1 Introduction

CS 189 / 289A [Spring 2017]
Machine Learning
Jonathan Shewchuk

http://www.cs.berkeley.edu/~jrs/189

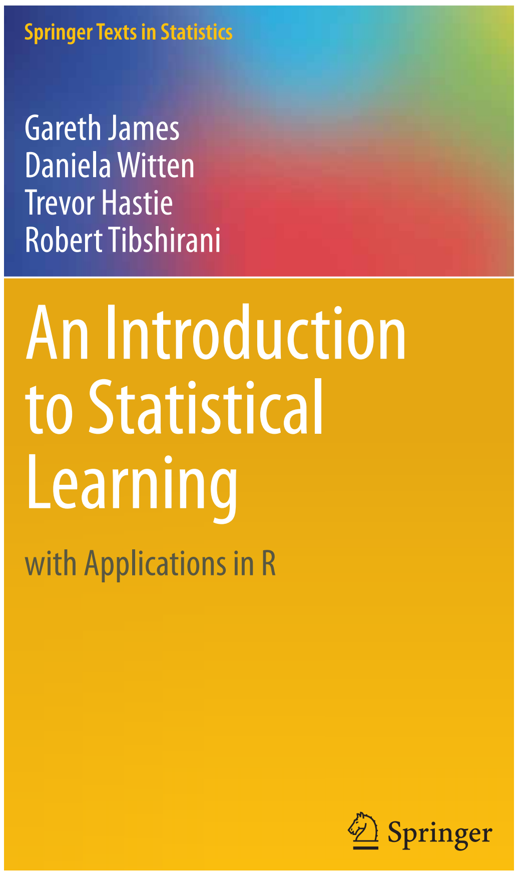
TAs: Daylen Yang, Ting-Chun Wang, Moulos Vrettos, Mostafa Rohaninejad, Michael Zhang, Anurag Ajay, Alvin Wan, Soroush Nasiriany, Garrett Thomas, Noah Golmant, Adam Villaflor, Raul Puri, Alex Francis

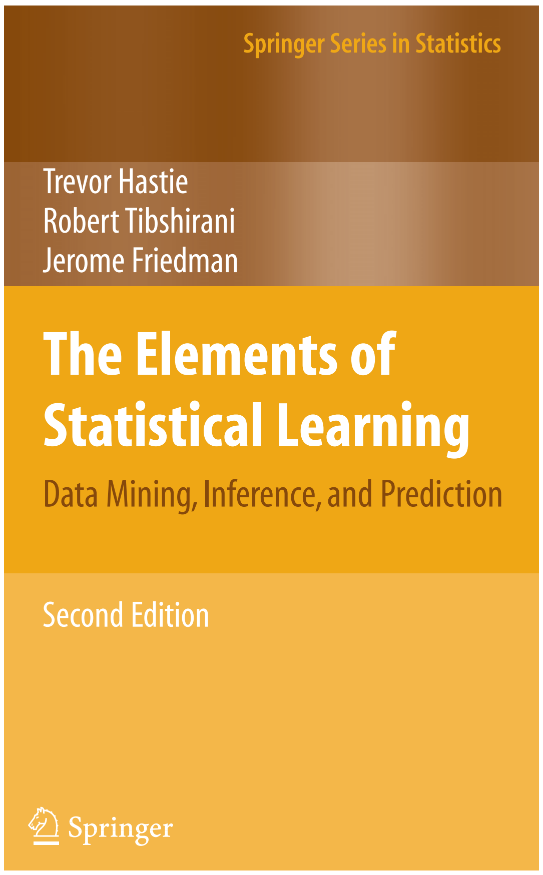
Questions: Please use Piazza, not email. [Piazza has an option for private questions, but please use public for most questions so other people can benefit.]
For personal matters only, jrs@cory.eecs.berkeley.edu

Discussion sections:
12 now; more will be added.
Attend any section. If the room is too full, please go to another one.
[However, to get into the course, you have to pick *some* section with space. Doesn't matter which one!]
No sections this week.

[Enrollment: We're trying to raise it to 540. After enough students drop, it's possible that everyone might get in. Concurrent enrollment students have the lowest priority; non-CS grad students the second-lowest.]

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


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer

Prerequisites

Math 53 (vector calculus)
Math 54 or 110 (linear algebra)
CS 70 (probability)
NOT CS 188 [Might still be listed as a prerequisite, but we're having it removed.]
[BUT be aware that 189 midterm starts 10 minutes after 188 midterm ends.]

Grading: 189

40% 7 Homeworks. Late policy: 5 slip days total
20% Midterm: Wednesday, March 15, in class (6:30–8 pm)
40% Final Exam: MOVED to Monday, May 8, 3–6 PM (Exam group 3)
[Good news for some of you who had final exam conflicts.]

Grading: 289A

40% HW
20% Midterm
20% Final
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 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 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.]

CLASSIFICATION

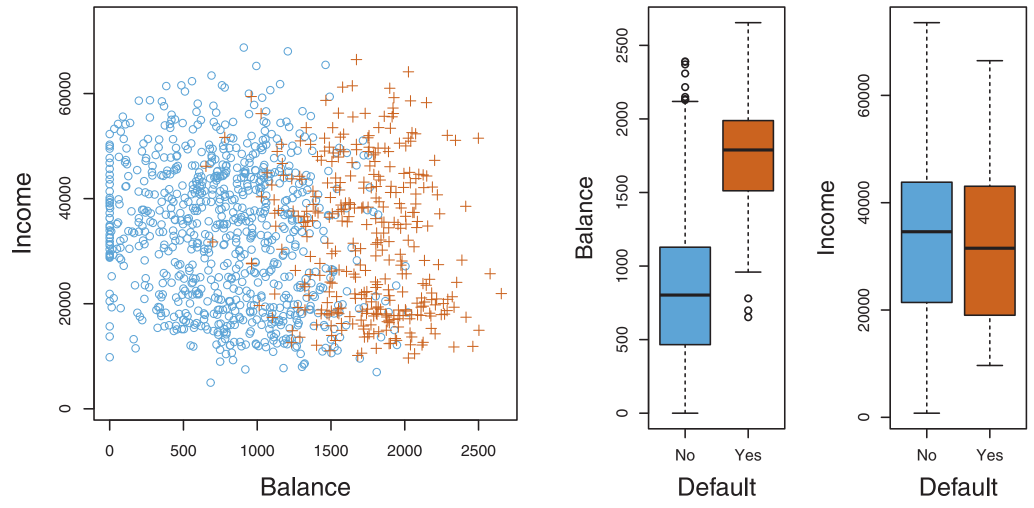
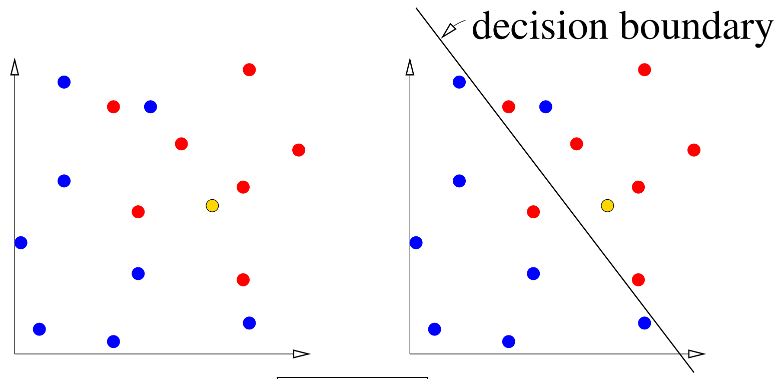


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of*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

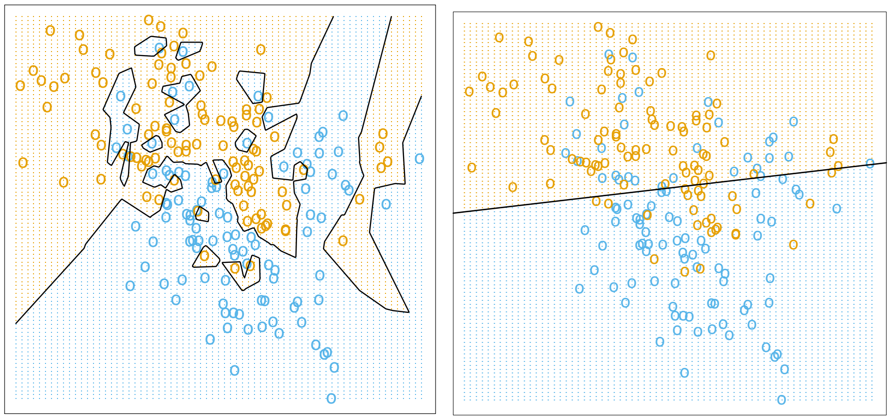
$\boxed{\text{creditcards.pdf (ISL, Figure 4.1)}}$ [The problem of classification. We are given data points, each belonging to one of two classes. Then we are given additional points whose class is unknown, and we are asked to predict what class each new point is in. Given the credit card balance and annual income of a cardholder, predict whether they will default on their debt.]

– Collect training data: reliable debtors & defaulted debtors
– Evaluate new applicants (prediction)

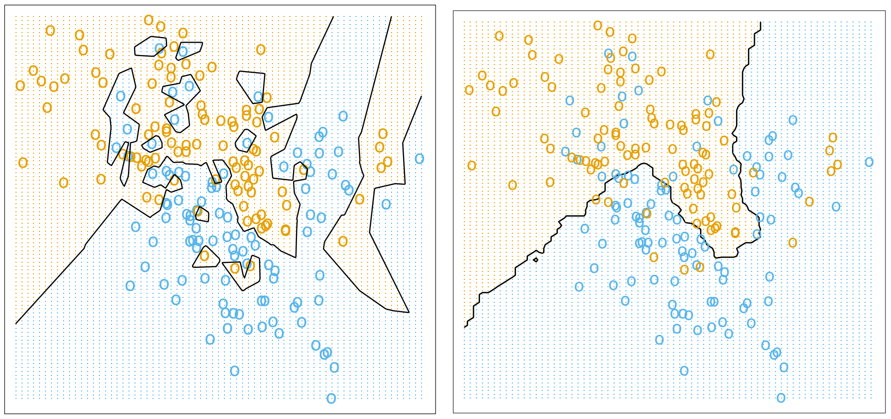


[Draw this figure by hand. $\boxed{\text{classify.pdf}}$]
[Draw 2 colors of dots, almost but not quite linearly separable.]
["How do we classify a new point?" Draw a point in a third color.]
[One possibility: look at its nearest neighbor.]
[Another possibility: draw a linear <u>decision boundary</u>; label it.]
[Those are two different *models* for the data.]

[We'll learn some ways to draw these linear decision boundaries in the next several lectures. But for now, let's compare this method with another method.]



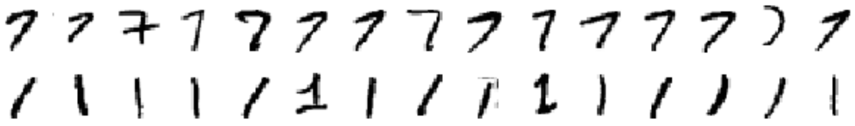
classnear.pdf, classlinear.pdf (ESL, Figures 2.3 & 2.1) [Here are two examples of classifiers for the same data. At left we have a nearest neighbor classifier, which classifies a point by finding the nearest point in the input data, and assigning it the same class. At right we have a linear classifier, which guesses that everything above the line is brown, and everything below the line is blue. The decision boundaries are in black.]



classnear.pdf, classnear15.pdf (ESL, Figures 2.3 & 2.2) [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

$$\begin{array}{|c|c|c|c|}\hline 3 & 3 & 3 & 3 \\ \hline 0 & 0 & 2 & 3 \\ \hline 0 & 0 & 1 & 3 \\ \hline 3 & 3 & 3 & 3 \\ \hline \end{array} \rightarrow \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0 \\ 0 \\ 1 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$$

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

Validation

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

2 kinds of error:

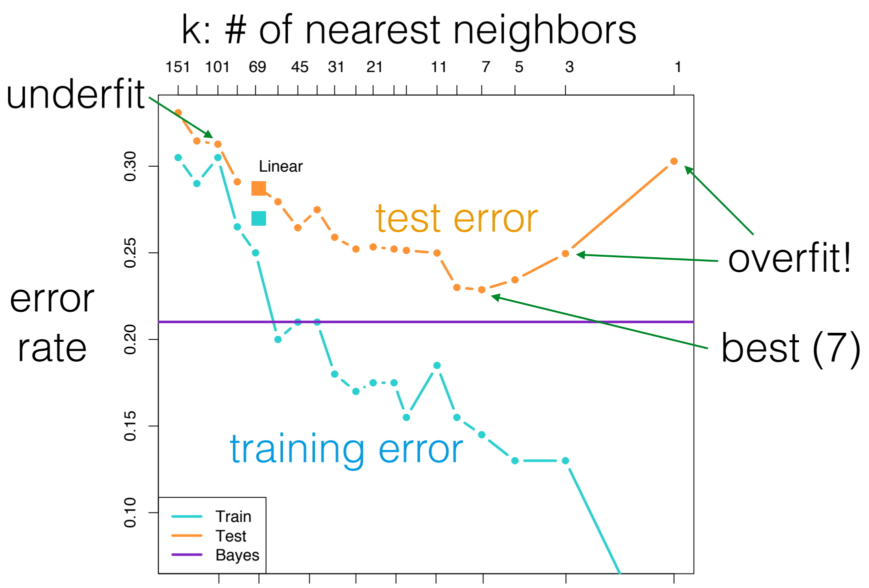
- Training set error: fraction of training images not classified correctly
  [This is zero with the 1-nearest neighbor classifier, but nonzero with the 15-nearest neighbor and linear classifiers we've just seen.]
- Test set error: fraction of misclassified NEW images, not seen during training.

[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.



overfitlabeled.pdf (modified from ESL, Figure 2.4)

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