TTIC 31020: Introduction to Statistical Machine Learning
Autumn 2017

Problem Set #4

Out: November 18, 2017

Due: Tuesday November 28, 11:59pm

Instructions

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with 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. Please name this document ⟨`firstname-lastname`⟩`-sol1.pdf`.
2. 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. Name the notebook ⟨`firstname-lastname`⟩`-sol1.ipynb`.

Late submissions: there will be a penalty of 25 points for any solution submitted within 24 hours past the deadlne. No submissions will be accepted past then.

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 int the `ipynb` file 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. If there is a mathematical answer, provide is precisely (and accompany by only succinct words, if appropriate).

When submitting code, please make sure it's reasonably documented, and describe succinctly in the written component of the solution what is done where.

Collaboration policy : collaboration is allowed and encouraged, as long as you (1) write your own solution entirely on your own, (2) specify names of student(s) you collaborated with on the PDF.

1 Kernels

Here we will look at an example of constructing feature spaces for classification problems. We will explore the idea of the "kernel trick" applied to classifiers other than SVM (i.e., classifiers using loss functions other than hinge loss). Specifically, we will introduce the kernel logistic regression (KLR). Recall that in class we have described the general logistic regression model as

$$\hat{p}\left(y=1 \,|\, \mathbf{x};\, \mathbf{w}, w_0\right) \;=\; \frac{1}{1+\exp\left(\sum_{j=1}^{d} w_j \phi_j(\mathbf{x})\right)},$$

where $\phi_j(\mathbf{x})$ is the j-th *basis function*, or *feature* - generally, a function mapping $\mathcal{X} \to \mathbb{R}$.

Problem 1 [20 points]
Show how by appropriate choice of basis functions $\boldsymbol{\phi}$, given a training set $\mathbf{x}_1, \ldots, \mathbf{x}_N$, one can obtain a logistic regression model whose predictions on a test point $\mathbf{x}_0$ depend on the training data only through the kernel values $K(\mathbf{x}_i, \mathbf{x}_0)$ for $i = 1, \ldots, N$. Then write down the gradient of the loss, with L_2 regularization, on a single example, and show that the training for this model via gradient descent also depends on the training data only through kernel computations.

End of problem 1

2 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:

$$\mathbf{\Sigma}_c = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \sigma_d^2 \end{bmatrix}. \tag{1}$$

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 $p(\mathbf{x} \mid y)$ and $p(y)$ to the training data, and use the resulting discriminant analysis to produce a decision rule which predicts

$$\widehat{y}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \left\{ p(\mathbf{x} \mid y = c)\, p(y = c) \right\}. \tag{2}$$

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

Problem 2 [15 points]
Show that the posterior $p(y = c \mid \mathbf{x})$ resulting from the generative model above has the same form as the posterior in logistic regression model,

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

for appropriate values of $w_0 \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d$.

End of problem 2

Advice: Start with Bayes rule, and use the simplifying assumptions in (1) *to derive the posterior.*

Problem 3 [10 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 $p(y \mid \mathbf{x})$. Will the two models produce the same classifier when applied to a given training set? Why or why not?

End of problem 3

2.1 Gaussian mixtures

In this section we will fit a Gaussian mixture model (GMM) to the Fashion MNIST data set familiar from Problem Set 2. To speed things up and reduce memory footprint of our code, we will reduce the dimension of the problem by resizing the images to 16 by 16 pixels, from the original 28 by 28. This is incorporated into the data loading function we have given you in the notebook. Note that while this does lead to some reduction in accuracy for many methods, the data are still possible to classify pretty accurately.

Our goal will be to use the GMM as a generative model for classifying the images into the ten Fashion MNIST classes

We have provided most of the code implementing the EM for GMM learning (as usual, with some missing pieces you need to fill in).

Problem 4 [30 points]
Fill in the missing pieces of code to make the EM algorithm work. These are found in the `Estep` and `Mstep` methods of the `GaussianMixtureModel` class.

Using the code, fit mixture of 3 Gaussians to each of the ten Fashion MNIST classes. Fill in the missing code in the `get_accuracy` function, which evaluates the accuracy of the GMM-based generative model on a data set, and use it to report training and validation accuracy

End of problem 4

2.2 Quadratic Discriminative Analysis

Now that you have the code for the EM, you should be able to use this code to fit a *single* Gaussian to the data.

Problem 5 [9 points]
For each class, fit a Gaussian (with full covariance matrix, separate per class) and report the classification accuracy of the resulting generative model on training and validation sets.

End of problem 5

Advice: You should be able to literally call the EM code, with the right parameters.

2.3 Naïve Bayes

In this problem you will implement a Naïve Bayes (NB) classifier to be applied for binary (sentiment) and multi-class (digit) classification.

We have not discussed Naïve Bayes classifiers in class; below is the brief overview of this approach.

A generative model, as we have seen in class, assigns the label according to a discriminant rule

$$\widehat{y}(\mathbf{x}) \;=\; \operatorname*{argmax}_{c} \left\{ \log p\left(\mathbf{x} \,|\, y=c\right) \,+\, \log p(y=c) \right\}. \tag{4}$$

The model of class-conditional densities $p\left(\mathbf{x} \,|\, y=c\right)$ for each class c is typicaly parametric and fit separately to examples from each class. The model for the priors $p(c)$ is usually simply based on counting – relative frequency of the classes in the training data. In the common scenario of balanced data where all C classes are represented with the same number of examples, the priors boil down to the uniform $p(c) = 1/C$ and so can be ignored since they don't contribute to the decision in (4).

A commonly used approach to regularization of such models is to limit, or penalize, the complexity of the class-conditionals, by imposing additional assumptions. One such limitation is to assume that given the class, individual features (dimensions of $\mathbf{x}$) are statistically independent. This is the Naïve Bayes model.

In its general form, the NB model for input examples $\mathbf{x}$ can be written as

$$\log p\left(\mathbf{x} \,|\, y=c\right) \;=\; \sum_{j=1}^{d} \log p\left(x_j \,|\, y=c\right), \tag{5}$$

where $p\left(x_j \,|\, y\right)$ specifies the class-conditional model for a one-dimensional (scalar) value of the j-th feature of $\mathbf{x}$ given the class.

It's called "naïve" since this conditional independence assumption is somewhat simplistic. However, it often works well in practice, both because of the regularizing effect of the independence assumption (think how many parameters does the general Gaussian model have vs. the NB Gaussian model) and because estimating class-conditional density is much easier computationally if it needs to be done per feature rather than jointly for many features.

Problem 6 [9 points]
Using the code created for the EM part of the problem set, train a Naïve

Bayes classifer for the Fashion MNIST data set, in which $p(x_j \mid y)$ is Gaussian. Report its training and validation performance.

End of problem 6

Advice: Just like for the previous problem, you literally should be able to call the code you have for the EM, with certain argument settings.

Problem 7 [**7 points**]
Now you can explore the models as you see fit: modify the number of components, restrictions on the covariance matrices, initialization strategy, stopping criteria, or any other aspect, as long as the resulting model is still generative model with one or more Gaussian per class. Using validation set to select what you expect to be the best model, submit its predictions on test to Kaggle,

End of problem 7