Midterm, CS-GY 6923

Basic Info

Date: Thursday 3/10 from 11am - 12pm
Location: Our usual classroom
Supplies needed: Pencil or pen. I will provide scrap paper.
Supplies allowed: One-double sided sheet of paper containing whatever you like. Equations, notes, examples, etc. This can be handwritten or typewritten -- there are no restrictions.

Preparation

The best way to prepare is to review your homework and labs. If there are any homework problems you did not understand or fully solve, please ask about them in office hours!

Question Types

- Short questions similar to the homework but easier.
- True/false, with a short statement to justify your answer.
- "Always, sometimes, never" questions. I give you a statement, you say if it's always true, sometimes true, or never true, and give a short written justification.
- Data diagnostics: I show you a plot or table of data and ask you what machine learning method, or data modification, might be appropriate for the data.

Topics

Below are a list of things you should know, roughly catagorized into topics.

Introduction to Machine Learning

- What is *supervised* learning?
- What is *classification*? What is *regression*?
- What is a *predictor variable* what is a *target variable*? What is a *data matrix*?
- Loss minimization framework: what is a *model*, the *model parameters*, and the *loss function*. The objective of supervised learning in the loss minimization framework is to choose model parameters which minimize the loss.

Simple Linear Regression

- What is the model for simple linear regression? What are the parameters?
- Know these different loss functions for regression: ℓ_2, ℓ_1, ℓ_∞. You should be able to come up with other reasonable loss functions on your own.
- Why might you use one loss over another? Understand the intuition for why ℓ_2 and ℓ_1 are more robust to outliers than ℓ_∞. Come up with an example problem where you might want to use ℓ_∞ loss any way.
- To minimize a function with multiple inputs, *set all partial derivatives to 0* to find extreme points.

- Or run brute force search!
- How do you set β_0 and β_1 to minimize the squared loss for simple linear regression. *You should be able to derive these expressions entirely on your own.*
- What are some reasons we might get a poor fit during linear regression? Which problems can we address? Which are inherent issues with the data or model?
- Given a non-linear problem, know how to transform variables to convert it into a linear model (when possible).

Multiple Linear Regression

- What is the multiple linear regression model? Write it using matrix/vector notation.
- What losses could be used for multiple linear regression? Write using matrix/vector notation. E.g. the squared loss would be $L(\boldsymbol{\beta}) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2$.
- Adding a column of all 1s on to the front of the data matrix allow us to eliminate the intercept parameter. Adding a column of all 2s would have worked as well. Or a column of all -2s. We would get the exact sample predictions from a linear regression model. Why? A column off al 0s would not work. Why?
- What is the *gradient* of a function? Setting the gradient to $\vec{0}$ is the same as setting all partial derivatives to 0 .
- What is the gradient for multiple linear regression under the squared loss? Be able to compute the gradient for other loss functions (as in HW 1) using calculus + chain rule.
- Make sure you sanity check all gradient computations: if L is a function which takes a d-dimensional vector as input and outputs a number (the loss), ∇L takes in a d-dimensional vector and outputs a d-dimensional vector (the gradient).
- Be able to compute gradients of other simple multivariate functions. E.g. what is the gradient of $f(x) = \langle \mathbf{a}, \mathbf{x} \rangle$, or $f(\mathbf{x}) = e^{\langle \mathbf{a}, \mathbf{x} \rangle}$?
- Once you have the gradient, how do you solve for the optimal parameter vectors $\boldsymbol{\beta}$ for linear regression under squared loss: Compute $(X^TX)^{-1}X^Ty$.
- Data scaling and centering has *no effect* on the optimal parameters for multiple linear regression.
- How can multiplie linear regression be used with data transformations to fit a q-degree polynomial? A sinusoidal curve? Other models?
- What is *one-hot encoding*? Why is it necessary?

Model Selection, Train-Test Split

- What is *overfitting*? What is *generalization*? You don't need to know mathematical definitions, but understand what they mean intuitvely.
- Does better loss on the training set mean better generalization? Always, sometimes, or never?
- What is *model selection* formally? See Lecture 3, slide 4.
- Here are some examples of model classes of varying complexity: polynomials with different degrees, bag-of-words with different values of n in the n-gram computation, time series models with different delays (like on the lab). You might use model selection in all of these cases to determine the right model complexity. You might use to deterimine the right regularization parameter.
- How to split a data into a training set and a test set.
- What is the definition of *population risk*? Our objective in machine learning is almost always to find models that minimize population risk.

- The expected value of the *empirical risk on the test set* is equal to the population risk. Why is this observation useful?
- What is k-fold cross validation? Why would you use it instead of train-test split? Why would you not use it?

Feature Selection and Regularization

- What is regularization? How does it change the loss-minimization framework?
- For what types of machine learning problems do you need to use feature selection or regularization for?
- Typically we need one or the other for over-parameterized models, where the number of predictor features is larger than the number of training data points. Why are you essentially *guaranteed* to overfit in this setting when you have a linear regression model?
- Both approaches can also prevent overfitting even when you aren't obviously overparameterized.
- What is *ridge regularization* (ℓ_2) and *LASSO regularization* (ℓ_1)?
- If you add a regularization penalty, do the parameters in your optimal β increase or decrease in comparison to if you had minimized the loss without regularization?
- Note that not *every* parameter necessarily decreases, but we proved that if you add a regularization term of $\lambda\|\boldsymbol{\beta}\|$ with $\lambda > 0$, then $\|\boldsymbol{\beta}\|$ decreases. This is true for *any regularization norm*.
- What's one advantage of feature selection over regularization? It leads to a simpler model, where y is predicted using fewer parameters.
- How is regularization justified from a Bayesian perspective?
- What is the maximum likelihood (MLE) estimate for $\boldsymbol{\beta}$ under an assumption that our model is linear with Gaussian noise? What about under Laplace noise?
- Be able to derive why a Gaussian prior on the parameters $\boldsymbol{\beta}$ tells us to use ℓ_2 regularization and why a Laplace prior tells use to use ℓ_1 (LASSO regularization). What is the *log likelihood function*?

Naive Bayes + Probabilistic Modeling

- Be able to come up with a probabilistic model for a given data set and understand how you would learn parameters of the model from data.
- Definition of *joint probability* $p(a, b)$ and *conditional probability* $p(a \mid b)$. Which is larger?
- Bayes Rule: $p(a \mid b) = \frac{p(b|a)p(a)}{p(b)}$. $p(b \mid a)$ is called the *likelihood*, $p(a)$ is the *prior*, $p(b)$ is the *evidence* and $p(a \mid b)$ is the *posterior*.
- What is a *bag-of-words vector representation* of a document? Why is it useful?
- What probabilistic model for binary vectors justifies the *Naive Bayes Classifier*? How are the parameters of this model learning from past data.
- What is the final *Naive Bayes* classification rule?
- What is the equation for a Gaussian/normal PDF? For a Laplace PDF? You might want to write this on your cheat sheet!
- What is the difference between maximum likelihood estimation (MLE) and maximum a posterior (MAP) estimation?

Linear Classifiers and Logistic Regression

- What is a linear classifier? How would you right down a model for linear classification?
- What is the 0-1 loss?
- The logistic function $h(z)) = \frac{1}{1+e^{-z}}$ and what it looks like if you plotted it.
- What is the model and loss function which define *logistic regression*.
- What is the *error rate*, *precision*, and *recall* of a classifier? How would you adjust a classifier to improve precision or recall? In what real world scenario would you care about precision more? Recall more?
- How do the *one-vs-all* and *one-vs-all* methods work for multiclass classification? In general, one-vs.-one leads to more accurate results, but at the cost of increase computational complexity. What is that complexity overhead?
- What is a *confusion matrix*.

Optimization and Gradient Descent

- Why do we need to use search methods to minimize loss funtions instead of just figuring out when the gradient equals $\vec{0}$.
- How would you do a brute force search for the optimal parameters $\boldsymbol{\beta}$ of a model? Why would this be slow when $\boldsymbol{\beta}$ has more than a few dimensions?
- The gradient desent update is $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \nabla L(\beta)$ where η is the *stepsize*, aka *learning rate*, parameter.
- $\lim_{\eta \to 0} L(\boldsymbol{\beta} + \vec{v}) - L(\boldsymbol{\beta}) = \eta \langle \nabla L(\boldsymbol{\beta}), \vec{v} \rangle$.
- Why is the gradient descent update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \nabla L(\beta)$ guaranteed to decrease $L(\boldsymbol{\beta})$ for a sufficiently small step-size η?
- Why is gradient descent sometimes called *steepest descent*? Understand why $\nabla L(\boldsymbol{\beta})$ is the "best" update.
- What is the definition of a convex function. Be able to prove or argue why a basic function like $f(x) = x^2$ is convex. What stronger guarantees can we prove for gradient descent when our function is convex? Convergence to local minimum, number of steps depending on parameters of the function.
- What is the stochastic gradient descent and what functions can it be applied to? Be able to compute the cost of computing a full gradient and/or a stochastic gradient for various loss function. SGD trades cheaper iterations for slower convergence.