Introduction to Deep Learning M2177.0043 April 3, 2018

Due on April 10, 2017

Seoul National University

Homework 2

INSTRUCTIONS

- The homework is due at 9:00am on April 10, 2017. Anything that is received after that time will be considered to be late and we do not receive late homeworks. We do however ignore your lowest homework grade.
- Homeworks need to be submitted electronically on ETL. Only PDF generated from LaTex is accepted.
- Make sure you prepare the answers to each question separately. This helps us dispatch the problems to different graders.
- Collaboration on solving the homework is allowed. Discussions are encouraged but you should think about the problems on your own.
- If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution.

1 Eigenvalues [30 points]

Find all eigenvalues of the $X \in \mathbb{R}^{p \times p}$ real matrix filled with some constant value c except the diagonal component. Concretely, the matrix X is structured as following:

```
X = c * np.ones((p,p)) + (-c + 1)*np.eye(p)
```

2 Multivariate normal [35 points]

1. Describe the transformation which takes n independent $N(0,1)$ standard normal samples $Z = [z_1, \ldots, z_n]^\intercal$, and generates correlated random variables that follow a n-dimensional multivariate normal distribution $X = [X_1, \ldots, X_n]^\intercal \sim N(\mu, \Sigma)$.
2. Simulate $10,000$ draws of $N\left(0, \Sigma = \begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{bmatrix}\right)$ using std samples drawn from $N(0,1)$. Contrast the results with $10,000$ draws using Numpy's existing implementation for multivariate sampling `np.random.multivariate_normal` function. Visualize three scatter plots side-by-side (in 1×3 subplot format, use `markersize=0.1` for better visualization) for 1) the std normal, 2) your transformed multivariate samples, and 3) multivariate samples obtained from the Numpy code. Does your result closely match Numpy's results? Fix the random seed with `np.random.seed(1337)` in your code to make the experiment deterministic.

3 Learning a binary classifier with gradient descent [35 points]

We wish to learn a binary classifier $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$ with hinge loss. Concretely,

$$\text{minimize } \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^\intercal \mathbf{x}_i) + \frac{\lambda}{2} ||\mathbf{w}||^2$$

1. Derive the gradient of the loss function.
2. Use $\lambda = 0.1, n = 1000, d = 100$, and use fixed step size of 0.01. Also, use the following code to generate the data set $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\intercal \in \mathbb{R}^{n \times d}$, the label vector $\mathbf{y} = [y_1, \ldots, y_n] \in \mathbb{R}^n$, and the initial value $\mathbf{w}^{(0)} \in \mathbb{R}^d$.

Homework 2

```
X = np.vstack([np.random.normal(0.1, 1, (n//2, d)),
               np.random.normal(-0.1, 1, (n//2, d))])
y = np.hstack([np.ones(n//2), -1.*np.ones(n//2)])
w0 = np.random.normal(0, 1, d)
```

Then, solve the optimization problem of finding the optimal separating hyperplane $\mathbf{w}^*$ with gradient descent. Attach the a) source code, b) iteration vs function value plot, and c) iteration vs classification accuracy plot. Accuracy is defined by the fraction of data points your prediction $y_{\text{prediction}} = \text{sign}\left(\mathbf{w}^\intercal \mathbf{x}_i\right)$ matches the label y_i. Fix the random seed with `np.random.seed(1337)` in your code to make the experiment deterministic.