IFT3395/6390, Assignment 1

Date posted: Oct. 1, 2014,
Due: Oct. 8, 2014, at the *start* of class

---

1. (1/10) Show that the maximum likelihood estimates for the parameters of a Bernoulli distribution, given training observations $x_1, \ldots, x_N$ is $\frac{1}{N}\sum_n x_n$ by setting the derivative of the log-likelihood to zero.

2. (1/10) In class, we discussed Gaussian generative models for classification, and discrete Naive Bayes classifiers. Discuss how one may define a *Gaussian naive Bayes classifier*. Describe what its parameters are, and write down the maximum likelihood estimates.

3. (3/10) In this question you will "misuse" a multi-dimensional linear regression model to solve a classification task.

   Download the following files:

   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/train_images.txt`

   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/test_images.txt`

   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/train_labels.txt`

   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/test_labels.txt`

   and save them locally. Take a look at the contents of the files. The files contain 1000 binary train and test images of handwritten digits and corresponding class-labels, each encoded as comma-separated matrix of real numbers. More specifically, the first two files each represent 1000 binary images of size 28×28 pixels (there is one image per row). The last two files each represent 1000 labels in one-hot encoding (there is one 10-dimensional label vector per row). Files one and three represent training data, files two and four represent test data.

   Use the *training data* to estimate a regularized linear regression model (without bias term), that predicts the 10-dimensional output vector $\mathbf{t}$ from the 784-dimensional input vector $\mathbf{x}$.

   Recall that the model is defined by the (784×10)-matrix $\mathbf{W}$ that minimizes the cost $E(\mathbf{W}) = \sum_n \|\mathbf{t}_n - \mathbf{W}^{\mathrm{T}}\mathbf{x}_n\|^2 + \lambda \sum_k \|\mathbf{w}_k\|^2$ where $\mathbf{w}_k$ is a column of $\mathbf{W}$, and λ is a constant that has to be chosen by you. For training, you can either use gradient descent or use the closed form solution discussed in class. Experiment with different values for λ (or likewise, for λ in $(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda I)^{-1}$ in the closed form solution). You may find that for some values, learning becomes unstable or does not work.

After estimating the regression model, classify the training and test cases as follows: For each training/test case $\mathbf{x}$ compute $\mathbf{W}^{\mathrm{T}}\mathbf{x}$. Define the class label for this input to be the index of the dimension that yields the largest value. In other words your classier output for input $\mathbf{x}$ is defined as:

$$\mathrm{argmax}_k \mathbf{w}_k^{\mathrm{T}}\mathbf{x}$$

Hint:

- If you use python, you can use the function numpy.linalg.inv() to invert a matrix.

What to hand in: Of all values you tried, which value for the regularization parameter, λ, worked best on the test data? For this value, what is the mis-classification rate on your training data and what is the mis-classification on the test data? Do not hand in any code.

4. (4/10) Estimate a *Bernoulli Naive Bayes classifier* using the data in the previous question, and apply the classifier on both the training and the test data.

   What to hand in:

   - The 10 prior class-probabilities.
   - Ten images of size 28×28 pixels, each showing the class-conditional probabilities $p(\mathbf{x}|\mathcal{C}_k)$ for class $\mathcal{C}_k$ as a gray value image.
   - Two numbers: The mis-classification rate on the training data and the mis-classification on the test data.

   Hints:

   - To learn the naive Bayes classifier, you will need to estimate prior probabilities over classes as well as class-*conditional* Bernoulli probabilities over the 784-dimensional image vectors.
   - You may want to represent all probabilities using their log's when doing any calculations with them, to avoid underflow.
   - To avoid taking the log of 0.0 you can do the following: When estimating the class-conditional probabilities, assume your training dataset contains *two* extra images for each class: One all-white image consisting only of ones, and one all-black image consisting only of zeros. (This means, you assume the total number of images in the training set is 1020 not 1000.)
   - If you use python/matplotlib, you can use pylab.imshow() for visualizing images. Reshape the 784-dimensional vectors into matrices of size 28×28 before passing them to imshow(). You can use pylab.gray() to set the colormap so images are represented with gray values.

5. (1/10) A classification neural network combines K class "scores" y_k into a probability over classes by using the softmax function

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\exp(y_k(\mathbf{x}))}{\sum_j \exp(y_j(\mathbf{x}))}$$

What is the derivative of the log-probability of a training example under this model, $\log p(t_n|\boldsymbol{x}_n)$, with respect to one input unit y_k?