP)\cdot(P'NP) = M \cdot N$, on obtient donc le résultat désiré, puisque $A \cdot \Sigma A\Sigma = (P'BP) \cdot (P'DBDP)$ et qu'on peut ainsi se ramener au cas diagonal. C'est bien le cas, car, après expansion,

$$\begin{aligned}
(P'MP) \cdot (P'NP) &= \sum_{i,j,k,v,s,t} P_{ki} P_{vj} P_{si} P_{tj} N_{kv} M_{st} \\
&= \sum_{k,v,s,t} N_{kv} M_{st} \left(\sum_i P_{ki} P_{si} \right) \left(\sum_j P_{vj} P_{tj} \right) \\
&= \sum_{k,v,s,t} N_{kv} M_{st} \delta_{ks} \delta_{vt} \\
&= \sum_{k,v} N_{kv} M_{kv} \\
&= M \cdot N.
\end{aligned}$$

Le coût C'' est donc bien strictement convexe en Σ^{-1}, ce qui permet d'affirmer que (**??**) est un minimum global.

Une manière d'optimiser conjointement Σ et les paramètres de f consisterait à initialiser les paramètres θ de f aléatoirement, estimer Σ par (**??**), faire un pas de descente de gradient en considérant Σ fixe en utilisant

$$\begin{aligned}
\frac{\partial C''}{\partial \theta} = \frac{\partial C'}{\partial \theta} &= \sum_{t=1}^{n} \frac{\partial C'}{\partial f(x_t)} \frac{\partial f(x_t)}{\partial \theta} \\
&= \sum_{t=1}^{n} (f(x_t) - y_t)' \Sigma^{-1} \frac{\partial f(x_t)}{\partial \theta}
\end{aligned}$$

puis recalculer Σ par (**??**), et itérer ainsi jusqu'à ne plus observer d'amélioration significative de C''.

Mais la méthode ci-dessus est approximative, puisqu'à chaque étape de descente de gradient on suppose Σ fixe, alors que, comme le montre (**??**), Σ dépend de f. Il faudrait donc calculer le vrai gradient par rapport à θ en tenant compte de cette interaction, par

$$\frac{\partial C''}{\partial \theta} = -\frac{n}{2} \frac{\partial \log |\Sigma^{-1}|}{\partial \Sigma^{-1}} \frac{\partial \Sigma^{-1}}{\partial \Sigma} \frac{\partial \Sigma}{\partial \theta} + \frac{1}{2} \sum_{t=1}^{n} \left(\frac{\partial C_t'}{\partial f(x_t)} \frac{\partial f(x_t)}{\partial \theta} + \frac{\partial C_t'}{\partial \Sigma^{-1}} \frac{\partial \Sigma^{-1}}{\partial \Sigma} \frac{\partial \Sigma}{\partial \theta} \right)$$

où $C_t' = (y_t - f(x_t))' \Sigma^{-1} (y_t - f(x_t))$. Tous ces termes sont assez facilement calculables, et on peut donc faire de la descente de gradient (il faudra juste inverser Σ à chaque itération de l'algorithme).

(c) Dans le cas où $\Sigma(x)$ doit être obtenue à partir de la sortie $S(x)$ (sous forme matricielle) d'un réseau de neurones, on peut faire un peu la même chose que dans (**?**), c'est-à-dire paramétriser $\Sigma(x)$ sous la forme $S(x)S(x)' + \lambda I$ avec $\lambda > 0$ fixé, ce qui assure d'obtenir une matrice symétrique définie positive. Le gradient peut se calculer comme mentionné ci-dessus (à ceci près qu'on ne peut plus sortir le déterminant de $\Sigma(x_t)$ de la somme sur t), et on utilisera

$$\frac{\partial \Sigma(x)}{\partial \theta} = \frac{\partial \Sigma(x)}{\partial S(x)} \frac{\partial S(x)}{\partial \theta}$$

avec $\frac{\partial \Sigma(x)_{ij}}{\partial S(x)_{kl}} = \delta_{ik} S(x)_{jl} + \delta_{jk} S(x)_{il}$.

Q2. Soit $\tilde{Y}$ la variable aléatoire représentant vraie classe. On a donc $P(Y = \tilde{Y}|X = x) = \epsilon$ et $f_i(x) = P(\tilde{Y} = i|X = x, f)$ (où conditionner en f signifie "en supposant que les données suivent la loi de notre modèle"). Le critière de maximum de vraisemblance pour la classification probabiliste est :

$$\begin{aligned} C &= -\sum_t \log P(Y = y_t|X = x_t, f) \\ &= -\sum_t \log \Big(P(Y = y_t|\tilde{Y} = y_t, X = x_t, f) P(\tilde{Y} = y_t|X = x_t, f) \\ &\quad + P(Y = y_t|\tilde{Y} = 1 - y_t, X = x_t, f) P(\tilde{Y} = 1 - y_t|X = x_t, f) \Big) \\ &= -\sum_t \log \left((1 - \epsilon) f_{y_t}(x_t) + \epsilon f_{1-y_t}(x_t) \right) \\ &= -\sum_t \log \left((1 - \epsilon) f_{y_t}(x_t) + \epsilon (1 - f_{y_t}(x_t)) \right). \end{aligned}$$

On remarque que si $\epsilon = \frac{1}{2}$, C ne dépend plus de f: on ne peut en effet rien apprendre de données pour lesquelles les observations ne donnent aucune information sur la vraie cible.

Q3. Soit $\sigma(x) = \text{softmax}(x)$, i.e.

$$\sigma_i(x) = \frac{e^{x_i}}{\sum_k e^{x_k}}.$$

Alors

$$\frac{\partial \sigma_i(x)}{\partial x_j} = \frac{\delta_{ij} e^{x_i}}{\sum_k e^{x_k}} - \frac{e^{x_i + x_j}}{(\sum_k e^{x_k})^2} = \delta_{ij} \sigma_i(x) - \sigma_i(x) \sigma_j(x) = \sigma_i(x)(\delta_{ij} - \sigma_j(x)).$$

et par la formule des dérivées en chaîne, si $Q = -\log f_y(x)$ où $f(x) = \sigma(a(x))$:

$$\begin{aligned} \frac{\partial Q}{\partial a_i(x)} &= \frac{\partial Q}{\partial f(x)} \frac{\partial f(x)}{\partial a_i(x)} \\ &= \left(0, \ldots, 0, -\frac{1}{f_y(x)}, 0, \ldots, 0 \right) \frac{\partial f(x)}{\partial a_i(x)} \\ &= -\frac{1}{f_y(x)} \frac{\partial f_y(x)}{\partial a_i(x)} \\ &= -\frac{1}{f_y(x)} f_y(x)(\delta_{yi} - f_i(x)) \\ &= f_i(x) - \delta_{yi}. \end{aligned}$$

Q4. La significativité des résultats dépend grandement du choix des 50 partitions. Voici deux exemples typiques d'exécution du programme ci-dessous, chacun menant à une conclusion différente :

- laissant penser que la différence est statistiquement significative :

```
E       [MSE_valid - MSE_test] = -0.184032547451
VAR     [MSE_valid - MSE_test] = 0.0949009371198
STDDEV  [MSE_valid - MSE_test] = 0.308059957021
STDERROR[MSE_valid - MSE_test] = 0.0435662569243
The difference is statistically significative
```

- laissant penser que la différence n'est pas statistiquement significative :

```
E       [MSE_valid - MSE_test] = -0.0959738635443
VAR     [MSE_valid - MSE_test] = 0.164478739055
STDDEV  [MSE_valid - MSE_test] = 0.40555978481
STDERROR[MSE_valid - MSE_test] = 0.0573548148032
The difference is NOT statistically significative
```

La taille de l'ensemble V de validation est trop petite pour obtenir des statistiques fiables (avec 50 répétitions), même si le signe de la différence semble au moins être systématiquement négatif, indiquant que l'hypothèse selon laquelle notre estimateur est optimiste est sans doute vraie. Voici le code(ecrit par Olivier Delalleau) :

```
import random
from math import sqrt
from numpy import dot, exp, Infinity, mean, repmat, reshape, std, sum, var, \
                  zeros
from pylab import load

############
## parzen ##
############
def parzen(x, train, sigma_square):
    """Compute the output of Parzen windows regression on test points x, with
    training dataset train and Gaussian kernel with variance sigma_square"""
    x_width = x.shape[1]
    k_x_xi = reshape(
            exp(-0.5 * sum((repmat(x, 1, len(train)).reshape(-1, x_width) -
                            repmat(train[:, 0:x_width], len(x), 1))**2,
                            axis = 1) \
                     / sigma_square),
            (len(x), len(train)))
    k_x_xi /= sum(k_x_xi, axis = 1).reshape(-1, 1)
    return dot(k_x_xi, train[:, x_width:])

################
## parzen_mse ##
################
def parzen_mse(train, test, sigma):
    """Compute the Mean Squared Error (MSE) of Parzen windows applied on the
    'test' set, with training set 'train' and kernel bandwidth 'sigma'."""
    return mean((test[:,1:] - parzen(test[:,0:1], train, sigma**2))**2)

##################
## random_split ##
##################
def random_split(set, n1, n2):
    """Return a split of 'set' into two random subsets, respectively of length
    n1 and n2"""
    indices = range(len(set))
    random.shuffle(indices)
    return [ set[indices[0:n1]], set[indices[n1:n1+n2]] ]

##########
## main ##
##########
def main():
    """Main body of the program"""

    # Load datasets.
    train = load('train_data.txt')
    valid = load('valid_data.txt')

    n_tries = 50
```

```
    diff = zeros(n_tries)
    for i in range(n_tries):
        # First randomly split the validation set into (valid, test).
        [ V, T ] = random_split(valid, 10, 90)
        # Then find the optimal value of sigma on V.
        best_valid_mse = Infinity
        for sigma in [ 0.005, 0.015, 0.020, 0.025, 0.030, 0.035, 0.1 ]:
            mse = parzen_mse(train, V, sigma)
            if mse < best_valid_mse:
                best_valid_mse = mse
                best_sigma = sigma
        # Compute MSE on the test set with the best value of sigma we found.
        test_mse = parzen_mse(train, T, best_sigma)
        # And store the difference between validation and test MSE.
        diff[i] = best_valid_mse - test_mse

    # Output difference and test whether it is statistically significative.
    mean_diff = mean(diff)
    std_diff = std(diff)
    stderr_diff = std(diff) / sqrt(n_tries)
    print """\
E        [MSE_valid - MSE_test] = %s
VAR      [MSE_valid - MSE_test] = %s
STDDEV   [MSE_valid - MSE_test] = %s
STDERROR[MSE_valid - MSE_test] = %s""" \
        % (mean_diff, var(diff), std_diff, stderr_diff)

    if abs(mean_diff) > 2 * stderr_diff:
        print 'The difference is statistically significative'
    else:
        print 'The difference is NOT statistically significative'

# Run.
main()
```