CSE 512 – Homework II

Dr. Ritwik Banerjee

Due by Apr 19 (Sunday), 2020 [11:59 pm]

1 Theory

I: Gaussian distributions **10 points**

A Gaussian distribution is defined by two parameters: a mean μ, and a variance σ^2. For a scalar x, it is defined as

$$\mathcal{N}\left(x|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{1}$$

Using this definition, we can find expectations of various functions of x. Prove the following

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) x dx = \mu \tag{2}$$

Also, this is a valid probability distribution, and therefore

$$\int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) dx = 1 \tag{3}$$

Differentiate the above w.r.t. σ^2 to show that

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) x^2 dx = \mu^2 + \sigma^2 \tag{4}$$

Using the above results, finally show that the variance of a Gaussian distribution, defined as $var(x) = \mathbb{E}[x^2] - \mathbb{E}[x]^2$, is indeed σ^2.

II: Strongly convex function **10 points**

Show that if f is a strongly convex function with paramere λ, then for every $\mathbf{w}$, $\mathbf{u}$, and $\mathbf{v} \in \partial f(\mathbf{w})$, the following holds:

$$\langle \mathbf{w}-\mathbf{u}, \mathbf{v}\rangle \geq f(\mathbf{w}) - f(\mathbf{u}) + \frac{\lambda}{2}\left\|\mathbf{w}-\mathbf{u}\right\|^2.$$

III: Kernel construction **10 points**

Suppose K_1 and K_2 are valid kernels on a domain $\mathcal{X}$, i.e., their Gram matrix is symmetric and positive-definite. Show that the following are also, then, valid kernel functions:

(a) $K(\mathbf{u},\mathbf{v}) = \alpha K_1(\mathbf{u},\mathbf{v}) + \beta K_2(\mathbf{u},\mathbf{v})$, for any scalars $\alpha, \beta \geq 0$.

(b) $K(\mathbf{u}, \mathbf{v}) = K_1(\mathbf{u}, \mathbf{v})K_2(\mathbf{u}, \mathbf{v})$.

IV: Local minimum **10 points**

Construct an example showing that the 0-1 loss function may suffer from local minima. That is, construct a training sample $S \in (\mathcal{X} \times \{\pm 1\})^m$ (for simplicitiy, you may assume that $X\mathcal{X} = \mathbb{R}^2$) for which there exist a vector $\mathbf{w}$ and some $\epsilon > 0$ such that

(1) For any $\mathbf{w}'$ such that $\|\mathbf{w} - \mathbf{w}'\| \leq \epsilon$, we have $L_S^{(01)}(\mathbf{w}) \leq L_S^{(01)}(\mathbf{w}')$, and

(2) There exists some $\mathbf{w}^*$ such that $L_S^{(01)}(\mathbf{w}^*) < L_S^{(01)}(\mathbf{w})$.

The first condition formally shows that $\mathbf{w}$ is a local minimum, while the second condition shows that it is not a global minimum.

V: Learnability of logistic regression **10 points**

Let $\mathcal{H} = \mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq B\}$ for some positive scalar B. Let the label set $\mathcal{Y} = \{\pm 1\}$, and the loss function be defined as $\ell(\mathbf{w}, (\mathbf{x}, y)) = \log[1 + \exp(-y\langle \mathbf{w}, \mathbf{x}\rangle)]$. Note that this is the formal definition of the logistic regression problem. Show that this problem is both convex-Lipschitz-bounded, and convex-smooth-bounded. Specify the parameters for Lipschitzness and smoothness.

VI: Learnability of Halfspaces with hinge loss **10 points**

Consider the bounded domain $\mathcal{X} = \{\mathbf{x} : \|\mathbf{w}\| \leq R\}$ with the label set $\mathcal{Y} = \{\pm 1\}$, and the loss function defined as $\ell(\mathbf{w}, (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}$. Note that this is the formal definition of halfspace learning with hinge loss. The formulation already covers boundedness. Show that this learning problem is also convex and R-Lipschitz.

2 Programming

In this section, you will first generate a small toy-dataset using the **Scikit-learn**[1] library in Python. Such a toy dataset can be quickly generates as follows:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets.samples_generator import make_blobs

X0, y = make_blobs(n_samples=100, n_features = 2, centers=2,
                   cluster_std=1.05, random_state=10)
X1 = np.c_[np.ones((X0.shape[0])), X0] # add one to the x-values to incorporate bias
```

You can see what your sample looks like, using the `plt.scatter()` and `plt.show()` functions. The next step is to create a data dictionary and see what the highest feature value is in your sample dataset. You can create the data dictionary are follows:

```
positive_x =[]
negative_x =[]
for i,label in enumerate(y):
    if label == 0:
        negative_x.append(X[i])
    else:
        positive_x.append(X[i])

data_dict = {-1:np.array(negative_x), 1:np.array(positive_x)}
```

[1] https://scikit-learn.org/stable/index.html

and then get the highest feature value

```
max_fval = float('-inf')

for y_i in data_dict:
    if np.amax(data_dict[y_i]) > max_fval:
        max_fval=np.amax(data_dict[y_i])
```

Your programming task for this assignment is to train a SVM classifier on your sample data. Note that this is fundamentally an optimization problem, and you should use the gradient descent algorithm to solve it (as discussed in lecture). It is easy to step across the minimum if you are not adjusting your step size properly. To resolve that, I would advise you to progressively decrease the step size using the highest feature value that you obtained earlier. Perhaps the easiest approach is to simply iterate over the data dictionary using smaller and smaller step sizes by maintaining an array of the form

```
step_sizes = [max_fval * 0.1, max_fval * 0.01, max_fval * 0.001, ... ]
```

I: Training SVM **25 points**

Write a function called `train(data_dict)`, and train your SVM using this function on the first 80% of your sample data. You do not need to shuffle the data at this stage, since the sample data generation code already performs shuffling.

II: Visualizing the data and the maximum-margin separator **10 points**

Write a function called `draw()`, which will show the sample data along with the maximum-margin separating hyperplane. An example of such a visualization is shown below:

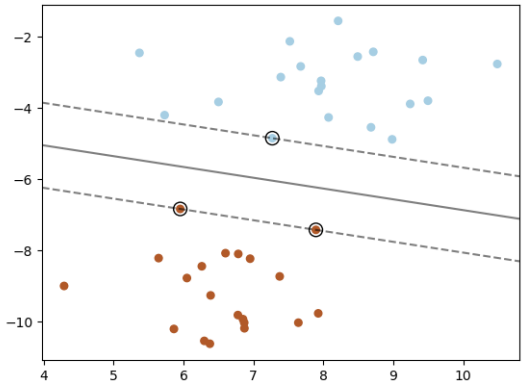

Figure 1: Visualization of a maximum-margin hyperplane (Source: Scikit-learn)

III: Testing the fit of the classifier **5 points**

Write another function called `test`, and test the remaining 20% of your sample data using the hyperplane obtained through your `train` method. When you have obtained the result, simply add the result in a `README.txt` file. The description should only include two things: the total number of data points on which your test method was run, and how many of those data points ended up being misclassified.

Notes

The programming language and library has been fixed for this homework. This was a conscious decision, mainly because any graduate student of machine learning, data science, etc. should have working knowledge of at least one or two of the most widely-used machine learning libraries.

What should we submit, and how?

All submissions must be through Blackboard.

Submit a single `.zip` archive containing (i) one folder simply called "code", containing all your code, (ii) a PDF document for the theory part of your submission. For the theory component, please do **NOT** handwrite the solutions and scan or take pictures.

Include the sample data generation code in your submission.

Use either LaTeXor MS Word to write the solutions, and export to PDF. Anything else will not be graded.