1 Introduction

CS 189 / 289A  [Spring 2022]
Machine Learning
Jonathan Shewchuk

https://people.eecs.berkeley.edu/~jrs/189/

Questions: Please use Piazza, not email. [Piazza has an option for private questions, but please use public for most questions so other people can benefit.]
   For personal matters only, cs189-sp22@berkeley.edu (goes to me and Head TA Kevin Li).

Discussion sections (Mon–Wed):
   Attend any section. [We’ll put up a list on Piazza.]
   [We might have a few advanced sections, including research discussion or exam problem preparation.]
   Sections start Monday. [Next week.]

Homework 1 (see Piazza) due next Wednesday.

[Enrollment: 660 students max. 314 waitlisted. Expecting many drops. EECS grads have highest priority; undergrads second; non-EECS grads third; little hope for concurrent enrollment students.]

[Textbooks: Available free online. Linked from class web page.]
Prerequisites

- Vector calculus: Math 53 [or another vector calculus course]
- Linear algebra: Math 54, Math 110, or EE 16A+16B [or another linear algebra course]
- Probability: CS 70, EECS 126, or Stat 134 [or another probability course]
- Plentiful programming experience [TAs have no obligation to look at your code.]
- NOT CS 188

Grading: 189

- 40% 7 Homeworks. Late policy: 5 slip days total
- 20% Midterm: Tentatively Wednesday, March 16, in class (6:30–8 PM)
- 40% Final Exam: Friday, May 13, 3–6 PM

Grading: 289A

- 40% HW
- 20% Midterm
- 20% Final
- 20% Project

Cheating

- Discussion of HW problems is encouraged. Showing other students small amounts of code is okay.
- All homeworks, including programming, must be written individually.
- We will actively check for plagiarism.
- Typical penalty is a large NEGATIVE score, but I reserve right to give an instant F for even one violation, and will always give an F for two.

[Last time I taught CS 61B, we had to punish roughly 100 people for cheating. It was very painful. Please don’t put me through that again.]

CORE MATERIAL

- Finding patterns in data; using them to make predictions.
- Models and statistics help us understand patterns.
- Optimization algorithms “learn” the patterns.

[The most important part of this is the data. Data drives everything else.

You cannot learn much if you don’t have enough data.

You cannot learn much if your data sucks.

But it’s amazing what you can do if you have lots of good data.

Machine learning has changed a lot in the last two decades because the internet has made truly vast quantities of data available. For instance, with a little patience you can download tens of millions of photographs. Then you can build a 3D model of Paris.

Some techniques that had fallen out of favor, like neural networks, have come back big in recent years because researchers found that they work so much better when you have vast quantities of data.]
CLASSIFICATION

FIGURE 4.1. The Default data set. Left: The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue. Center: Boxplots of balance as a function of default status. Right: Boxplots of income as a function of default status.

4.2 Why Not Linear Regression?

We have stated that linear regression is not appropriate in the case of a qualitative response. Why not?

Suppose that we are trying to predict the medical condition of a patient in the emergency room on the basis of her symptoms. In this simplified example, there are three possible diagnoses: stroke, drug overdose, and epileptic seizure. We could consider encoding these values as a quantitative response variable, $Y$, as follows:

- $Y = 1$ if stroke;
- $Y = 2$ if drug overdose;
- $Y = 3$ if epileptic seizure.

Using this coding, least squares could be used to fit a linear regression model to predict $Y$ on the basis of a set of predictors $X_1, \ldots, X_p$. Unfortunately, this coding implies an ordering on the outcomes, putting drug overdose in between stroke and epileptic seizure, and insisting that the difference between stroke and drug overdose is the same as the difference between drug overdose and epileptic seizure. In practice there is no particular reason that this needs to be the case. For instance, one could choose an equally reasonable coding, $Y = 1$ if epileptic seizure; $Y = 2$ if stroke; $Y = 3$ if drug overdose.

The problem of classification. We are given data points, each belonging to one of two classes. Then we are given additional points whose class is unknown, and we are asked to predict what class each new point is in. Given the credit card balance and annual income of a cardholder, predict whether they will default on their debt.

- Collect training data: reliable debtors & defaulted debtors
- Evaluate new applicants (prediction)
[Here are two examples of classifiers for the same data. At left we have a nearest neighbor classifier, which classifies a point by finding the nearest point in the training data, and assigning it the same class. At right we have a linear classifier, which guesses that everything above the line is brown, and everything below the line is blue. The decision boundaries are in black.]

[At right we have a 15-nearest neighbor classifier. Instead of looking at the nearest neighbor of a new point, it looks at the 15 nearest neighbors and lets them vote for the correct class. The 1-nearest neighbor classifier at left has a big advantage: it classifies all the training data correctly, whereas the 15-nearest neighbor classifier at right figure does not. But the right figure has an advantage too. Somebody please tell me what.]

[The left figure is an example of what’s called overfitting. In the left figure, observe how intricate the decision boundary is that separates the positive examples from the negative examples. It’s a bit too intricate to reflect reality. In the right figure, the decision boundary is smoother. Intuitively, that smoothness is probably more likely to correspond to reality.]
Classifying Digits

[sevensones.pdf] In this simplified digit recognition problem, we are given handwritten 7’s and 1’s, and we are asked to learn to distinguish the 7’s from the 1’s.

Express these images as vectors

3 3 3 3
0 0 2 3
0 0 1 3
3 3 3 3

Images are points in 16-dimensional space. Linear decision boundary is a hyperplane.

TESTING AND VALIDATION

- Train a classifier: it learns to distinguish 7 from not 7
- Test the classifier on NEW images

2 kinds of error:
- Training set error: fraction of training images not classified correctly
  [This is zero with the 1-nearest neighbor classifier, but nonzero with the 15-nearest neighbor and linear classifiers we’ve just seen.]
- Test set error: fraction of misclassified NEW images, not seen during training.

[When I underline a word or phrase, that usually means it’s a definition. If you want to do well in this course, my advice to you is to memorize the definitions I cover in class.]

outliers: points whose labels are atypical (e.g., solvent borrower who defaulted anyway).
overfitting: when the test error deteriorates because the classifier becomes too sensitive to outliers or other spurious patterns.

[In machine learning, the goal is to create a classifier that generalizes to new examples we haven’t seen yet. Overfitting is counterproductive to that goal. So we’re always seeking a compromise: we want decision boundaries that make fine distinctions without being downright superstitious.]
Most ML algorithms have a few hyperparameters that control over/underfitting, e.g. $k$ in $k$-nearest neighbors.

![Graph showing test error and training error with k: # of nearest neighbors](overfitted.pdf)

We select them by validation:
- Hold back a subset of the labeled data, called the validation set.
- Train the classifier multiple times with different hyperparameter settings.
- Choose the settings that work best on validation set.

Now we have 3 sets:
- **training set** used to learn model weights
- **validation set** used to tune hyperparameters, choose among different models
- **test set** used as FINAL evaluation of model. Keep in a vault. Run ONCE, at the very end.

[It’s very bad when researchers in medicine or pharmaceuticals peek into the test set prematurely!]

Kaggle.com:
- Runs ML competitions, including our HWs
- We use 2 data sets:
  - “public” set labels available during competition
  - “private” set labels known only to Kaggle

[If your public results are a lot better than your private results, we will know that you overfitted.]

**Techniques [taught in this class, NOT a complete list]**

Supervised learning:
- Classification: is this email spam?
- Regression: how likely does this patient have cancer?

Unsupervised learning:
- Clustering: which DNA sequences are similar to each other?
- Dimensionality reduction: what are common features of faces? common differences?