236606 - Deep Learning - Homework 1

Spring 2018

Submission Instruction

Submission deadline: **24/04/2018 23:59**
TA in charge: Izik Golan

- Submit a **zip** file containing **only** your code and the written answers, in the webcourse site: `http://webcourse.cs.technion.ac.il/236606/`
- Answers must be submitted as a PDF file, edited by a document editor (**not** a scan of handwritten answers)
- Submission is **pairs only**
- There will be **no** late submissions

1 Distance of Point to Hyperplane

In tutorial 1, we calculated the distance of a point from a hyperplane using geometry. The same result can be achieved with constrained optimization.

A. Given a hyperplane, $H_{\boldsymbol{w},b} = \left\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{w}^\top \boldsymbol{x} + b = 0\right\}$, and a vector $\boldsymbol{x}_0 \in \mathbb{R}^n$, write the constrained optimization problem whose solution is the Euclidean distance of $\boldsymbol{x}_0$ to $H_{\boldsymbol{w},b}$. Use the following standard form for your optimization problem:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f\left(\boldsymbol{x}, \boldsymbol{x}_0, \boldsymbol{w}, b\right) \\ \text{subject to} \quad & g\left(\boldsymbol{x}, \boldsymbol{x}_0, \boldsymbol{w}, b\right) = 0 \end{aligned}$$

B. Using the method of Lagrange multipliers, solve the optimization problem from 1(A) and show that the solution is identical to the one we showed in Tutorial 1. You can read about the method of Lagrange multipliers at `https://en.wikipedia.org/wiki/Lagrange_multiplier`.

2 Perceptron

In Lecture 2, we proved that the perceptron algorithm makes at most $1/\left(\gamma\right)^2$ mistakes on any sequence of labeled examples that is contained inside the unit hyper-sphere, and linearly-separable by a margin γ. The algorithm does **not** guarantee anything about the margin of the final classifier.

Assume that $\mathcal{X}$ is in the unit hyper-sphere. Let $S \subseteq \mathbb{R}^d \times \{-1, +1\}$ be a sequence of labeled examples. Consider the following variation of the vanilla perceptron algorithm, where the only difference is that we now update $\boldsymbol{w}_{t+1}$ also for margin error (examples that are closer than $\gamma/2$ to the decision boundary), and not just prediction mistakes:

Margin Perceptron algorithm

procedure MARGINPERCEPTRON(S, γ)
  $\boldsymbol{w}_1 \leftarrow \vec{\mathbf{0}}$
  for $t = 1, \ldots, T$ **do**
    $\boldsymbol{x}_t, y_t \leftarrow next\,(S)$
    if $\frac{\boldsymbol{w}_t \cdot \boldsymbol{x}_t}{\|\boldsymbol{w}_t\|} \in \left(-\frac{\gamma}{2}, +\frac{\gamma}{2}\right)$ **or** $y_t\,(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \leq 0$ **then**
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + y_t \boldsymbol{x}_t$
    else
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t$
    end if
  end for
  return $\boldsymbol{w}_{T+1}$
end procedure

In Lecture 2, we proved that if there exists a unit-vector $\boldsymbol{w}^* \in \mathbb{R}^d$ such that $\forall\,(\boldsymbol{x}, y) \in S, y\,(\boldsymbol{w}^* \cdot \boldsymbol{x}) > 0$, then the number of mistakes M on S made by the online Perceptron algorithm is at most $1/\,(\gamma)^2$, where

$$\gamma = \min_{\boldsymbol{x} \in \mathcal{S}} |\boldsymbol{w}^* \cdot \boldsymbol{x}|$$

A. Prove that if $\boldsymbol{w}^*$ is not a unit vector, we can normalize both the sequence of examples, S, and $\boldsymbol{w}^*$ to be on the unit hyper-sphere, without changing the predictions made by $\boldsymbol{w}^*$ on each sample .

B. Prove that if there exists a linear separator with margin γ for S, then the margin perceptron algorithm will make at most $\frac{8}{\gamma^2}$ mistakes on S. **Hint:** prove that $\|\boldsymbol{w}_{t+1}\| \leq \|\boldsymbol{w}_t\| + \frac{1}{2\|\boldsymbol{w}_t\|} + \frac{\gamma}{2}$, and see what happens when $\|\boldsymbol{w}_t\| \geq \frac{2}{\gamma}$

3 Ridge Regression

Consider the data set $S_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ where $\boldsymbol{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}$. Also, assume the samples are centered, i.e., $\forall j = 1, \ldots, n : \frac{1}{m}\sum_{i=1}^m (\boldsymbol{x}_i)_j = 0$. Ridge regression is a variant of linear regression, where a regularization term is added to the loss function. This is also called Tichonov regularization. The minimization objective (loss) for an ERM training strategy is

$$L\,(\boldsymbol{w}, b) = \frac{1}{2m}\|\boldsymbol{X}\boldsymbol{w} + b\mathbf{1} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2,$$

where $\boldsymbol{X}$ is the data matrix with row i being $\boldsymbol{x}_i$, $\boldsymbol{y}$ being a vector where $\boldsymbol{y}_i = y_i$, and $\mathbf{1}$ is a vector of 1s of size m.

A. Compute $\frac{\partial L}{\partial b}$. Simplify as much as possible. **Hint:** This should not be a function of $\boldsymbol{w}$ or $\boldsymbol{X}$

B. Compute $\frac{\partial L}{\partial \boldsymbol{w}}$. Simplify as much as possible. Write your answer in vector form.

C. Suppose that $\lambda > 0$. Infer the solution obtained by equating the derivatives from 3(A) and 3(B) to 0. Are these solutions a global optimum for L?

D. In 3(C), you need to take an inverse of some matrix $\boldsymbol{M}$ to get the solution for $\boldsymbol{w}$. It is well known that a matrix $\boldsymbol{M}$ is invertible if and only if it is *positive definite*, meaning that $\forall \boldsymbol{v} \neq 0 \in \mathbb{R}^n : \boldsymbol{v}^\top \boldsymbol{M}\boldsymbol{v} > 0$. Prove that if $\lambda > 0$ than $\boldsymbol{M}$ is invertible. You can read more about positive definite and semi-definite matrices at:
`https://en.wikipedia.org/wiki/Positive-definite_matrix`.

E. What advantage of ridge regression over vanilla linear regression does 3(D) highlight?

4 Binary Classification Using Ridge Regression

Use the following script to obtain the MNIST dataset:

```
import pickle, gzip, numpy, urllib.request, json

data_url = "http://deeplearning.net/data/mnist/mnist.pkl.gz"
# Load the dataset
urllib.request.urlretrieve(data_url, "mnist.pkl.gz")
with gzip.open('mnist.pkl.gz', 'rb') as f:
    train_set, valid_set, test_set = pickle.load(f, encoding='latin1')
```

The MNIST dataset contains grayscale images of handwritten digits (0-9). Each image is 28x28 pixels, each of which with a label corresponding to the written digit in it. For the purpose of linear regression, we flatten each image to be a vector of length 784. Since we are dealing with binary classification, consider all even digits to have label $+1$, and all odd digits to have label -1.

Throughout this exercise, make sure to take advantage of NumPy vectorizations as much as you can.

A. Pre-process the dataset by subtracting from each sample (including samples from the validation and test sets) the vector which is the mean of all vectors in the training set.

B. Build a linear regressor by minimizing the objective from question 3 w.r.t the training set. Implement both the analytic solution and a gradient descent solution with 100 gradient steps and learning rate 0.001. Train the regressor on the training data using 8 values of λ taken from a logarithmic scale, $\left[10^{-5}, 10^{2}\right]$ (i.e., take powers of 10).

C. For the purpose of classification, the classifier which is based on the linear regressor is given by $h_{\boldsymbol{w},b}(\boldsymbol{x}) = \text{sign}(\boldsymbol{x} \cdot \boldsymbol{w} + b)$. We define the 0-1 loss and squared loss on a set of m labeled examples as:

$$L_{0-1}(\boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{I}\left(h_{\boldsymbol{w},b}(\boldsymbol{x}_i) \neq y_i\right)$$

$$L_{squared}(\boldsymbol{w}, b) = \frac{1}{m} \|\boldsymbol{X} \cdot \boldsymbol{w} + b\boldsymbol{1} - \boldsymbol{y}\|_2^2$$

For each value of λ, and each training scheme (gradient descent and analytic), What is the 0-1 loss of the training and validation sets? What is the squared loss?

D. For each training scheme, write the 0-1 loss and squared loss on the test set, of the model which performed best on the validation set in terms of 0-1 loss.

E. For the value of λ you found to perform best on the validation set in terms of 0-1 loss using the gradient descent scheme, plot the learning curves consisting of the train and test losses which are defined in 4(C), as a function of the number of gradient steps.

F. Describe the simplest possible scenario you can think of for which the use of linear regression for classification is a bad idea?

Good Luck!