

CS 475 Machine Learning (Spring 2017): Assignment 5

Due on May 5th, 2017 at 3:00PM

Raman Arora

Instructions: Please read these instructions carefully and follow them precisely. Feel free to ask the instructor if anything is unclear!

1. Please submit your solutions electronically via [Gradescope](#).
2. Please submit a PDF file for the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want: typeset the solution in LATEX (recommended), type it in Word or a similar program and convert/export to PDF, or even hand write the solution (legibly!) and scan it to PDF. We recommend that you restrict your solutions to the space allocated for each problem; you may need to adjust the white space by tweaking the argument to `\vspace{xpt}` command. Please name this document `<firstname-lastname>-sol5.pdf`.
3. Submit the empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file. In addition, we require that you save the Python notebook as a pdf file and append it to the rest of the solutions.
4. In addition, you will need to submit your predictions on digit recognition and sentiment classification tasks to [Kaggle](#), as described below, according to the competition rules.
5. **Late submissions:** You have a total of 72 late hours for the entire semester that you may use as you deem fit. **After you have used up your quota, there will be a penalty of 100% of your grade on late submission.** Note that this is different from previous homework assignments for which we gave you additional 48 hours with 50% penalty. This is to ensure release of solutions for HW5 in time for the final exam.
6. **What is the required level of detail?** When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include the figure as well as the code used to plot it. If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important. When submitting code, please make sure it's reasonably documented, and describe succinctly in the written component of the solution what is done in each `py`-file.

Name: _____

I. Generative Models

Here we will consider the Gaussian generative model for $\mathbf{x} \in \mathbb{R}^d$ which assumes equal, isotropic covariance matrices for each class:

$$\Sigma_c = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \sigma_d^2 \end{bmatrix}. \quad (1)$$

The classes of course have separate means. That is, the conditional density of the j -th coordinate of \mathbf{x} given class c is a Gaussian $\mathcal{N}(\mu_{c,j}, \sigma_j)$, with these densities independent (given class) for different values of j .

For simplicity, let's consider a two-class problem, with $y \in \{0, 1\}$. As we discussed in class, when training the generative model, we simply fit $p(\mathbf{x} | y)$ and $p(y)$ to the training data, and use the resulting discriminant analysis to produce a decision rule which predicts

$$\hat{y}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \{p(\mathbf{x} | y = c) p(y = c)\}. \quad (2)$$

However, this model does also produce an (implicit) estimate for the posterior $p(y = c | \mathbf{x})$.

Problem 1 [35 points] Show that the posterior $p(y = c | \mathbf{x})$ resulting from the generative model above has the same form as the posterior in logistic regression model,

$$p(y = c | \mathbf{x}) = \frac{1}{1 + \exp(w_0 + \mathbf{w} \cdot \mathbf{x})}, \quad (3)$$

for appropriate values of $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^d$. (*Advice: Start with Bayes rule, and use the simplifying assumptions in (1) to derive the posterior.*)



Problem 2 [15 points] The previous problem established that the two models – logistic regression and the linear discriminant analysis based on the isotropic Gaussian model (1) – have the same form of the posterior $p(y|\mathbf{x})$. Will the two models produce the same classifier when applied to a given training set? Why or why not?

II. Multilayer neural networks

In this section we will return to the problem of handwritten digit recognition and attack it with a two-layer neural network. The general architecture of the network is as follows: let \mathbf{x} be the input (pixels of the $d \times d$ digit image). Then, the first layer (with k hidden units) computes

$$\mathbf{f}_1(\mathbf{x}) = h(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad (4)$$

where \mathbf{W}_1 is a $k \times d^2$ matrix of weights, \mathbf{b}_1 is the k -dimensional vector of bias terms, and h is a nonlinearity (activation function for the hidden units); the syntax in (4) means that h is computed on each element of its k -dimensional argument vector (the output of $\mathbf{W}_1\mathbf{x} + \mathbf{b}_1$), yielding k -dimensional activation vector, called in neural network literature the feature map of the layer.

The second layer computes the network activation:

$$\mathbf{f}_2(\mathbf{x}) = \mathbf{W}_2\mathbf{f}_1(\mathbf{x}) + \mathbf{b}_2, \quad (5)$$

where \mathbf{W}_2 is a $10 \times k$ matrix and \mathbf{b}_2 a 10-dimensional vector. The values of the 10-dimensional vector \mathbf{f}_2 are interpreted as the scores (unnormalized log-probabilities) for the 10 classes. So, the posterior for class c computed by the network is

$$p(c|\mathbf{x}) = \frac{\exp(f_{2,c}(\mathbf{x}))}{\sum_j \exp(f_{2,j}(\mathbf{x}))}, \quad (6)$$

and the loss used to train the network is

$$-\frac{1}{n} \sum_{i=1}^n \log p(y_i|\mathbf{x}_i) \quad (7)$$

where the summation is over the indices of training examples. You can of course add regularization (e.g., squared norm penalty over elements of \mathbf{W}_1 , \mathbf{W}_2) to the loss.

Again, we will revisit and improve our results on a familiar data set, this time MNIST.

Problem 3 [50 points] Implement a two-layer neural network for MNIST. We have set up a Kaggle competition for the clean version:

- <https://inclass.kaggle.com/c/cs475-neuralnet-mnist>

We have provided skeleton code `PS5_neuralnet.ipynb` that features a complete forward and backward pass. However, you will have to work out what the gradients are.

We provide several different update algorithms in `utils.py`: SGD, SGD with momentum and SGD with Nestorov momentum. Try these methods, and report the differences in their performances.

You should also decide what the activation function h is. The options are ReLU, Logistic and Tanh. For all activations you try, you need to implement their forward and backward functions.

This task will require you to understand the code that we have provided, so read the inline documentation carefully. We encourage you to re-use parts of your code and hyperparameters from

Problem Set 2, e.g. `softmaxloss`, stopping criteria, stepsize reduction, `batch_size`, etc.

If you used loops in your solution in Problem Set 2, you will quickly find that it is not fast enough. Learn to write things in terms of matrix multiplications.

Optional:

More layers You can try more than two-layers, and it should be trivial given the code we provide.

Dropout Dropout is a popular technique in neural network. Instead of using (5), the feed-forward operation during training becomes:

$$r_j \sim \text{Bernoulli}(p) \tag{8}$$

$$\mathbf{f}_2(\mathbf{x}) = \mathbf{W}_2\left(\frac{1}{p}\mathbf{r} * \mathbf{f}_1(\mathbf{x})\right) + \mathbf{b}_2, \tag{9}$$

\mathbf{r} is a vector of independent Bernoulli random variables each of which has probability p of being 1. This vector is multiplied element-wisely to the output of previous layer. Some of the outputs of previous layer are dropped out, and the remained are scaled by $\frac{1}{p}$ to make sure the "scale" of the input to remain the same. During evaluation, we do the same thing as (5).