ELEN 4903: Machine Learning

Columbia University, Spring 2016

Homework 2: Due February 26, 2016 by 11:59pm

Please read these instructions to ensure you receive full credit on your homework. Submit the written portion of your homework as a *single* PDF file through Courseworks (less than 5MB). In addition to your PDF write-up, submit all code written by you in their original extensions through Courseworks (e.g., .m, .r, .py, etc.). Any coding language is acceptable. Do not wrap your files in .rar, .zip, .tar and do not submit your write-up in .doc or other file type. Your grade will be based on the contents of *one* PDF file and the original source code. Additional files will be ignored. We will not run your code, so everything you are asked to show should be put in the PDF file. Show all work for full credit.

Late submission policy: Late homeworks will have 0.1% deducted from the final grade for each minute late. *Your homework submission time will be based on the time of your* ***last*** *submission to Courseworks. I will not revert to an earlier submission!* Therefore, do not re-submit after midnight on the due date unless you are confident the new submission is significantly better to overcompensate for the points lost. Submission time is non-negotiable and will be based on the time you submitted your last file to Courseworks. The number of points deducted will be rounded to the nearest integer.

Problem 1 (multiclass logistic regression) – 30 points

Logistic regression with more than two classes can be done using the softmax function. For data $x \in \mathbb{R}^d$ and k classes (where class i has regression vector w_i) the class y of x is distributed as

$$P(y|x, w_1, \ldots, w_k) = \prod_{i=1}^{k} \left(\frac{e^{x^T w_i}}{\sum_{j=1}^{k} e^{x^T w_j}} \right)^{\mathbb{1}(y=i)}.$$

1. Write out the log likelihood $\mathcal{L}$ of data $(x_1, y_1), \ldots, (x_n, y_n)$ using an i.i.d. assumption.
2. Calculate $\nabla_{w_i} \mathcal{L}$ and $\nabla^2_{w_i} \mathcal{L}$.

Problem 2 (Gaussian kernels) – 15 points

We saw how we can construct a kernel between two points $u, v \in \mathbb{R}^d$ using the dot product (or integral) of their high-dimensional mappings $\phi(u)$ and $\phi(v)$. In the integral case,

$$k(u, v) = \int_{\mathbb{R}^d} \phi_t(u)\phi_t(v)dt,$$

where t is some parameter that is integrated out. Show that the mapping

$$\phi_t(u) = \frac{1}{(2\pi\nu)^{d/2}} \exp\left\{-\frac{\|u-t\|^2}{2\nu}\right\}$$

reproduces the Gaussian kernel $k(u,v) = \alpha \exp\left\{-\frac{\|u-v\|^2}{\beta}\right\}$ for an appropriate setting of α and β.

Hint: This will be difficult to do without using properties of multivariate Gaussians and their marginal distributions. Try framing the integral as a marginalization problem with two probability distributions.

Problem 3 (Classification) – 75 points

In this problem you will implement three classifiers and run them on the MNIST Handwritten Digits data set posted on Courseworks and the class website. Do not preprocess the data *except* in the way discussed at the end of the README below. You must use your own implementation of the following classifiers to receive full credit. Information about the data is given below.

All three sub-problems ask for you to show your results in a 10×10 "confusion matrix" (call it C). This can be done as follows: For each of the 500 predictions you make from the test set, let y_t be the *true* class label and y_p be the *predicted* class label using your algorithm. Update $C(y_t, y_p) \leftarrow C(y_t, y_p) + 1$ for each prediction. At the end, C should sum to 500 and each row should sum to 50. (C can then be normalized, but leave it unnormalized for this assignment.)

See the README below regarding the images you are asked to show.

Problem 3a (20 points) :

- Implement the k-NN classifier for $k = 1, 2, 3, 4, 5$.
- For each k calculate the confusion matrix and show the trace of this matrix divided by 500. This is the prediction accuracy. You don't need to show the confusion matrix.
- For $k = 1, 3, 5$, show three misclassified examples as images and indicate the true class and the predicted class for each one.

Problem 3b (25 points) :

- Implement the Bayes classifier using a class-specific multivariate Gaussian distribution. Derive the maximum likelihood estimate for the mean and covariance for a particular class j. Show the answer you obtain for the mean and covariance, as well as the estimate for the class prior.
- Show the confusion matrix in a table. As in Problem 3a, indicate the prediction accuracy by summing along the diagonal and dividing by 500.
- Show the mean of each Gaussian as an image using the provided Q matrix.
- Show three misclassified examples as images and show the probability distribution on the 10 digits learned by the Bayes classifier for each one.

Problem 3c (30 points) : Please see the README for an important data preprocessing step!

- Implement the multiclass logistic regression classifier you derived in Problem 1.

  (COMMENT: You technically only need to use $\nabla_w \mathcal{L}$ to do this. In this case, you might want try a stepsize on the order of $\rho = 0.1/5000$.)

- After making an update of each $w_0, \ldots, w_9$, calculate $\mathcal{L}$ (see Problem 1) and plot as a function of iteration. Run your algorithm for 1000 iterations.

  (COMMENT: 1000 iterations means each w_i has been updated 1000 times.)

- Show the confusion matrix in a table. Indicate the prediction accuracy by summing along the diagonal and dividing by 500.

- Show three misclassified examples as an image and show the probability distribution on the 10 digits learned by the softmax function for each one.

```
DATA README

The following data is included:

Xtrain : 20 x 5000 matrix of training data for digits 0 through 9.

label_train : 5000 dimensional vector containing the corresponding
class (digit) in Xtrain.

Xtest : 20 x 500 matrix of testing data for digits 0 through 9.

label_test : 500 dimensional vector containing the corresponding
class (digit) in Xtest.

Q : 784 x 20 matrix of principal components. You will only need this
matrix for visualizing the images asked for. The original data contains
28 x 28 images that are vectorized into 784 x 1 vectors. I projected the
data onto its first 20 "principle components" to reduce the dimensionality
of the problem. These components are contained in Q. You can see the
information captured by these 20 vectors as follows: For a 20 x 1 vector
"x" from a training or testing matrix, let y = Q*x. Then show this as an
image, e.g., in Matlab using imagesc(reshape(y,28,28)). The image will
probably need to be rotated and/or flipped. Some information is lost in
this process (and the data was centered) so the resulting images won't
be totally clear, but the digit should be recognizable.


***IMPORTANT (FOR PART 3c ONLY)***
Before training and testing the classifier, be sure to add a 21st
dimension to the data set equal to one in order to allow for a shift
in the decision boundary! Therefore, the classification vectors
w_0, ..., w_9 should be 21 dimensions. And the 21st row of X show
be entirely equal to one.
```