SB2b HT 2017 - Problem Sheet 3

1. The binary Naive Bayes classifier has interesting connections to the logistic regression classifier. You will show that, under certain assumptions, the Naive Bayes likelihood function is identical in form to the likelihood function for logistic regression. You will then derive the MLE parameter estimates under these assumptions.

   (a) Suppose $X = \{X_1, \ldots, X_D\}$ is a continuous vector in $\mathbb{R}^D$ representing the features, and Y is a binary random variable with values in $\{0, 1\}$ representing the class labels. Let the following assumptions hold:

   - The label variable Y follows a Bernoulli distribution, with parameter $\pi = P(Y = 1)$.
   - For each feature X_j, we have $P(X_j|Y = k)$ follows a Gaussian distribution of the form $\mathcal{N}(\mu_{jk}, \sigma_j)$.

   Using the Naive Bayes assumption that states *"for all $j' \neq j$, X_j and $X_{j'}$ are conditionally independent given Y"*, compute $P(Y = 1|X)$ and show that it can be written in the following form:

   $$P(Y = 1|X) = \frac{1}{1 + \exp(-w_0 + \mathbf{w}^\top \mathbf{X})}.$$

   Specifically, you need to find the explicit form of w_o and $\mathbf{w}$ in terms of π, μ_{jk}, σ_j, for $j = 1, \ldots, D$ and $k \in \{0, 1\}$.

   (b) Suppose a training set with N examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ is given, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})^\top$ is a D-dimensional feature vector, and $y_i \in \{0, 1\}$ is its corresponding label. Using the assumptions in 1.a (not the result), provide the maximum liklihood estimation for the parameters of the Naive Bayes with Gaussian assumption. In other words, you need to provide the estimates for π, μ_{jk}, and σ_j, for $j = 1, \ldots, D$ and $k \in \{0, 1\}$.

2. Assume you are interested in clustering n binary images. Binary images are modelled as i.i.d. samples $\{x_i\}_{i=1}^n$ for each $i = 1, \ldots, n$ from a random vector $X_i = (X_{i1}, \ldots, X_{ip})$ of p binary random variables (p being the number of pixels). Probability mass function of X_i is a mixture with mixing proportions $\pi_1, \ldots, \pi_K$ satisfying $\pi_k \geq 0$ for each k and $\sum_{k=1}^K \pi_k = 1$, and each mixture component k is modelled as a product of p independent Bernoulli variables with parameters $\phi_k = (\phi_{k1}, ..., \phi_{kp}) \in [0, 1]^p$. In other words, $Z_i \sim$ Discrete $(\pi_1, \ldots, \pi_K)$ is the variable on $\{1, \ldots, K\}$ indicating which component X_i belongs to, and $X_{ij}|Z_i = k \sim$ Bernoulli (ϕ_{kj}) independently for $j = 1, \ldots, p$.

   (a) Having observed dataset $\{x_i\}_{i=1}^n$, write down the log-likelihood explicitly as a function of the parameters $\theta = (\pi_k, \phi_k)_{k=1}^K$.

   (b) We want to estimate the unknown parameters by maximizing the log-likelihood using the EM algorithm. Denote by z_i a value in $\{1, \ldots, K\}$, and $\mathbf{z} = (z_i)_{i=1}^n \in \{1, \ldots, K\}^n$. Let us write the "variational free energy" $\mathcal{F}(\theta, q)$ as a function of $q(\mathbf{z})$ and of the parameters (π_k, ϕ_k), where the variational free energy is so-called because it is the sum of the energy $E_q[\log P(Z, X)]$ plus the entropy of q:

$$\mathcal{F}(\theta, q) = \sum_{\mathbf{z}} q(\mathbf{z}) \left[\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{1}(z_i = k) \left(\log \pi_k + \sum_{j=1}^{p} \left[x_{ij} \log \phi_{kj} + (1 - x_{ij}) \log(1 - \phi_{kj}) \right] \right) \right] - \sum_{\mathbf{z}} q(\mathbf{z}) \left[\log q(\mathbf{z}) \right].$$

Derive explicitly the EM update equations by setting derivatives of $\mathcal{F}$ w.r.t. π_k and ϕ_{kj} to zero and solving.

3. Let $x_1, \ldots, x_n$ be a dataset of p-dimensional vectors and $C = \{C_1, C_2, \ldots, C_K\}$ a partition of $\{1, \ldots, n\}$. For each cluster C_k, denote $n_k = |C_k|$ and define

$$\bar{x}_k = \frac{1}{n_k} \sum_{i \in C_k} x_i \qquad \text{to be the within-cluster mean}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^{K} n_k \bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \text{to be the overall mean}$$

and

$$T = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - \bar{x})(x_i - \bar{x})^\top \qquad \text{to be the total deviance to the overall mean}$$

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - \bar{x}_k)(x_i - \bar{x}_k)^\top \qquad \text{to be the within-cluster deviance to the cluster mean}$$

$$B = \sum_{k=1}^{K} n_k (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^\top \qquad \text{to be the between-cluster deviance}$$

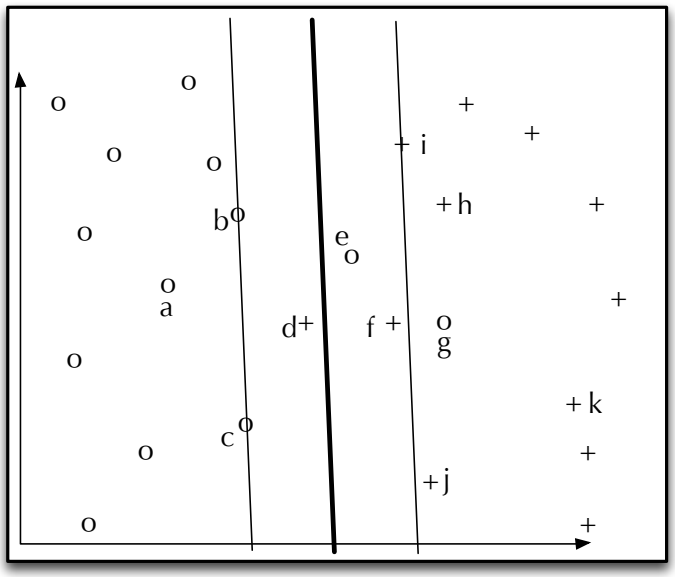
where T, W and B are all $p \times p$ matrices.

(a) Verify that $T = W + B$.

(b) Explain how the K-means objective is related to W.

(c) How does T change during the course of the K-means algorithm? How does B change?

4. The figure below shows a binary classification dataset and the optimal the decision boundary and margins of a soft-margin SVM.



(i) Which of the points a,...,k are support vectors?

(ii) For points a, b and d what are the range of possible values for the corresponding dual variables?

5. Parameter C in C-SVM can sometimes be hard to interpret. An alternative parametrization is given by ν-SVM:

$$\min_{w,b,\rho,\xi} \left(\frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n} \xi_i \right)$$

subject to

$$\begin{aligned} \rho &\geq 0, \\ \xi_i &\geq 0, \\ y_i \left(w^\top x_i + b \right) &\geq \rho - \xi_i. \end{aligned}$$

(note that we now directly adjust the constraint threshold ρ).

Using complementary slackness, show that ν is an upper bound on the proportion of non-margin support vectors (margin errors) and a lower bound on the proportion of all support vectors with non-zero weight (both those on the margin and margin errors). You can assume that $\rho > 0$ at the optimum (non-zero margin).

6. (OPTIONAL): This problem is going to give you the chance to try out TensorFlow, one of a number of popular deep learning packages. TensorFlow must be installed in python, but once installed there is an R library. Instructions:

   - Install TensorFlow using the instructions here: `https://www.tensorflow.org/get_started/os_setup`
   - Verify that you can use TensorFlow in python:

```
$ python
>>> import tensorflow
>>>
```

- If you would like to use R, follow the instructions here: `https://github.com/rstudio/tensorflow`

Returning to the digit classification problem from problem sheet 2 (part 5), try once again to improve performance using a multi-layer neural network. Hint: use a convolutional layer (`conv2d` in TensorFlow).