EECS 545 HW 4

Due Wednesday, Oct. 12, at 11 pm via gradescope

1. Logistic Regression as ERM (5 points)

This problems asks you to solve an exercise that was stated in the logistic regression notes.

Show that if $L(y,t) = \log(1+\exp(-yt))$, then

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, w^T x_i + b)$$

is proportional to the negative log likelihood for logistic regression. Therefore ERM with the logistic loss is equivalent to logistic regression, as was stated in class.

Note: In the above expression, y is assumed to be -1 or 1. In the logistic regression notes we had $y = 0$ or 1.

2. Subgradient methods for the optimal soft margin hyperplane (25 points)

In this problem you will implement the subgradient and stochastic subgradient methods for minimizing the convex but nondifferentiable function

$$J(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \mathbf{w}^T \boldsymbol{x}_i + b) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where $L(y,t) = \max\{0, 1-yt\}$ is the hinge loss. As we saw in class, this corresponds to the optimal soft margin hyperplane.

(a) (5 points) Determine $J_i(\mathbf{w}, b)$ such that

$$J(\mathbf{w}, b) = \sum_{i=1}^{n} J_i(\mathbf{w}, b).$$

Determine a subgradient $\mathbf{u}_i$ of each J_i with respect to the variable $\boldsymbol{\theta} = [b\ \mathbf{w}^T]^T$. A subgradient of J is then $\sum_i \mathbf{u}_i$.

Note: Recall that if $f(\mathbf{z}) = g(h(\mathbf{z}))$ where $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}$, and both g and h are differentiable, then

$$\nabla f(\mathbf{z}) = \nabla h(\mathbf{z}) \cdot g'(h(\mathbf{z})).$$

If g is convex and h is differentiable, the same formula gives a subgradient of f at $\mathbf{z}$, where $g'(h(\mathbf{z}))$ is replaced by a subgradient of g at $h(\mathbf{z})$.

Download the file `nuclear.mat` from Canvas. The variables `x` and `y` contain training data for a binary classification problem. The variables correspond to the total energy and tail energy of waveforms produced by a nuclear particle detector. The classes correspond to neutrons and gamma rays. Neutrons have a slightly larger tail energy for the same total energy relative to gamma rays, which allows the two particle types to be distinguished. This is a somewhat large data set ($n = 20,000$), and subgradient methods are appropriate given their scalability.

(b) (6 points) Implement the subgradient method for minimizing J and apply it to the nuclear data. Submit two figures: One showing the data and the learned line, the other showing J as a function of iteration number. Also report the estimated hyperplane parameters and the minimum achieved value of the objective function.

Comments:

- Please execute the line

  ```
  rng(0); \% in Matlab
  ```

  or

  ```
  random.seed(0) \# in Python
  ```

  at the beginning of your code to seed the random number generator.
- Use $\lambda = 0.001$. Since this is a linear problem in a low dimension, we don't need much regularization.
- Use a step-size of $\alpha_j = 100/j$, where j is the iteration number.
- To compute the subgradient of J, write a subroutine to find the subgradient of J_i, and then sum those results.
- Since the objective will not be monotone decreasing, determining a good stopping rule can be tricky. Just look at the graph of the objective function and "eyeball it" to decide when the algorithm has converged.
- Debugging goes faster if you just look at a subsample of the data. To debug in Matlab, the command `keyboard` is very helpful. Just type `help keyboard` and also look up related commands. For something analagous in Python, try `http://stackoverflow.com/questions/13432717/enter-interactive-mode-in-python`.

(c) (6 points) Now implement the stochastic subgradient method, which is like the subgradient method, except that your step direction is a subgradient of a random J_i, not J. Be sure to cycle through all data points before starting a new loop through the data. Report/hand in the same items as for part **(b)**.

More comments:

- Use the same λ, stopping strategy, and α_j as in part **(b)**. Here j indexes the number of times you have cycled (randomly) through the data.
- Your plot of J versus iteration number will have roughly n times as many points as in part **(b)** since you have n updates for every one update of the full subgradient method.
- To generate a random permutation use

  ```
  randperm
  ```

  in Matlab, or

  ```
  import numpy as np
  np.random.permutation
  ```

  in Python.
- Please reseed the random number generator as decribed previously.

(d) (3 points) Comment on the (empirical) rate of convergence of the stochastic subgradient method relative to the subgradient method. Explain your finding.

(e) (5 points) Submit your code.

3. Kernels (10 points)

(a) (3 points) To what feature map Φ does the kernel

$$k(u, v) = (\langle u, v \rangle + 1)^3$$

correspond? Assume the inputs have an arbitrary dimension d and the inner product is the dot product.

(b) (3 points) Show that if k_1 and k_2 are inner product kernels and $a_1, a_2 \geq 0$, then $a_1 k_1 + a_2 k_2$ is an inner product kernel.

(c) (4 points) Prove one direction of an equivalence stated in the notes, namely, that if k is an inner product kernel, then k is a symmetric, positive definite kernel.

SUPPLEMENTAL EXERCISES

Not to be turned in.

1. The Gaussian Kernel is a Kernel

In this problem you are asked to show that the Gaussian kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = C \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right), \qquad C > 0, \sigma > 0,$$

is an inner product kernel (equivalently, symmetric and positive definite kernel). You will be asked to verify various properties that let you construct kernels from other kernels, and you will see that both characterizations of kernels (IP/SPD) are useful in this regard.

(a) Consider a sequence of symmetric, positive definite (SPD) kernels $\{k_n\}$ that coverges pointwise to a function k, i.e., for all $\boldsymbol{x}, \boldsymbol{x}'$, $\lim_{n\to\infty} k_n(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}')$. Show that k is also a SPD kernel.

(b) Prove that if $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and $k_2(\boldsymbol{x}, \boldsymbol{x}')$ are SPD kernels, then so is $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}')$.
Hint: Consider zero mean, independent random vectors V_1 and V_2 whose covariances are kernel matrices K_1 and K_2. Determine another random vector whose covariance is the element-wise product of K_1 and K_2.

(c) Let $f(t) = \sum_{n=0}^{\infty} a_n t^n$ be a power series that converges for all $t \in \mathbb{R}$, and for which all $a_n \geq 0$. Argue that $k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{n=0}^{\infty} a_n (k(\boldsymbol{x}, \boldsymbol{x}'))^n$ is a kernel.

(d) Deduce that the exponential kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(\gamma \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)$ is an inner product kernel, where $\gamma > 0$ and $\langle \cdot, \cdot \rangle$ denotes the dot product on $\mathbb{R}^d$.

(e) Show that if k is an inner product kernel, then so is the normalized kernel

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} 0, & \text{if } k(\boldsymbol{x}, \boldsymbol{x}) = 0 \text{ or } k(\boldsymbol{x}', \boldsymbol{x}') = 0 \\ \frac{k(\boldsymbol{x}, \boldsymbol{x}')}{k(\boldsymbol{x}, \boldsymbol{x}) k(\boldsymbol{x}', \boldsymbol{x}')}, & \text{otherwise.} \end{cases}$$

(f) Deduce that the Gaussian kernel is an inner product kernel.