Problem Set 2

Introduction to Machine Learning

October 14, 2024

Problem 0: Summary, tips and instructions

Congratulations on completing the first part of Introduction to Machine Learning! We started with a brief overview of the applications, looked at various types of data and then delved into the fascinating realm of fitting a line to a bunch of points. Alright, it's not that fascinating, but it's certainly very powerful. You learned the steps it takes to go from data to a model and back:

1. Collect data
2. Split it into training, validation, and test sets (keep the test set on the side)
3. Visualize the training and validation sets
4. Pre-process the data. Before learning a model, you might want to look at it correlations, see if there are missing values, clean it, denoise etc. This is mostly an art and is a function of the problem you're solving. Once you do that, depending on the domain of interest, you can get rid of some features x_i or append functions of the existing ones, such as squaring them or applying other nonlinear transformations to them; e.g. x_1^2 and $x_1 x_2$, where x_i is the i-th entry of the input features $x \in \mathbb{R}^n$. This is called **feature engineering** and is also an art. We introduced it in class as a function: $\phi(x)$, where x can be a vector or a scalar.

   Here's a good place to mention notation. You'll see in the CS229 notes that there's no mention of $\phi()$ until later chapters. The assumption is that whatever you do with the input x (square it, define a polynomial out of it, etc.) you can define as your new x (or new features). The notation $\phi(x)$ only makes it more explicit; and in some cases it complicates. So whenever you see a variable x, you can assume that it can be feature engineered into a $\phi(x)$; e.g. below, $h_\theta(x) = \theta^\top \phi(x)$.

5. Propose a hypothesis:

   (a) For a multidimensional input $x = [x_1, x_2, x_3, \ldots, x_n] \in \mathbb{R}^n$, and a real output $y \in \mathbb{R}$, we proposed the linear hypothesis:

   $$h_\theta(x) = \theta^\top x = \theta_1 x_1 + \theta_2 x_2 + \ldots \tag{1}$$

   where $\theta \in \mathbb{R}^n$. When you have more than one input, this is sometime called multiple linear regression. If you have more than one output (e.g. $y = [y_1, y_2, \ldots]$, you can simply run linear regression on each of the outputs separately: $y_i = \theta^\top x$. But open source packages automate the process of putting these together, so you don't have to worry about it.

   (b) For a multidimensional input $x = [x_1, x_2, x_3, \ldots, x_n] \in \mathbb{R}^n$ and a binary output, we proposed a logistic function of a linear hypothesis: $h_\theta(x) = \sigma(\theta^\top x)$. And for multi-dimensional outputs, we saw that we need to use a slightly different hypothesis (Softmax) to predict the probability of a label k rather than a single probability.

6. Define a loss (or cost) function as a function of the fitting parameters θ. The loss is typically a distance measure between the true output $y^{(i)}$ and the predicted output $h_\theta(x^{(i)})$ given its corresponding input $x^{(i)}$. The total loss is given by:

   $$J(\theta) = \sum_{i=1}^{n} \text{Distance}\left(h_\theta(x^{(i)}), y^{(i)}\right) \tag{2}$$

We saw the square loss for linear regression and the binary cross-entropy loss for logistic regression. In either case, the total loss $J(\theta)$ is defined in terms of the fitting parameters only, and it determines what we really care to minimize (no one likes to lose, but we have to define first what kind of loss we want to minimize).

7. When the loss is defined, now you have to minimize it:

$$\hat{\theta} = \text{argmin}_\theta J_{\text{train}}(\theta) \tag{3}$$

   If you haven't seen it, 'argmin' returns the minimum argument (variable) over which the optimization is being done. In linear regression with a square loss, we saw that it can be done analytically, but we didn't go through the derivation. You'll get to go through the derivation yourself in problem 2 and you'll see it in the CS229 notes. Try it yourself before looking at the solution. For the most part, the matrix algebra involved is very similar to the scalar algebra you're used to; you just have to be careful with matrix multiplication being consistent.

8. Having found the optimal fitting parameters, you replace it in the hypothesis and you test it on the validation set; evaluating $J_{\text{val}}(\hat{\theta})$. If you're not satisfied, you go back to step 4, propose another hypothesis and repeat the process.

9. Once done with iterations, you test your model on the test set, and deploy it.

This is the process in a nutshell; but a lot is involved in hyperparameter tuning, and the process might be highly nonlinear, going back and forth and sometimes having a rough idea of what you expect.

Before you start the problem set, I want to give you a few tips on how to study machine learning (and maybe most other topics). First, you've probably noticed that machine learning is a combination of two things: math and coding. And you can't learn coding or math by just reading, and even less by listening to me go through complex concepts in 50 minutes: you have to practice, practice, practice. You have to learn - like machine learning - from lots of data; otherwise, you'll overfit to the few problems you were told will show up on the exam and think that you actually understand the topic. It's an accurate analogy. Of course, this course is just the beginning of your journey, so don't expect to know everything by the end either. My purpose is to inspire you and motivate you to understand the inner workings of the topics we'll be covering in this class so you're excited to explore more when you're done with this course.

In light of this goal, the purpose of the problem set below is simply an opportunity for you to test your knowledge on simplified problems before you face much more complex ones. So it's more for you to learn the basics than for me to judge you. With LLMs and tons of online references, it's probably easy to find solutions to these, but I'm trusting you to try, as much as you can, to solve the problems on your own. Since I might not be able to truly test your understanding through homework assignments, I have to trust you with that task. This doesn't mean that you can't discuss the problems with others; I encourage you to do that. But **when you write down the solution, do it on your own. Copied solutions will be penalized**.

For problems 1, 2, and 3, submit a written/typed solution in class at the beginning of the lecture on Monday September 23. For problem 4, you should submit a notebook on Moodle. And for problem 5 - the only problem you can do with others - your team should submit your predictions, along with the notebook, on Kaggle. Finally, if you find mistakes/typos in the exercises below, let me know. Have fun!

Problem 1: True or False (15 points)

Determine if the following statements are True or False and provide a brief explanation:

1. If the linear predictor is overfitting on the training set, the training set error is smaller than the test set error.
2. The purpose of cross-validation is to prevent overfitting on the test set.
3. A development set (a.k.a. dev-set, a.k.a. validation set) is used to prevent overfitting on the test set.
4. Increasing the amount of data will prevent the model from overfitting.
5. Adding an L_1 regularization term will decrease the likelihood of underfitting.
6. Stochastic gradient descent requires less updates on the fitting parameters θ to converge to the optimal solution.
7. In linear regression, the least squares method minimizes the sum of the squared differences between the predicted and actual values of the outputs.
8. A coefficient θ_i in the linear hypothesis $h_\theta(x) = \theta^\top x$ can be interpreted as the change in the predicted value $h_\theta(x)$ of the output for a one-unit increase in the corresponding input feature x_i.
9. In logistic regression, the output is a continuous variable that represents the probability of the binary outcome.
10. For real number inputs and outputs, the absolute loss penalizes outliers more than the square loss (it does worse at capturing the trend).
11. In logistic regression, using a threshold of 0.5 on the predicted probability will always give the optimal classification accuracy.
12. Logistic regression is a special case of a generalized linear model (GLM) with binomial distribution for a response variable.
13. In linear regression, adding more independent variables to the model always improves the model's predictive power.
14. Given an input x and an output y, linear regression always constrains the hypothesis to be linear in x.
15. A Generalized Linear Model (GLM) allows for response variables that have distributions other than normal, such as binomial or Poisson.

Problem 2: Derive the Normal Form Equation (10 points)

Given the least mean square expression in matrix form: $\mathcal{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$, derive the normal equation which expresses the optimal weights (parameters) $\mathbf{w}$ in terms of $\mathbf{X}$ and $\mathbf{y}$. You might find the following linear algebra properties useful:

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$
$$\nabla_A tr(ABA^TC) = CAB + C^TAB^T$$
$$\nabla_{A^T} tr(ABA^TC) = BA^TC + B^TA^TC^T$$

Problem 3: Logistic Regression (15 points)

Given a dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ with $x^{(i)} \in \mathbb{R}$ and $y^{(i)} \in [0, 1]$, we would like to fit a logistic classifier with fitting parameters $\mathbf{w}$, and predictor

$$f_{\mathbf{w}}(x) = g(\phi(x) \cdot \mathbf{w}) = \frac{1}{1 + e^{-\phi(x) \cdot \mathbf{w}}} \tag{4}$$

where $\phi()$ is a feature map.

1. Find the derivative $g'(z) = dg/dz$ as a function of $g(z)$. Here $z \equiv \phi(x) \cdot \mathbf{w}$.
2. Find the log-likelihood $l(\mathbf{w})$ from the likelihood $p(y^{(i)}|x^{(i)}; \mathbf{w})$ in terms of $\mathbf{w}$, $x^{(i)}$, and $y^{(i)}$.
3. Derive the equation for the gradient of the log-likelihood $\nabla_{\mathbf{w}} l(\mathbf{w})$ (you can use vector identities or get the derivative with respect to one parameter (w_j) at a time, i.e. $\partial l(\mathbf{w})/\partial w_j$, where w_j is the j^{th} element of the vector $\mathbf{w}$.
4. What is the least mean squares (LMS) update rule to maximize the log-likelihood?

Problem 4: Classification with Scikit-Learn (30 points)

I have provided two files `p2_x.txt` and `p2_y.txt`. These files contain inputs $x^{(i)} \in \mathbb{R}^2$ and outputs $y^{(i)} \in \{-1, 1\}$, respectively, with one training example per row. This is a binary classification problem.

1. Read the data (you can use Pandas) from the files, and split it into training, validation and test sets (70-15-15%). Make sure to shuffle the data before splitting it.
2. Plot the training data (your axes should be x_1 and x_2, corresponding to the two coordinates of the inputs, and you should use a different symbol for each point plotted to indicate whether that example had label 1 or -1, and whether it is a training or test data point).
3. Use Logistic Regression to fit a logistic regression model to the data. (Extra credit (5 points): Write the stochastic gradient descent algorithm and check if you get a similar result).
4. Plot the decision boundary. This should be a straight line showing the boundary separating the region where $f_\theta(\mathbf{x}) > 0.5$ from where $f_\theta(\mathbf{x}) \leq 0.5$.). What is the test score of the model?
5. What is the purpose of the penalty argument in the LogisticRegression classifier?
6. Try the L_1, L_2 and *ElasticNet* (which includes both L_1 and L_2) penalties and compare their decision boundaries as well as their test scores. (Look these up in SKLearn)
7. How does SVM compare to Logistic Regression on this data-set? For this problem, you don't need to understand what SVM is, but you'll see that you can use it without understanding what it does. All you need to know here is that it's yet another algorithm for classification (there are a ton of these!).
8. Test all methods on the test set you've kept on the side at the beginning.
9. Open ended, extra credit (5 points): search for 3 other classification algorithms that you can use on this data-set and state their advantages over logistic regression?

Problem 5: Find The Polynomial (30 points)

Sign up to kaggle.com and make teams of up to 3. And join the competition through this link: Competition Link. Please don't share the link with people outside the class (for now). The problem statement is to fit a function given two inputs and one output (all real numbers). You're free to use whatever method you see fit. You're allowed only 2 submissions per day. You'll see more details on the link.