link to lecture

CS 189 / 289A Machine Learning Jonathan Shewchuk

Class website: cs.berkeley.edu/~jrs/189

TAs: Tuomas Haarnoja, Aldo Pacchiano, Rohan Chitnis, Shaun Singh, Marvin Zhang, Brian Hou

Discussion sections:

Go to schedule.berkeley.edu and look up CS 189 (NOT 289A). You choose your section. If the room is too full, please go to another one. Sections 102 and 103 are cancelled. No sections this week.

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, jrs@cory.eecs.berkeley.edu

Enrollment:

More than 100 of you were admitted around noon today. Most non-CS grad students won't be admitted.

For those of you taking CS 162, I have good news: CS 162 has been moved to exam group 3. There is no longer a final exam conflict between 189 and 162.

Textbooks:

An Introduction to Statistical Learning with Applications in R

The Elements of Statistical Learning: Data Mining, Inference, and Prediction.

Prerequisites

Math 53 (vector calculus)

Math 54 (linear algebra)

CS 70 (discrete math; probability)

CS 188 (AI; probability; decision theory)

Grading: 189

40% Homework: 7 assignments. Late policy: 5 slip days total.

20% Midterm: Wednesday, March 16, in class (6:30-8 pm)

40% Final Exam: Friday, May 13, 3-6 PM (Exam group 19)

Grading: 289A

40% HW

20% Midterm

20% Final Exam

20% Project

Cheating

- Discussion of HW problems is encouraged.
- 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 helps 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 decade 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 nets, have come back big in the last few years because researchers found that they work so much better when you have vast quantities of data.]

EXAMPLE: CLASSIFYING DIGITS

[Slide: Classifiers.]

[Slide: Example: Classifying Digits.]

- Collect training images: e.g. 7’s and digits that are not 7’s
- Express these images as vectors

$$\begin{bmatrix} 3 & 3 & 3 & 3 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 3 & 3 & 3 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0 \\ 0 \\ 1 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$$

- Draw 2 colors of dots, almost but not quite linearly separable.
- “How do we classify a new point?” Draw a third color point.
- One possibility: look at nearest neighbor.
- Another possibility: draw linear “decision boundary”; label it.

Slide: nearest neighbor classifier and linear classifier.

Slide: nearest neighbor and k-nearest neighbor. The left figure has a big advantage: it classifies all the test data correctly, whereas the 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.]

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: The linear classifier doesn't classify all 7's / not 7's correctly - Test set error: Try out new images, not used during training. Some of them might be classified wrong.

outliers: samples 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 between decision boundaries that make fine distinctions and decision boundaries that are downright superstitious.]

[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.]

Most ML algorithms have a few *hyperparameters* that control over/underfitting, e.g. k in k-nearest neighbors. We select them by

validation: - Hold back a subset of training 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 hyperparamters, 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 test sets:

- "public" set results available during competition
- "private" set revealed only after due date

    [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:

- Density estimation: what is probability density function of rainfall?
- Clustering: which DNA sequences are similar to each other?
- Dimensionality reduction: what are common features of faces? common differences?

[Show slides with examples of:

- Classification
- Regression
- Density estimation
- Clustering
- Dimensionality reduction

]