Introduction to Machine Learning (67577)

Exercise 6
PCA, kernels, SGD and DL

Second Semester, 2021

Contents

Submission

1. You should sumbit one file named **ex_6_FirstName_LastName.tar** containing the following files:
   - **Ex6_Answers.pdf** - a pdf file which contains all your answers to both theoretical parts and practical parts.
   - **network.py**

1 PCA

1. Let X be a random variable with some distribution over $\mathbb{R}^d$ so that its mean is $(0,\ldots,0) \in \mathbb{R}^d$ and its covariance matrix is $\Sigma \in \mathbb{R}^{d\times d}$. Show that for any $v \in \mathbb{R}^d$, where $\|v\|_2 = 1$, the variance of $\langle v, X\rangle$ is not larger then variance of the PCA embedding of X into 1- dimension (Assume that the PCA uses the actual Σ).

2 Kernels

2. Given some valid kernel $k(\mathbf{x},\mathbf{x}')$, provide a formula for a different valid kernel $\tilde{k}$ which is guaranteed to be normalized, in the sense that for all $\mathbf{x}$ we have $\tilde{k}(\mathbf{x},\mathbf{x}) = 1$.
3. Consider a data set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ and a feature map $\psi : \mathbb{R}^d \to \mathcal{F}$ where $\mathcal{F}$ is some feature space. Give an example of a data set S and a feature map ψ such that S is not linearly separable in $\mathbb{R}^d$ (for $d \geq 2$) but that the transformed data set $S_\psi = \{(\psi(\mathbf{x}_i), y_i)\}_{i=1}^m$ is linearly separable in $\mathcal{F}$.

3 Convex optimization

4. (a) Let $f_i : V \to \mathbb{R}$ for $i = 1,\ldots,m$ be functions and let $\gamma_1,\ldots,\gamma_m$ be non-negative scalars. Consider the function $g : V \to \mathbb{R}$ given by

$$g(u) = \sum_{i=1}^{m} \gamma_i f_i(u).$$

   Prove from the definition of a convex function that if $f_1,\ldots,f_m$ are convex, then g is also convex.

   (b) Give a counterexample for the following claim: Given two functions $f, g : \mathbb{R} \to \mathbb{R}$, define a new function $h : \mathbb{R} \to \mathbb{R}$ by $h = f \circ g$. If f and g are convex than h is convex as well.

   (c) **Prove:** A function $f : C \to \mathbb{R}$ defined over a convex set C is convex iff its *epigraph* is a convex set, where $\text{epi(f)} = \{(u,t) : f(u) \leq t\}$.

   (d) Let $f_i : V \to \mathbb{R}$, $i \in I$. Let $f : V \to \mathbb{R}$ given by

$$f(u) = \sup_{i\in I} f_i(u).$$

   If f_i are convex for every $i \in I$, then f is also convex.

5. (a) Given $x \in \mathbb{R}^d$ and $y \in \{\pm 1\}$. Show that the hinge loss is convex in w, b. Meaning, define

$$f(w) = l_{x,y}^{hinge}(w,b) = \max\left(0, 1 - y(w^T x + b)\right)$$

and show that f is convex in w, b.

(b) Deduce some $g \in \partial l_{x,y}^{hinge}(w,b)$.

(c) Given $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$ convex functions, and $\xi_k \in \partial f_k(x)$ for all k. Define $f : \mathbb{R}^d \to \mathbb{R}$ by $f(x) = \sum_{i=1}^m f_i(x)$. Show that $\sum_k \xi_k \in \partial \sum_k f_k(x)$.

(d) Given dataset $\{(x_i, y_i)\}_{i=1}^m \subset \mathbb{R}^d \times \{\pm 1\}$. Define the function $f : \mathbb{R}^d \to \mathbb{R}$ by:

$$f(w,b) = \frac{1}{m} \sum_{i=1}^{m} l_{x_i,y_i}^{hinge}(w,b) + \frac{\lambda}{2} \|w\|^2$$

Find a member of the sub-gradient of f for each w.

4 Multi Layer Perceptron (MLP) for for digit classification (MNIST)

6. In this question you will implement from scratch a feed-forward neural network, aka Multi Layer Perceptron (MLP), for classifying digits from the MNIST dataset. You are given the file `mnist.pkl.gz` which contains the data, the file `mnist_loader.py` which loads this data and the file `network.py` which contains a code skeleton of the `Network` class and its methods. Your job is to fill in the missing lines of code (where ever you see `TODO` in the code). All the method signatures and their outputs are specified in the code. You should go over what we have learned in the Tirgul about the backpropagation algorithm.
7. After all that coding you are now ready to train your network.
   - Load the data using:
     `training_data, validation_data, test_data = mnist_loader.load_data_wrapper()`
   - Create an instance of the network with:
     `net = network.Network([784, ..., 10])`
     where the `...` stand for the number of neurons in each of the hidden layers (one such number for each hidden layer).
   - Finally, if you want to use stochastic gradient descent to learn from the MNIST `training_data` over 30 epochs, with a mini-batch size of 10, and a learning rate of $\eta = 3.0$ use:
     `net.SGD(training_data, 30, 10, 3.0, test_data=test_data)`

   (a) You are free to experiment with different values of the learning rate η, the batch size, the number of layers and how many neurons in each layer. Scan some range of these parameters and plot the test accuracy vs. the number of epochs. For instance, fix everything except the batch size and report on the same plot several curves showing the accuracy vs epochs, where each curve is in a different color.

   (b) Try to play around some more with these settings to reach the highest accuracy possible, with a fixed number of epochs, say no more then 100, and preferably less.
   What are the optimal architecture (number of layers and number of neurons in each layer), batch size and learning rate that you found? which of these had the most dramatic effect? Demonstrate your results with appropriate plots.