TTIC 31020: Introduction to Machine Learning
Autumn 2019

Problem Set #1 (85 Points)

Out: Tuesday, October 8

Due: Friday, October 18, 8:00pm

Instructions

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want, e.g., LaTeX (recommended), Word + export as PDF, scan handwritten solutions (note: must be legible!), etc. Please name this document ⟨`firstname-lastname`⟩`-sol1.pdf`.

2. The empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file. Rename the notebook ⟨`firstname-lastname`⟩`-sol1.ipynb`.

Late submissions: there will be a penalty of 20 points for any solution submitted within 24 hours past the deadline. No submissions will be accepted after then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include in the `ipynb` file the figure as well as the code used to plot it. If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers) and references in a legend or in a caption. When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important. If there is a mathematical answer, provide it precisely (and accompany by succinct wording, if appropriate).

When submitting code (in Jupyter notebook), please make sure it's reasonably documented, runs and produces all the requested results. If discussion is required/warranted, you can include it either directly in the notebook (you may want to use markdown style for that) or in the PDF writeup.

Collaboration policy: collaboration is allowed and encouraged, as long as you (1) write your own solution entirely on your own, (2) specify names of student(s) you collaborated with in your writeup.

1 Linear regression

In the next two problems we will consider the effect that transforming the training data has on the optimal regression model. Suppose we fit a least squares linear model to a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$, yielding the parameter vector $\mathbf{w}^*$.

Problem 1 **[10 points]**
Suppose we take the dataset from which we learned (estimated) $\mathbf{w}^*$, and for every example $(\mathbf{x}_i, y_i)$, change y_i to $y_i' = ay_i + b$, for some constants a and b (same a and b for all i). Now suppose you fit the least squares model to the new dataset $\{(\mathbf{x}_i, y_i')\}_{i=1}^n$, yielding the parameter vector $\mathbf{w}'$.

Can $\mathbf{w}'$ be computed directly from $\mathbf{w}^*$, without looking at the data? If yes, how exactly? If not, why not?

End of problem 1

Problem 2 **[10 points]**
Now, instead of modifying ys, we will modify $\mathbf{x}$s. For every feature (dimension of $\mathbf{x}$) we will change x_{ij} to $\tilde{x}_{ij} = c_j x_{ij}$ for some constant c_j (same set of $c_1, \ldots, c_d$ for all i). Again, we fit a least squares model to the new dataset $\{(\tilde{\mathbf{x}}_i, y_i)\}_{i=1}^n$, and get $\tilde{\mathbf{w}}$.

Can $\tilde{\mathbf{w}}$ be computed directly from $\mathbf{w}^*$ without looking at the data? If yes, how exactly? If not, why not?

End of problem 2

Now we are going to look at a noise model which is a bit different from the i.i.d. Gaussian noise model described in class. Suppose that for every $\mathbf{x}$, the noise that affects y is Gaussian, but the variance of this noise depends on $\mathbf{x}$:

$$y, = \mathbf{w} \cdot \mathbf{x} + \nu, \qquad \nu \sim \mathcal{N}\left(0, \sigma_{\mathbf{x}}^2\right) \tag{1}$$

Problem 3 **[10 points]**
Without knowing anything else besides the assumptions in (1), can we compute the maximum likelihood estimate for the linear regression parameters $\mathbf{w}^*$ from a given dataset under this noise model? If yes, describe the procedure as precisely as you can; if not, explain why not.

End of problem 3

Problem 4 **[10 points]**
Now suppose we *know* the value of the noise variance $\sigma^2_{\mathbf{x}_i}$ at every training input $\mathbf{x}_i$ for $i = 1, \ldots, n$. With this additional assumption, can we compute the maximum likelihood estimate for linear regression parameters $\mathbf{w}^*$ from a given dataset? If yes, describe the procedure as precisely as you can; if not, explain why not.

End of problem 4

2 Loss functions

In this section we will focus on hands-on exploration of the interaction between loss functions used in regression and the models produced. We will work with the Boston Housing dataset[1] in which the task is to predict median home prices in various areas of the Boston suburbs (in the 1990s) from a variety of features.

We are going to assume that in our task, it is much worse to over-estimate the price of a house than to under-estimate it. (Perhaps we really want to be able to sell houses quickly...) Technically, we will express it as an **asymmetric squared loss**:

$$\ell_\alpha\left(\widehat{y}, y\right) = \begin{cases} \alpha\left(\widehat{y} - y\right)^2 & \text{if} \quad \widehat{y} < y \quad \text{(under-estimate)} \\ \left(\widehat{y} - y\right)^2 & \text{if} \quad \widehat{y} \geq y \quad \text{(over-estimate)} \end{cases} \tag{2}$$

where $\alpha < 1$ specifies how much more we worry about $\widehat{y}$ over- rather than under-estimating the house price y. For example, if $\alpha = 0.1$ then over-estimating is 10 times worse than under-estimating by the same amount.

We will now consider fitting polynomial regression models to predict house prices y from observations/measures x. The Jupyter notebook provided with this assignment contains most of the code you need to conduct experiments in this section; you will need to fill in some missing pieces of the code, however.

Problem 5 **[15 points]**
Complete the missing code in the gradient descent function `GD`. Experimenting with models and optimization parameters, fit polynomial models of degrees 1 (linear), 2, and some others (feel free to explore) to the training set.

Use the validation set to select one of the models; explain your selection process in the notebook (use a markdown cell) or in the PDF writeup.

End of problem 5

Advice: Use the code for the closed form solution provided earlier in the notebook to debug your implementation of gradient descent; you should expect very similar (perhaps not quite identical) results when fitting, say, a linear model to the training data using the closed form solution and with GD.

[1] `https://archive.ics.uci.edu/ml/machine-learning-databases/housing/`

Now we will move to the asymmetric loss. The optimization problem involving the asymmetric loss is convex, but unfortunately does not have a closed form solution. So the gradient descent code we have will come in handy.

Problem 6 [15 points]
Complete the code for the function computing the asymmetric loss function ℓ_α (and its gradient).

Using ℓ_α with $\alpha = 0.05$, fit polynomial models (of degrees 1, 2, and any others you wish to try) to the training set. Similarly to the symmetric loss case, select one model using the validation set and explain your selection process.

End of problem 6

Problem 7 [15 points]
Finally, evaluate your two chosen models (one trained with symmetric loss and the other trained with asymmetric loss) on the test set. For each model compute: the mean symmetric loss and the mean asymmetric loss (using squared roots of both for comparison is fine). Based on these results, discuss the relative merits of the two models for the data and task at hand. If you had to choose one, given the description of the prediction problem and the assumptions above, which one would you choose, and why?

End of problem 7