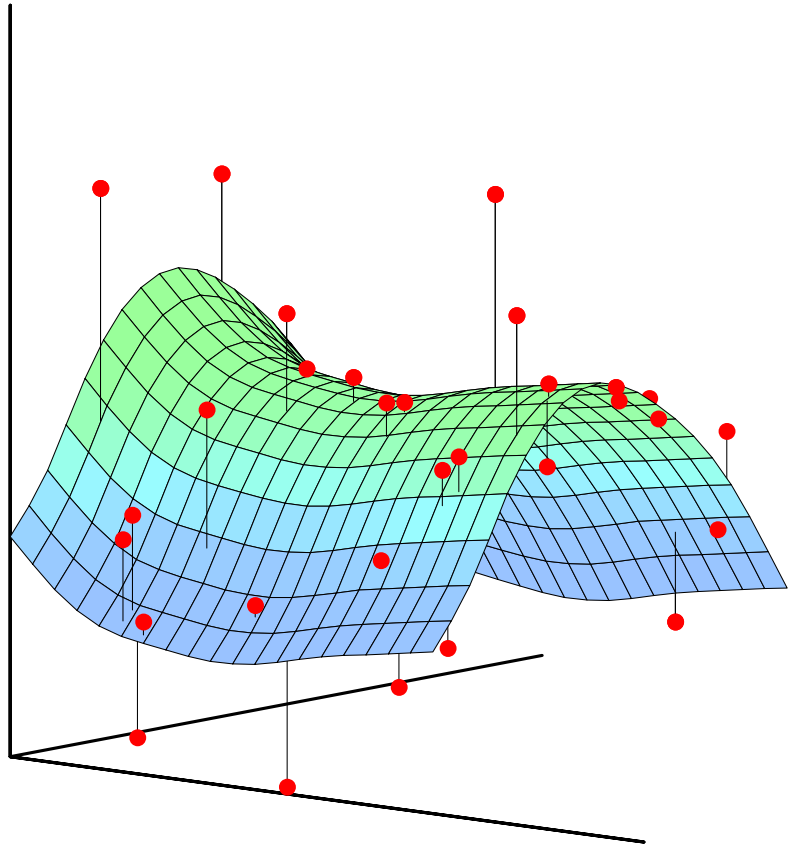
Modern Trends in Data Mining

Stanford University



For Today's Graduate, Just One Word: Statistics

By STEVE LOHR
Published: August 5, 2009

MOUNTAIN VIEW, Calif. — At Harvard, Carrie Grimes majored in anthropology and archaeology and ventured to places like Honduras, where she studied Mayan settlement patterns by mapping where artifacts were found. But she was drawn to what she calls "all the computer and math stuff" that was part of the job.

Enlarge This Image

Thor Swift for The New York Times
Carrie Grimes, senior staff engineer at Google, uses statistical analysis of data to help improve the company's search engine.

Multimedia

"People think of field archaeology as Indiana Jones, but much of what you really do is data analysis," she said.

Now Ms. Grimes does a different kind of digging. She works at Google, where she uses statistical analysis of mounds of data to come up with ways to improve its search engine.

Ms. Grimes is an Internet-age statistician, one of many who are changing the image of the profession as a place for dronish number nerds. They are finding themselves increasingly in demand — and even cool.

"I keep saying that the sexy job in the next 10 years will be statisticians," said Hal Varian, chief economist at Google. "And I'm not kidding."

Quote of the Day, New York Times, August 5, 2009

"I keep saying that the sexy job in the next 10 years will be statisticians. And I'm not kidding." - HAL VARIAN, chief economist at Google.

Datamining for Prediction

- We have a collection of data pertaining to our business, industry, production process, monitoring device, etc.
- Often the goals of data-mining are vague, such as *"look for patterns in the data"* — not too helpful.
- In many cases a *"response"* or *"outcome"* can be identified as a good and useful target for prediction.
- Accurate prediction of this target can help the organization make better decisions, and save time and money.
- Data-mining is particularly good at building such prediction models — an area known as *"supervised learning"*.

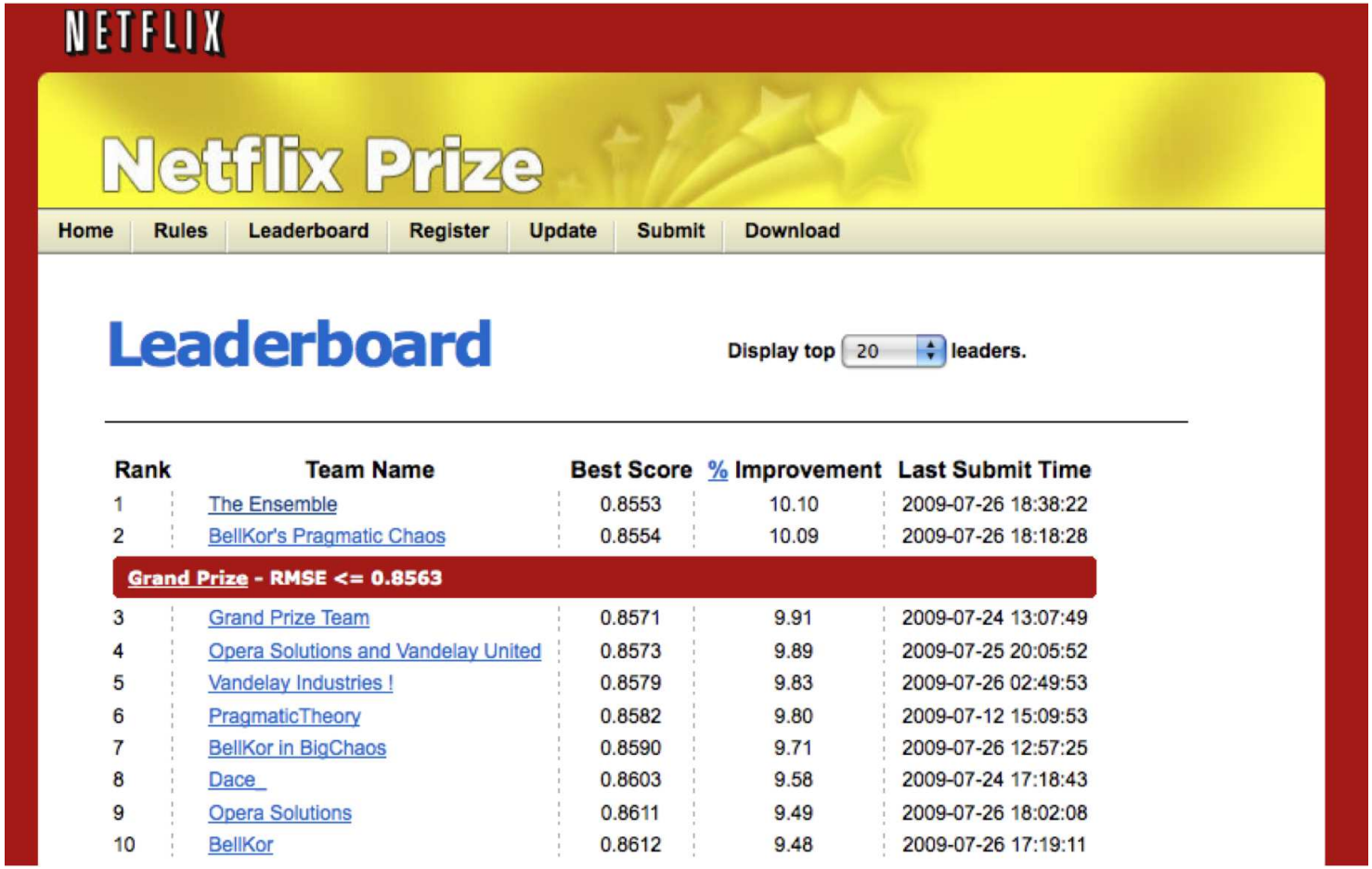
Example: Credit Risk Assessment

- Customers apply to a bank for a loan or credit card.
- They supply the bank with information such as age, income, employment history, education, bank accounts, existing debts, etc.
- The bank does further background checks to establish credit history of customer.
- Based on this information, the bank must decide whether to make the loan or issue the credit card.

Example continued: Credit Risk Assessment

- The bank has a large database of existing and past customers. Some of these defaulted on loans, others frequently made late payments etc. An outcome variable *"Status"* is defined, taking value *"good"* or *"defaulted"*. Each of the past customers is scored with a value for status.
- Background information is available for all the past customers.
- Using supervised learning techniques, we can build a risk prediction model that takes as input the background information, and outputs a risk estimate (probability of default) for a prospective customer.

The California based company *Fair-Isaac* uses a generalized additive model + boosting methods in the construction of their credit risk scores.



Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
Grand Prize - RMSE <= 0.8563				
3	Grand Prize Team	0.8571	9.91	2009-07-24 13:07:49
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-25 20:05:52
5	Vandelay Industries !	0.8579	9.83	2009-07-26 02:49:53
6	PragmaticTheory	0.8582	9.80	2009-07-12 15:09:53
7	BellKor in BigChaos	0.8590	9.71	2009-07-26 12:57:25
8	Dace	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-26 18:02:08
10	BellKor	0.8612	9.48	2009-07-26 17:19:11

Grand Prize: one million dollars, if beat Netflix's RMSE by 10%.

Competition ends after $\approx$ 3 years, two leaders, 41305 teams!

Netflix Challenge

Netflix users rate movies from 1-5. Based on a history of ratings, predict the rating a viewer will give to a new movie.

- Training data: sparse 400K (users) by 18K (movies) rating matrix, with 98.7% missing. About 100M movie/rater pairs.
- Quiz set of about 1.4M movie/viewer pairs, for which predictions of ratings are required (Netflix has held them back)
- Probe set of about 1.4 million movie/rater pairs similar in composition to the quiz set, for which the ratings are known.
- Both winning teams used ensemble methods to achieve their results.

The Supervised Learning Problem

Starting point:

- Outcome measurement Y (also called dependent variable, response, target, output)
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables)
- In the *regression problem*, Y is quantitative (e.g price, blood pressure, rating)
- In *classification*, Y takes values in a finite, unordered set (default yes/no, churn/retain, spam/email)
- We have training data $(x_1, y_1), \ldots, (x_N, y_N)$. These are observations (examples, instances) of these measurements.

Objectives

On the basis of the training data we would like to:

- Accurately predict unseen test cases for which we know X but do not know Y.
- In the case of classification, predict the probability of an outcome.
- Understand which inputs affect the outcome, and how.
- Assess the quality of our predictions and inferences.

More Examples

- Determine whether an incoming email is "spam", based on frequencies of key words in the message
- Identify the numbers in a handwritten zip code, from a digitized image
- Estimate the probability that an insurance claim is fraudulent, based on client demographics, client history, and the amount and nature of the claim.
- Predict the type of cancer in a tissue sample using DNA expression values, and more imprtantly, identify the relevant genes.
- Model the prevalence of disease as a function of genotype information at a small subset of 500K SNPs measured from a DNA sample, and thereby identify the relevant SNPs.

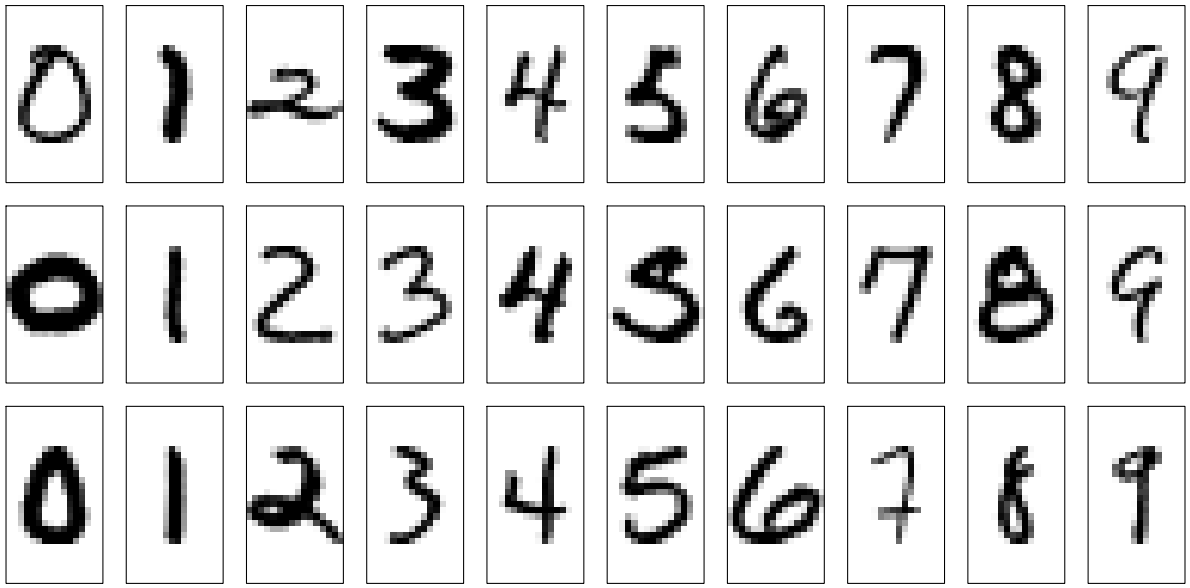
Email or Spam?

- data from 4601 emails sent to an individual (named George, at HP labs, before 2000). Each is labeled as *"spam"* or *"email"*.
- goal: build a customized spam filter.
- input features: relative frequencies of 57 of the most commonly occurring words and punctuation marks in these email messages.

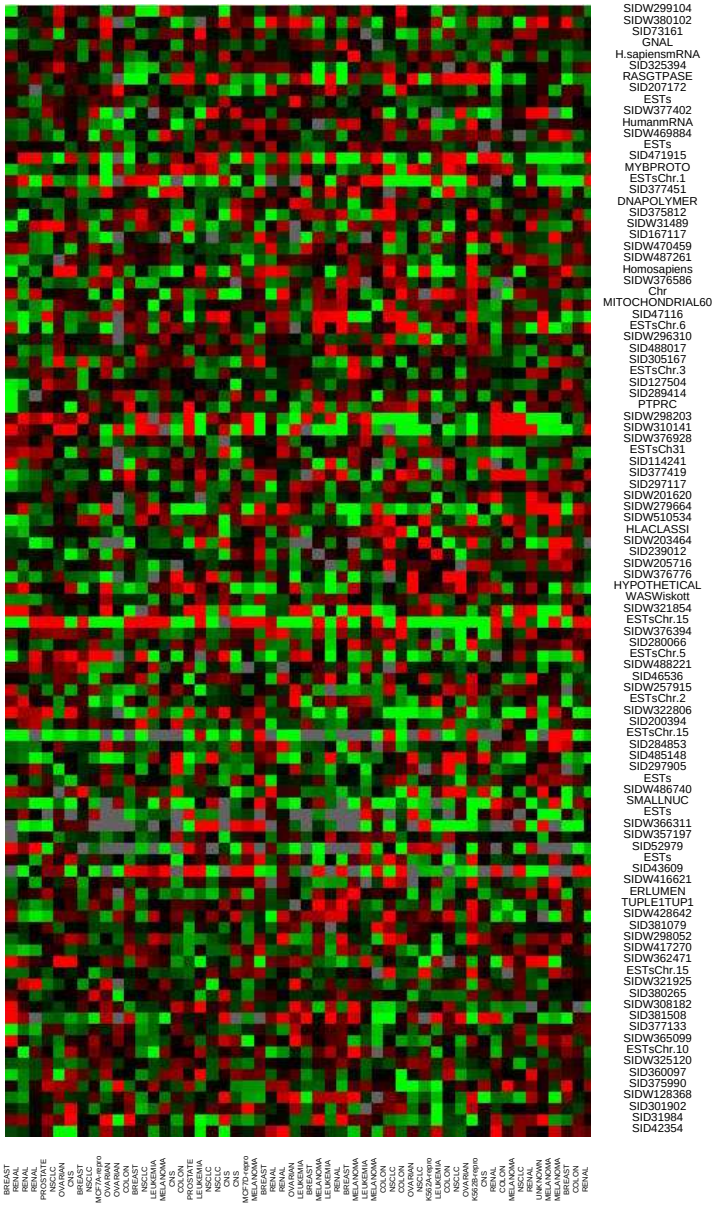
	george	you	hp	free	!	edu	remove
spam	0.00	2.26	0.02	0.52	0.51	0.01	0.28
email	1.27	1.27	0.90	0.07	0.11	0.29	0.01

Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between spam *and* email.

Handwritten Digit Identification



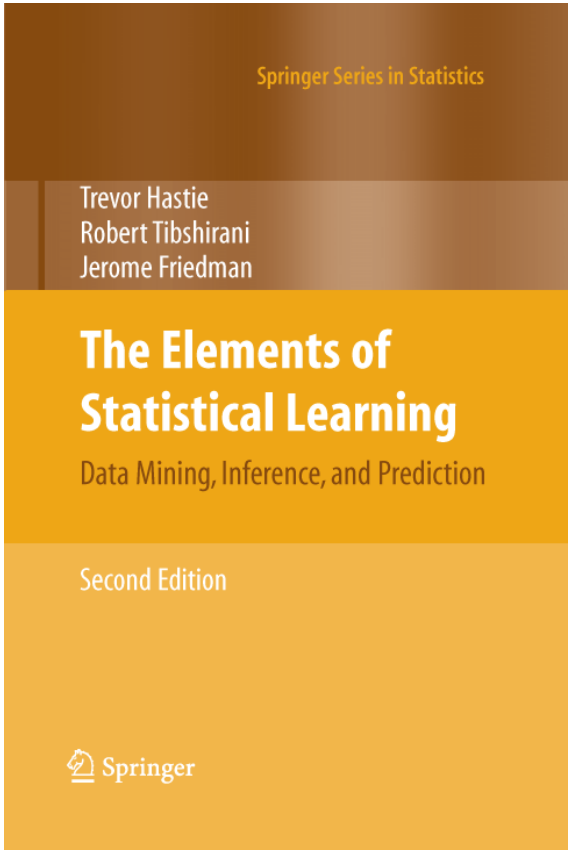
A sample of segmented and normalized handwritten digits, scanned from zip-codes on envelopes. Each image has 16×16 pixels of greyscale values ranging from $0 - 255$.



Microarray Cancer Data

Expression matrix of 6830 genes (rows) and 64 samples (columns), for the human tumor data (100 randomly chosen rows shown). The display is a heat map, ranging from bright green (under expressed) to bright red (over expressed).

Goal: identify which genes are over/under expressed in different cancer classes. Prediction models are a vehicle for achieving this goal.



Shameless self-promotion

All but the last of the topics in this lecture are covered in the 2009 second edition of our 2001 book. The book blends traditional linear methods with contemporary non-parametric methods, and many between the two.

Ideal Predictions

- For a *quantitative output* Y, the best prediction we can make when the input vector $X = x$ is

$$f(x) = \text{Ave}(Y|X = x)$$

  - This is the conditional expectation — deliver the Y-average of all those examples having $X = x$.
  - This is best if we measure errors by average *squared error* $\text{Ave}(Y - f(X))^2$.

- For a *qualitative output* Y taking values 1, 2,. . . , M, compute
  - $\Pr(Y = m|X = x)$ for each value of m. This is the conditional probability of class m at $X = x$.
  - Classify $C(x) = j$ if $\Pr(Y = j|X = x)$ is the largest — the majority vote classifier.

Implementation with Training Data

The ideal prediction formulas suggest a data implementation. To predict at $X = x$, gather all the training pairs (x_i, y_i) having $x_i = x$, then:

- For regression, use the mean of their y_i to estimate $f(x) = \text{Ave}(Y|X = x)$
- For classification, compute the relative proportions of each class among these y_i, to estimate $\Pr(Y = m|X = x)$; Classify the new observation by majority vote.

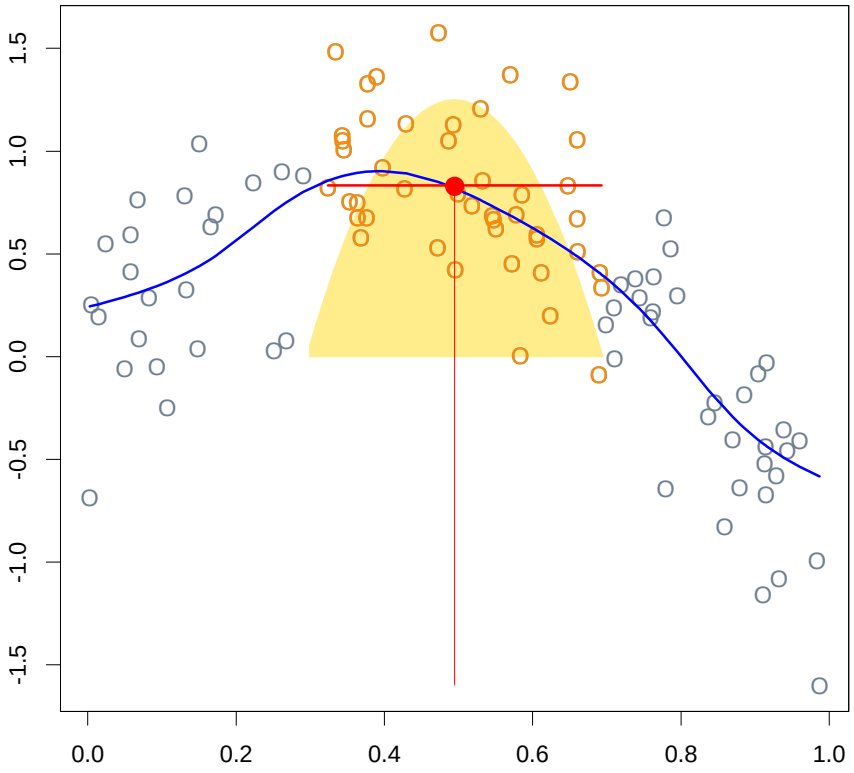
Problem: in the training data, there may be NO observations having $x_i = x$.

Nearest Neighbor Averaging

- Estimate $\text{Ave}(Y|X = x)$ by

  Averaging those y_i whose x_i are in a neighborhood of x.

- E.g. define the neighborhood to be the set of k observations having values x_i closest to x in euclidean distance $||x_i - x||$.
- For classification, compute the class proportions among these k closest points.
- Nearest neighbor methods often outperform all other methods — about one in three times — especially for classification.



Kernel smoothing

- Smooth version of nearest-neighbor averaging
- At each point x, the function $f(x) = Y(Y|X = x)$ is estimated by the weighted average of the y's.
- The weights die down smoothly with distance from the target point x (indicated by shaded orange region).

Structured Models

- When we have a lot of predictor variables, NN methods often fail because of the *curse of dimensionality:*

  It is hard to find nearby points in high dimensions!

- Near-neighbor models offer little interpretation.
- We can overcome these problems by assuming some structure for the regression function $\text{Ave}(Y|X = x)$ or the probability function $\Pr(Y = k|X = x)$. Typical structural assumptions:
  - Linear Models
  - Additive Models
  - Low-order interaction models
  - Restrict attention to a subset of predictors
  - . . . and many more

Linear Models

- Linear models assume

$$\text{Ave}(Y|X = x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

- For two class classification problems, linear logistic regression has the form

$$\log \frac{\Pr(Y = +1|X = x)}{\Pr(Y = -1|X = x)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

- This translates to

$$\Pr(Y = +1|X = x) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p}}$$

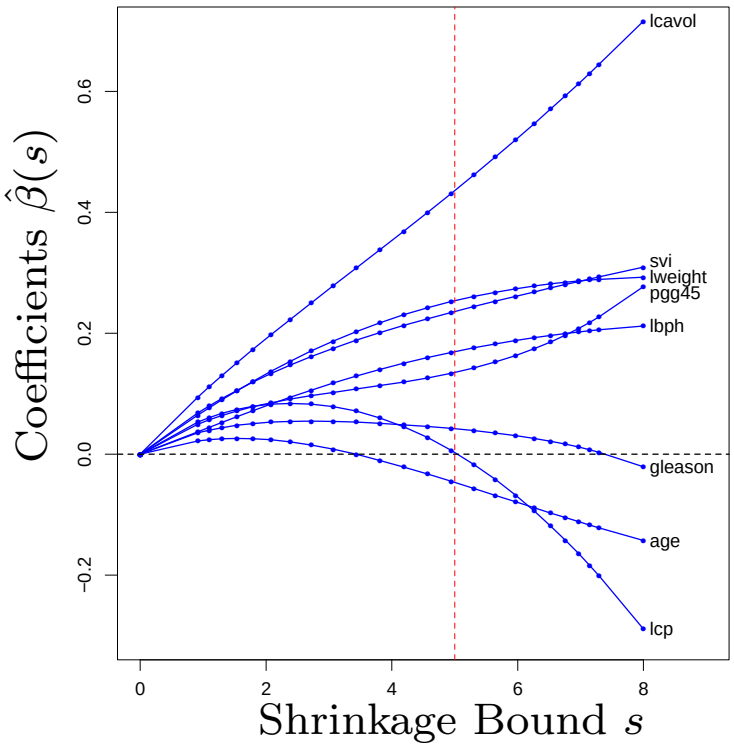
In many of the genomic and web-era problems, we fit linear models with potentially many 1000s of input features.

Linear Model Complexity Control

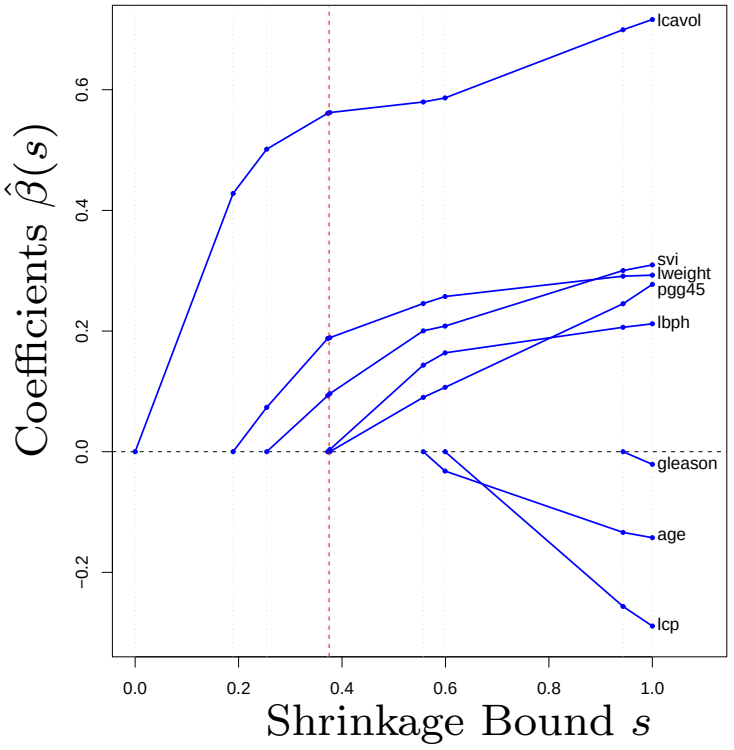
With many input features, linear regression will overfit the training data, leading to poor predictions on future data. Two general remedies are available:

- Variable selection: reduce the number of inputs in the model. For example, *forward stepwise selection* has regained popularity.
- Regularization: leave all the variables in the model, but when fitting the model, restrict their coefficients.
  - *Ridge*: $\sum_{j=1}^{p} \beta_j^2 \leq s$. All the coefficients are non-zero, but are shrunk toward zero (and each other).
  - *Lasso*: $\sum_{j=1}^{p} |\beta_j| \leq s$. Some coefficients drop out the model, others are shrink toward zero.

Ridge



Lasso



Both ridge and lasso coefficients paths can be computed very efficiently *for all values of s*.

Overfitting and Model Assessment

- In all cases above, the larger s, the better we will fit the training data. Often we *overfit* the training data.
- Overfit models can perform poorly on test data (high variance).
- Underfit models can perform poorly on test data (high bias).

Model assessment aims to

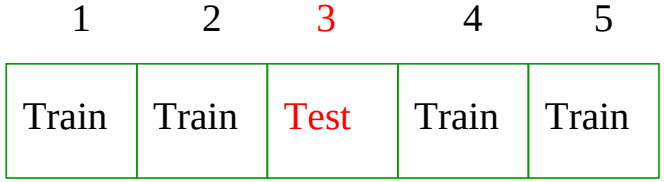
1. Choose a value for a tuning parameter s for a technique.
2. Estimate the future prediction ability of the chosen model.

- For both of these purposes, the best approach is to evaluate the procedure on an independent test set, if one is available.
- If possible one should use different test data for (1) and (2) above: a *validation set* for (1) and a *test set* for (2)

K-Fold Cross-Validation

Primarily a method for estimating a tuning parameter s when data is scarce; we illustrate for the regularized linear regression models.

- Divide the data into K roughly equal parts (5 or 10)

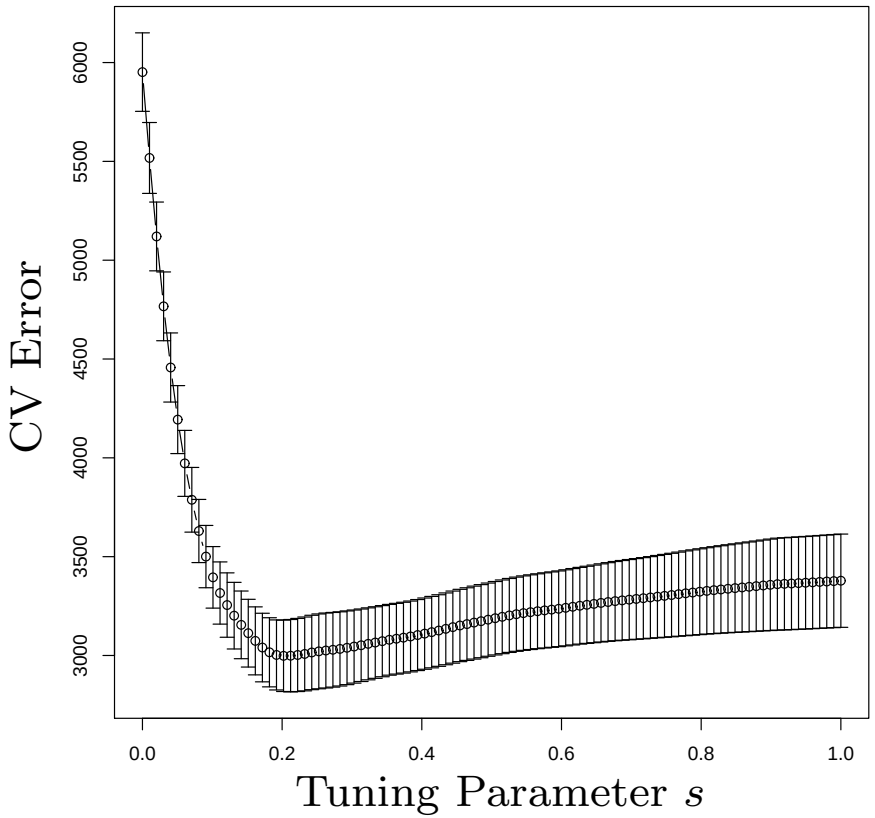


- for each $k = 1, 2, \ldots K$, fit the model with parameter s to the other $K - 1$ parts, giving $\hat{\beta}^{-k}(s)$ and compute its error in predicting the kth part: $E_k(\lambda) = \sum_{i \in k\text{th part}} (y_i - x_i \hat{\beta}^{-k}(s))^2$.
- This gives the overall cross-validation error

$$CV(s) = \frac{1}{K} \sum_{k=1}^{K} E_k(s)$$

- do this for many values of s and choose the value of s that makes $CV(s)$ smallest.

Cross-Validation Error Curve



- 10-fold CV error curve using lasso on some diabetes data (64 inputs, 442 samples).
- Thick curve is CV error curve
- Shaded region indicates standard error of CV estimate.
- Curve shows effect of overfitting — errors start to increase above $s = 0.2$.
- This shows a trade-off between bias and variance.

Modern Structured Models in Data Mining

The following is a list of some of the currently popular prediction models in data mining.

- Linear models with lasso or related regularization
- Generalized additive models/ naive Bayes — simple but effective.
- Neural networks — somewhat old but still used a lot.
- Trees, random forests and boosted tree models.
- Support vector and kernel machines.
- Matrix completion methods.

Generalized Additive Models

Allow a compromise between linear models and more flexible local models (kernel estimates) when there are a many inputs $X = (X_1, X_2, \ldots, X_p)$.

- Additive models for regression:

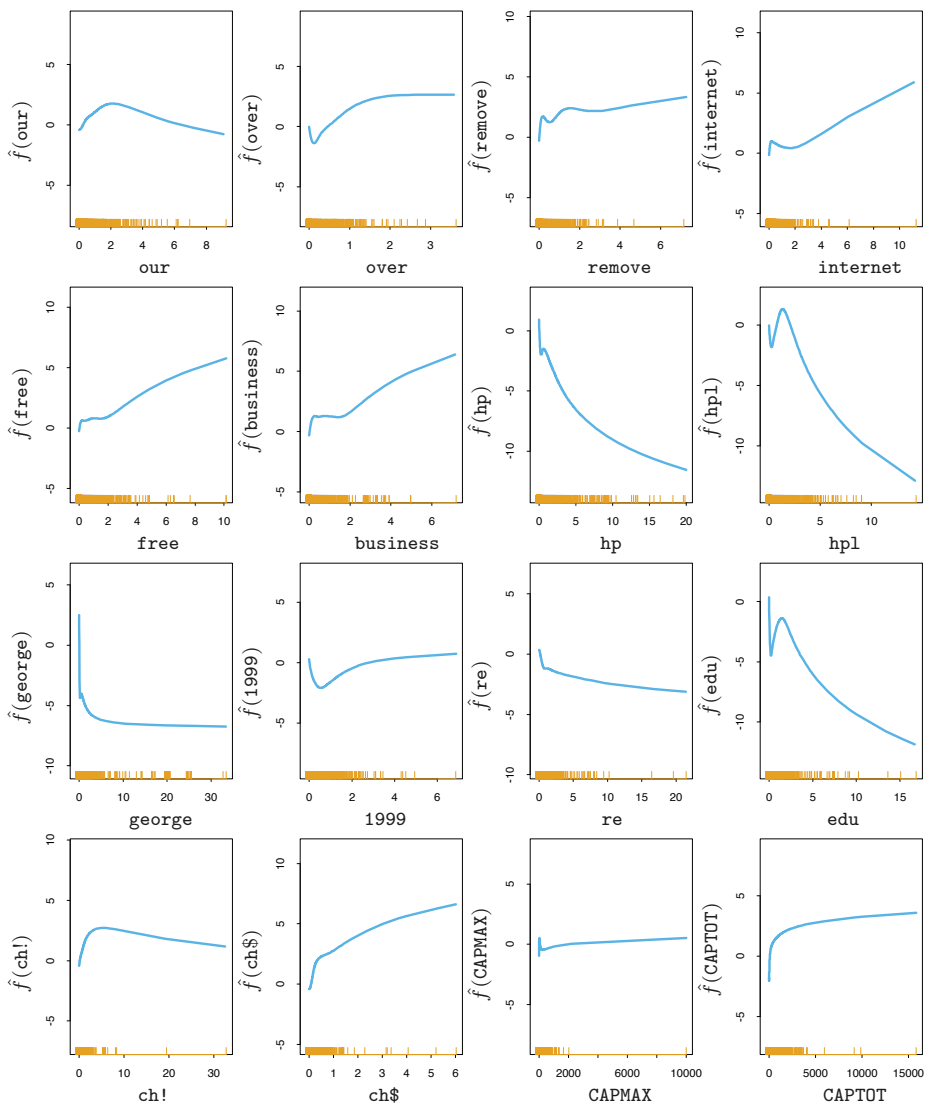
$$\text{Ave}(Y|X = x) = \alpha_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p).$$

- Additive models for classification:

$$\log \frac{\Pr(Y = +1|X = x)}{\Pr(Y = -1|X = x)} = \alpha_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p).$$

  Close relative of *naive Bayes* model.

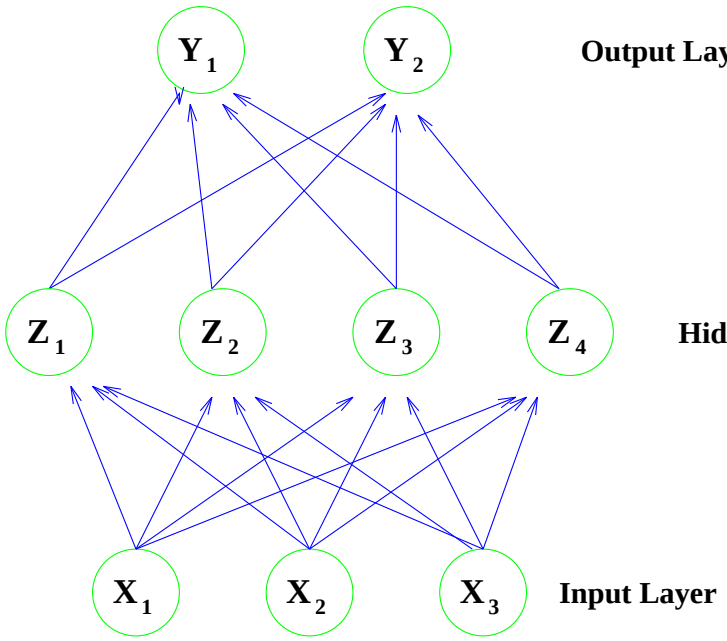
Each of the functions $f_j(x_j)$ (one for each input variable), can be a *smooth function* (ala kernel estimate), *linear*, or *omitted*.



GAM fit to SPAM data

- Shown are the most important predictors.
- Many show nonlinear behavior.
- Overall error rate 5.3% .
- Functions can be re-parametrized (e.g. log terms, quadratic, step-functions), and then fit by linear model.
- Produces a prediction per email $\Pr(\text{SPAM}|X = x)$

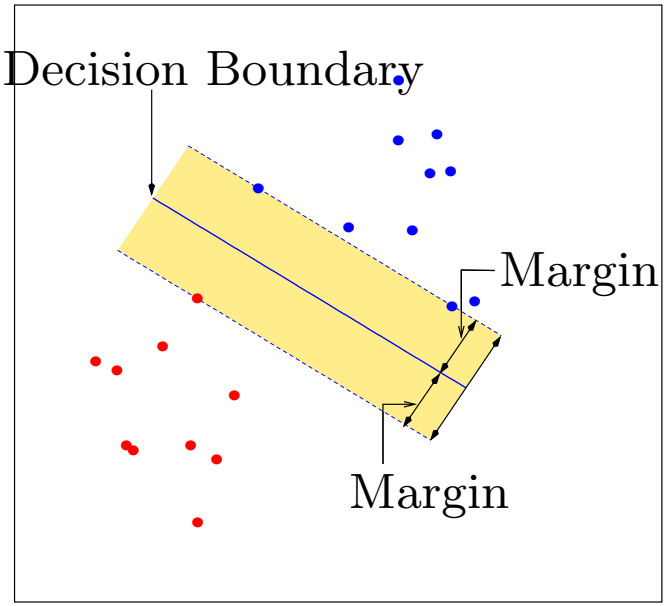
Neural Networks



Single (Hidden) Layer Perceptron

- Like a complex regression or logistic regression model — more flexible, but less interpretable — a *"black box"*.
- Hidden units $Z_1, Z_2, \ldots, Z_m$ (4 here):
  $Z_j = \sigma(\alpha_{0j} + \alpha_j^T X)$
  $\sigma(Z) = e^Z/(1+e^Z)$ is the logistic sigmoid *activation* function.
- Output is a linear regression or logistic regression model in the Z_j.
- Complexity controlled by m, ridge regularization, and *early stopping* of the *backpropogation algorithm* for fitting the neural network.

Support Vector Machines



- Maximize the gap (margin) between the two classes on the training data.
- If not separable
  - enlarge the feature space via basis expansions (e.g. polynomials).
  - use a *"soft"* margin (allow limited overlap).
- Solution depends on a small number of points (*"support vectors"*) — 3 here.

Decision boundary $x^T\beta + \beta_0 = 0$ is linear in the features (but could be nonlinear in original variables).

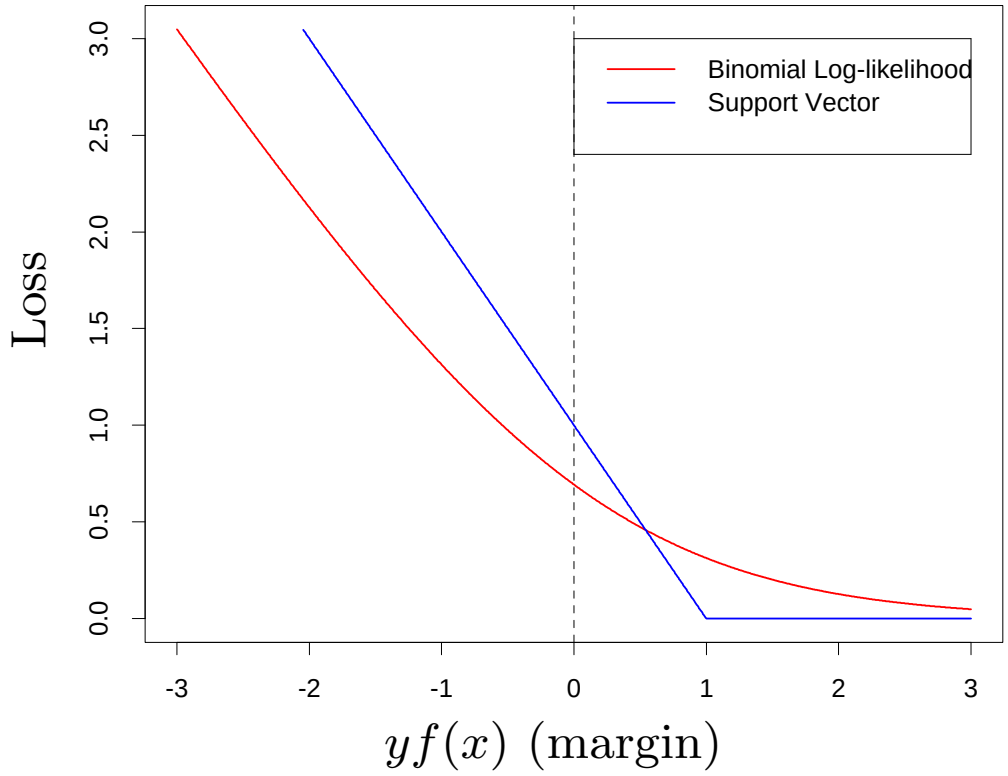
Properties of SVMs

- Primarily used for classification problems. Builds a linear classifier $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p$.
  If $f(X) > 0$, classify as +1, else if $f(X) < 0$, classify as -1.
- Generalizations use kernels (*"radial basis functions"*):

$$f(X) = \alpha_0 + \sum_{i=1}^{N} \alpha_i K(X, x_i)$$

  - K is a symmetric function, e.g. $K(X, x_i) = e^{-\gamma ||X - x_i||^2}$, and each x_i is one of the samples (vectors)
  - Many of the $\alpha_i = 0$; the rest are *"support points"*.
- Extensions to regression, logistic regression, PCA, ...
- Well developed mathematics — function estimation in *Reproducing Kernel Hilbert Spaces*.

SVM via Loss + Penalty



With $f(x) = x^T\beta + \beta_0$ and $y_i \in \{-1, 1\}$, consider

$$\min_{\beta_0,\,\beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2$$

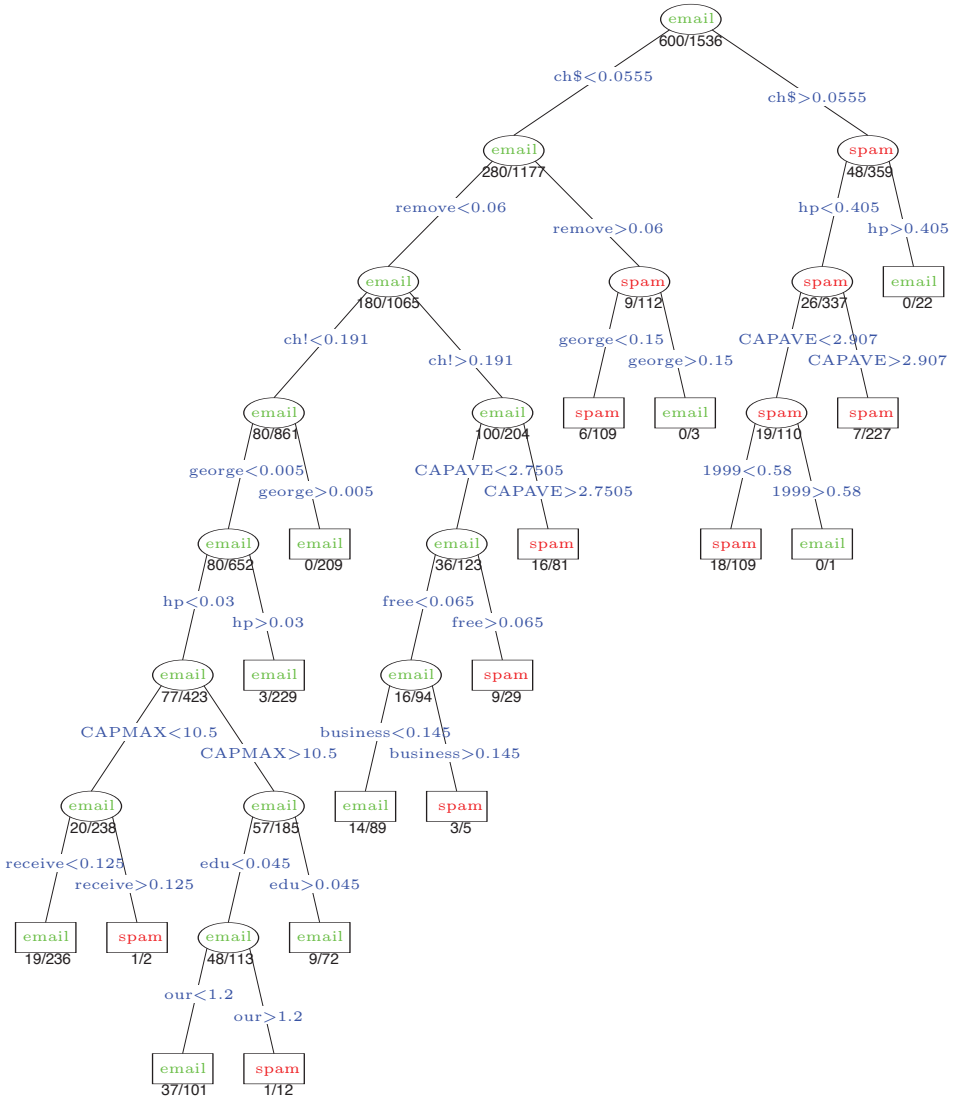
This *hinge loss* criterion is equivalent to the SVM, with λ monotone in B. Compare with

$$\min_{\beta_0,\,\beta} \sum_{i=1}^{N} \log\left[1 + e^{-y_i f(x_i)}\right] + \frac{\lambda}{2}\|\beta\|^2$$

This is *binomial deviance loss*, and the solution is "ridged" linear logistic regression.

Classification and Regression Trees

- ✔ Can handle huge datasets
- ✔ Can handle *mixed* predictors—quantitative and qualitative
- ✔ Easily ignore redundant variables
- ✔ Handle missing data elegantly
- ✔ Small trees are easy to interpret
- ✘ large trees are hard to interpret
- ✘ Often prediction performance is poor



Tree fit to the SPAM data

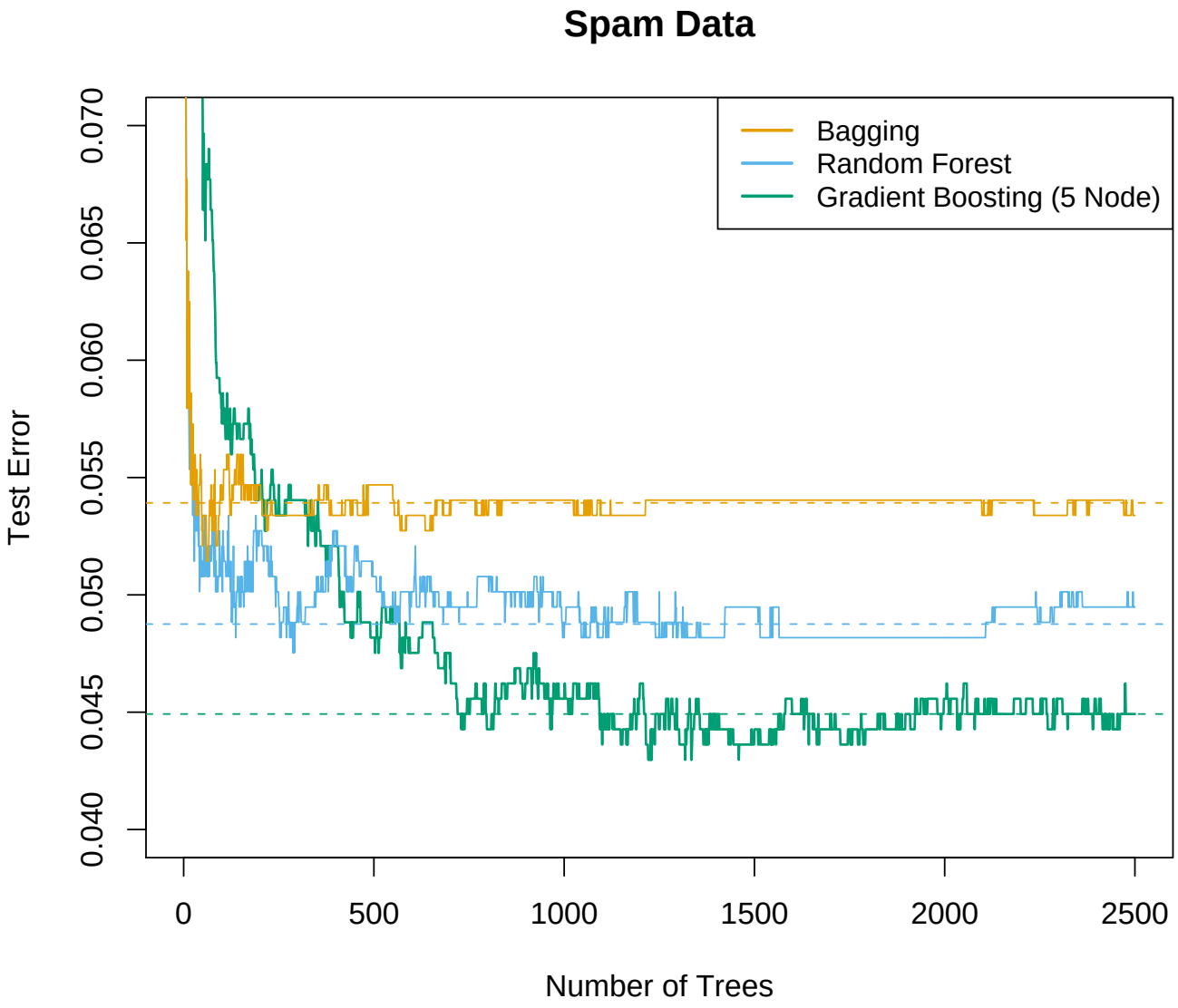
Misclassification error 8.7%

Ensemble Methods and Boosting

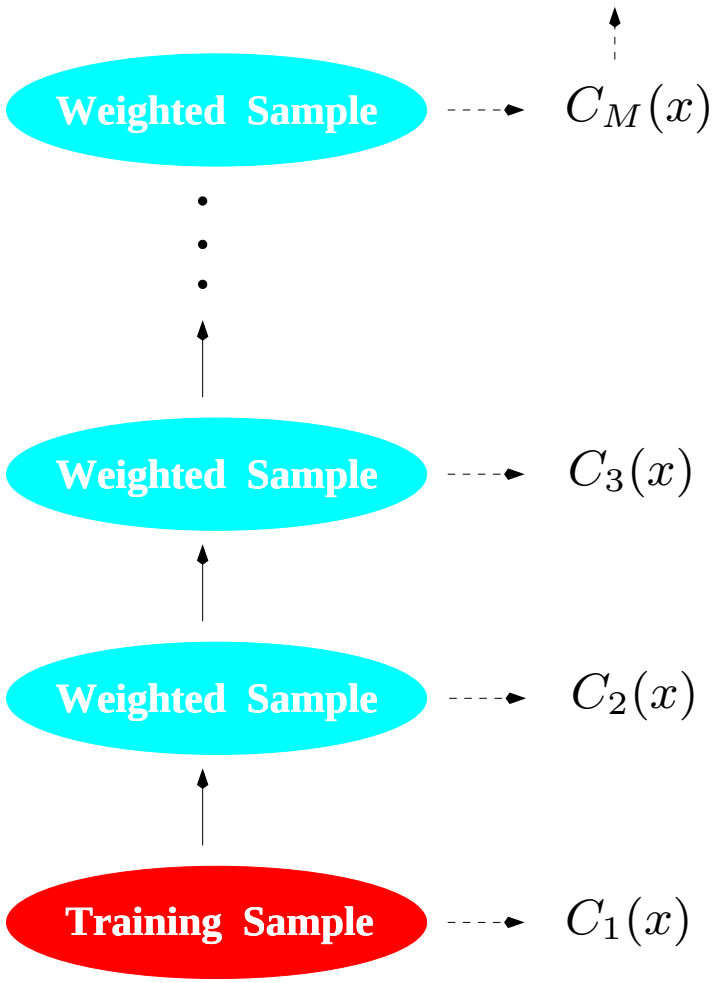
Classification trees can be simple, but often produce noisy (bushy) or weak (stunted) classifiers.

- *Bagging (Breiman, 1996):* Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.
- *Random Forests (Breiman 1999):* Improvements over bagging.
- *Boosting (Freund & Shapire, 1996):* Fit many smallish trees to *reweighted* versions of the training data. Classify by weighted majority vote.

In general *Boosting* $\succ$ *Random Forests* $\succ$ *Bagging* $\succ$ *Single Tree*.



Boosting



- Average many trees, each grown to re-weighted versions of the training data.
- Weighting *decorrelates* the trees, by focussing on regions missed by past trees.
- Final Classifier is weighted average of classifiers:

$$C(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m C_m(x)\right]$$

Modern Gradient Boosting (Friedman, 2001)

- Fits an additive model

$$F_m(X) = T_1(X) + T_2(X) + T_3(X) + \ldots + T_m(X)$$

  where each of the $T_j(X)$ is a *tree in* X.

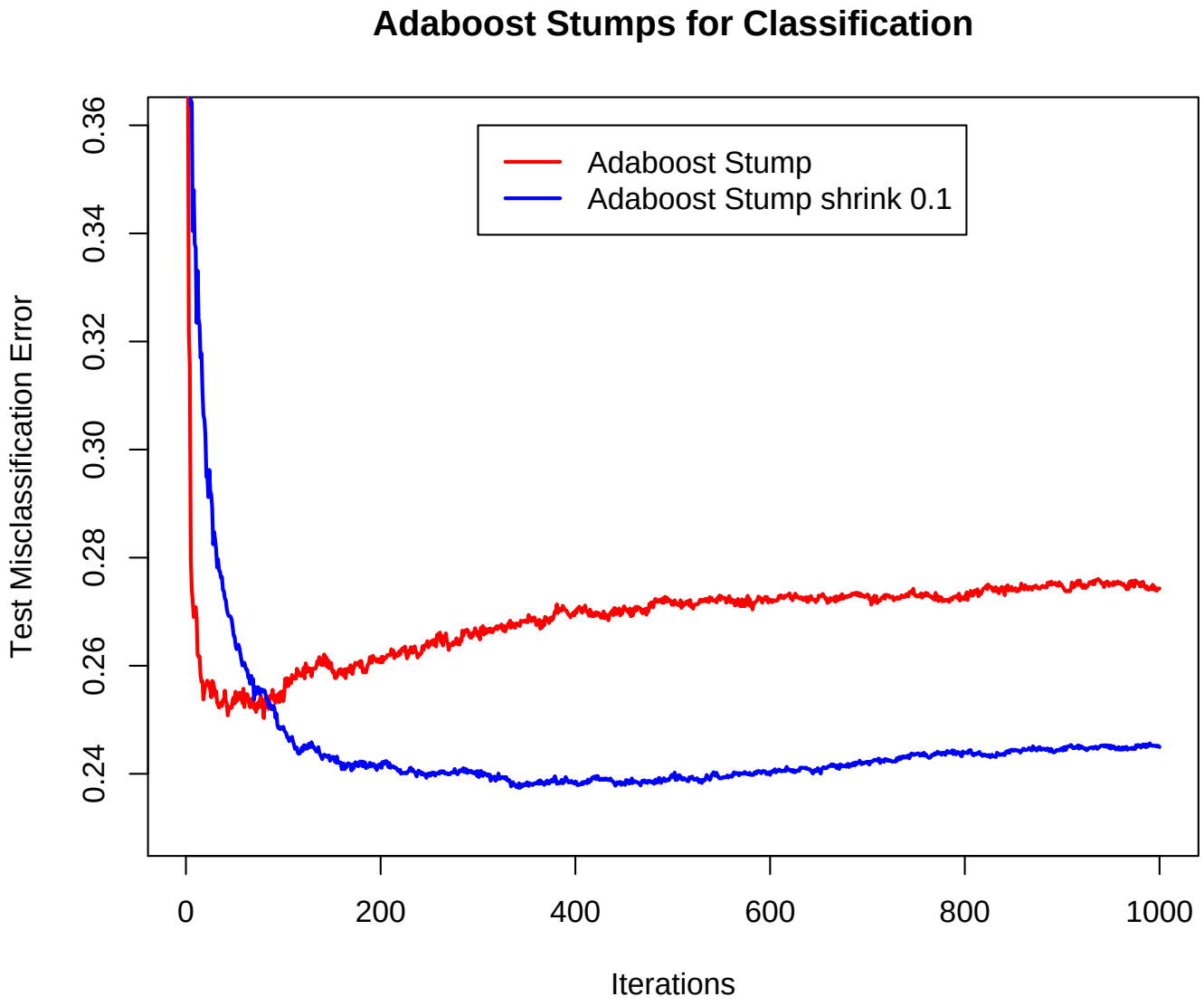
- Can be used for regression, logistic regression and more. For example, gradient boosting for regression works by repeatedly fitting trees to the residuals:
  1. Fit a small tree $T_1(X)$ to Y.
  2. Fit a small tree $T_2(X)$ to the residual $Y - T_1(X)$.
  3. Fit a small tree $T_3(X)$ to the residual $Y - T_1(X) - T_2(X)$.

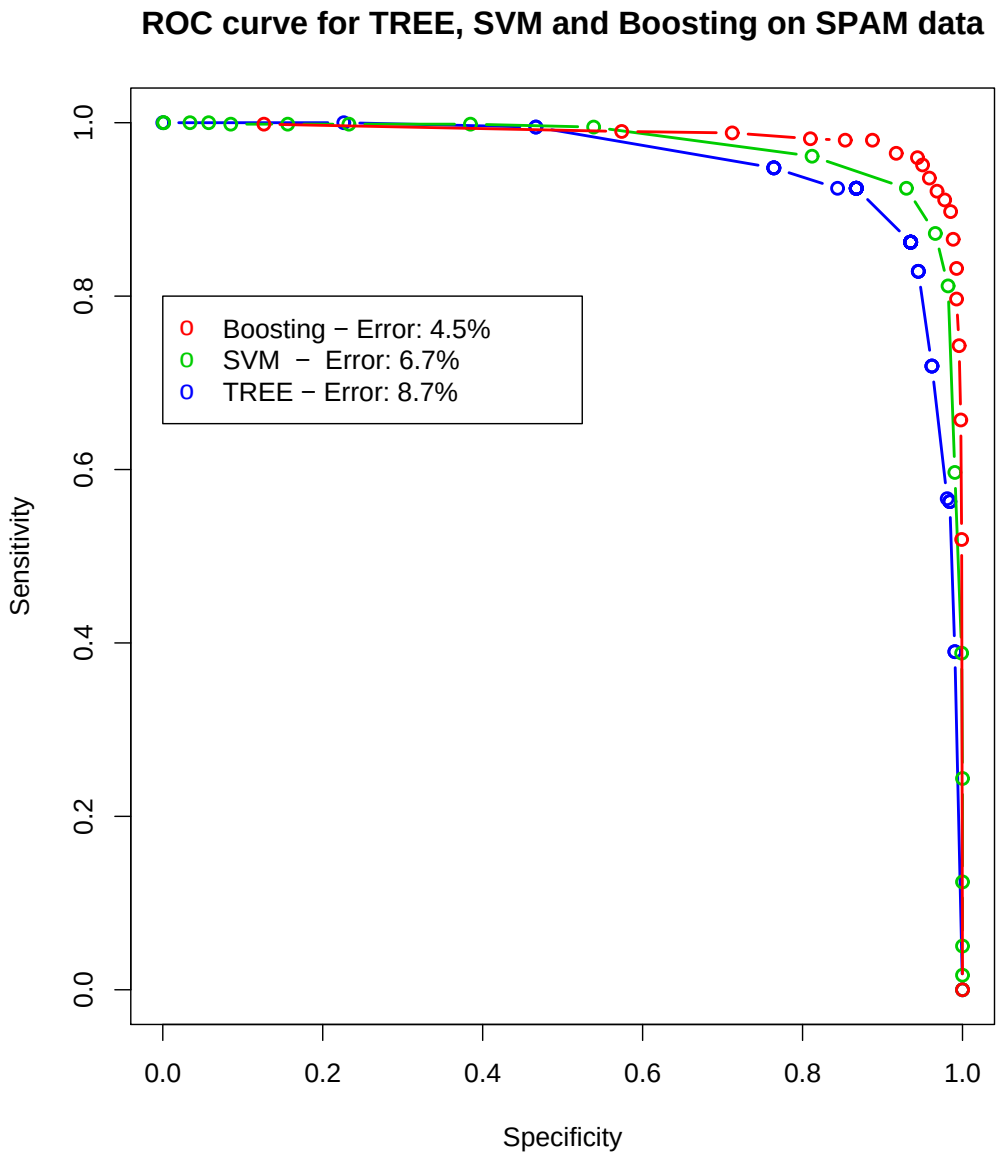
     and so on.

- m is the tuning parameter, which must be chosen using a validation set (m too big will overfit).

Gradient Boosting - Details

- For general loss function $L[Y, F_m(X) + T_{m+1}(X)]$, fit a tree to the *gradient* $\partial L/\partial F_m$ rather than residual.
- Shrink the new contribution before adding into the model: $F_m(X) + \gamma T_{m+1}(X)$. This slows the forward stagewise algorithm, leading to improved performence.
- Tree *depth* determines interaction order of the model.
- Boosting will eventually overfit; number of terms m is a tuning parameter.
- As $\gamma \downarrow 0$, boosting path behaves like ℓ_1 regularization path in the space of trees.





Boosting on SPAM

Boosting dominates all other methods on SPAM data — *4.5% test error.*

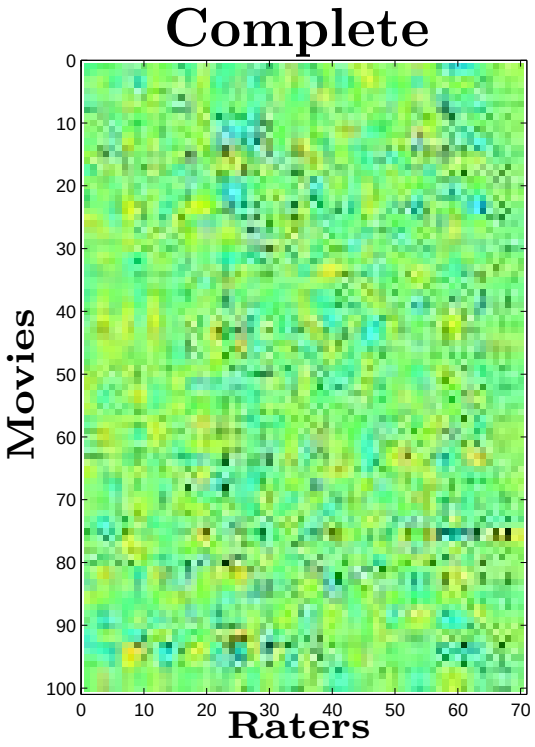
Used 1000 trees (depth 6) with default settings for `gbm` package in `R`.

ROC curve obtained by varying the *threshold* of the classifier.

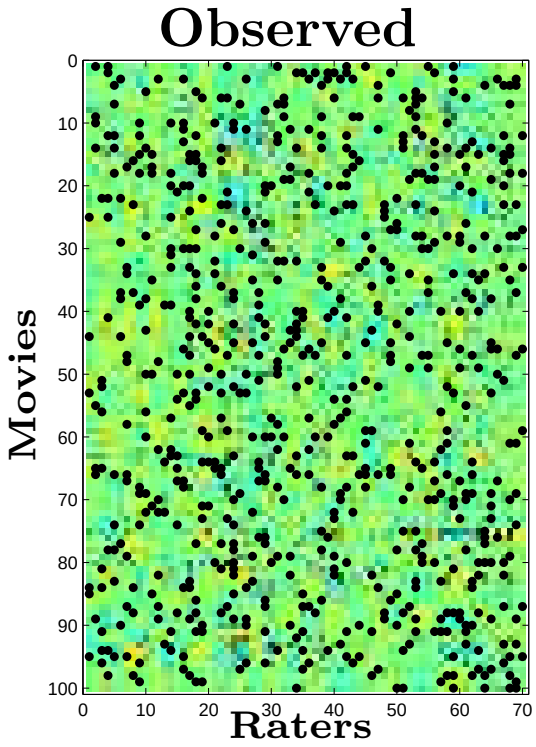
Sensitivity: proportion of true spam identified

Specificity: proportion of true email identified.

Matrix Completion



Example: Netflix problem. We partially observe a matrix of movie ratings (rows) by a number of raters (columns). The goal is to predict the future ratings of these same individuals for movies they have not yet rated (or seen).



We solve this problem by fitting an ℓ_1 regularized SVD path to the observed data matrix (Mazumder, Hastie and Tibshirani, 2009).

ℓ_1 regularized SVD

$$\min_{\hat{\mathbf{X}}} ||P_\Omega(\mathbf{X}) - P_\Omega(\hat{\mathbf{X}})||_F^2 + \lambda||\hat{\mathbf{X}}||_*$$

- P_Ω is projection onto observed values (sets unobserved to 0).
- $||\hat{\mathbf{X}}||_*$ is *nuclear norm* — sum of singular values.
- This is a convex optimization problem (Candes 2008), with solution given by a *soft thresholded* SVD — singular values are shrunk toward zero, many set to zero.
- Our algorithm iteratively soft-thresholds the SVD of

$$\begin{aligned} P_\Omega(\mathbf{X}) + P_\Omega^\perp(\hat{\mathbf{X}}) &= \left\{P_\Omega(\mathbf{X}) - P_\Omega(\hat{\mathbf{X}})\right\} + \hat{\mathbf{X}} \\ &= \text{Sparse + Low-Rank} \end{aligned}$$

- Using Lanczos techniques and warm starts, we can efficiently compute solution paths for very large matrices (50K $\times$50K)

Software

- *R* is free software for statistical modeling, graphics and a general programming environment. Works on PCs, Macs and Linux/Unix platforms. All the models here can be fit in R. R grew from its predecessor Splus, and both implement the S language developed at Bell Labs in the 80s.
- *SAS* and their *Enterprise Miner* can fit some of the models mentioned in this talk, with good data-handling capabilities, and high-end user interfaces.
- *Salford Systems* has commercial versions of trees, random forests and gradient boosting.
- SVM software is all over, but beware of patent infringements if put to commercial use.
- Many free versions of neural network software; *Google* will find.