Classification

May, 2019
Columbia University

Classification

- Regression involves predicting a continuous-valued response.
- Classification involves predicting a categorical / qualitative response:
  - Cancer versus Normal
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3
- Classification problems tend to occur even more frequently than regression problems in biomedical applications.
- Just like regression,
  - Classification cannot be blindly performed in high-dimensions because you will get zero training error but awful test error;
  - Properly estimating the test error is crucial; and
  - There are a few tricks to extend classical classification approaches to high-dimensions, which we have already seen in the regression context!

Classification

- Categorical / qualitative variables take values in an unordered set: e.g.
  eye color $\in \{\textit{brown}, \textit{blue}, \textit{green}\}$
  email $\in \{\textit{spam}, \textit{not spam}\}$.
- We want to build a function that takes as input the feature vector X and predicts the value for Y.
- Often we are more interested in estimating the probability that X belongs to a given category.
- For example: we might want to know the probability that someone will develop diabetes, rather than to predict whether or not they will develop diabetes.

Can't We Just Use Linear Regression?

- Classify an emergency room patient on the basis of her symptoms to one of three conditions:

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

- If we apply linear regression, then the results will depend on the choice of coding . . . and the coding implies an ordering among the medical conditions.
- A classification approach is more appropriate.

For binary outcome, could use "regression", but may get better estimates using classification methods.

Classification

For now, suppose we have binary outcome

(eg. tx response/survival-past-landmark-time)

Sometimes interested in predicting most-likely class...

May be interested in probability estimates

Many ideas similar to those from continuous outcome

Most importantly, overfitting is still an issue!!!

Classification

- There are many approaches for performing classification.
- We will discuss three important contemporary methods, logistic regression, support vector machines, and K-nearest neighbors.

Logistic Regression

Here we model each $y_i \in \{0, 1\}$ as a coin flip...

with success probability

$$p_i = \frac{e^{x_i^\top \beta}}{1 + e^{x_i^\top \beta}}$$

Or equivalently

$$\log\left(\frac{p_i}{1 - p_i}\right) = x_i^\top \beta$$

and want to find β so that...

- for succeses, have $p_i \sim 1$
- for failures, have $p_i \sim 0$

Logistic Regression

Just a different Loss function for measuring goodness-of-fit!

In linear regression we tried to minimize

$$\sum_{i=1}^{n} \ell\left(y_i, x_i^\top \beta\right)$$

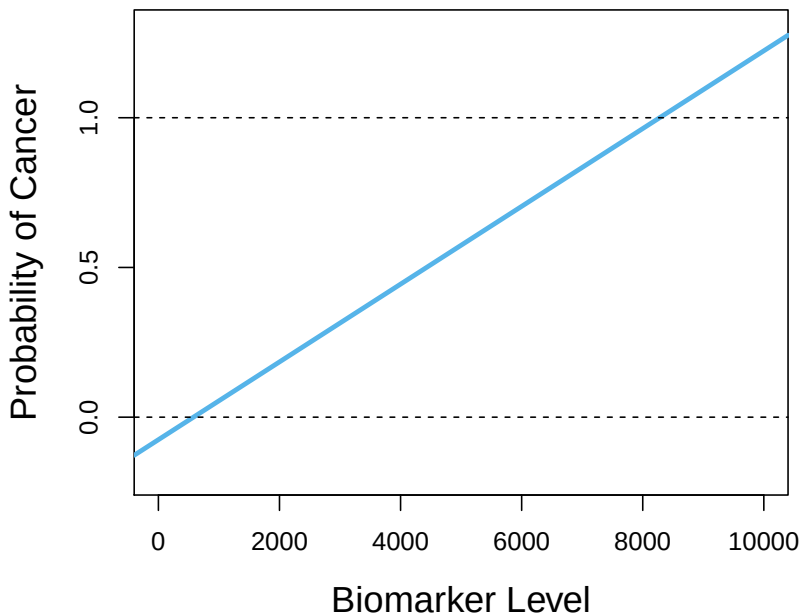
with

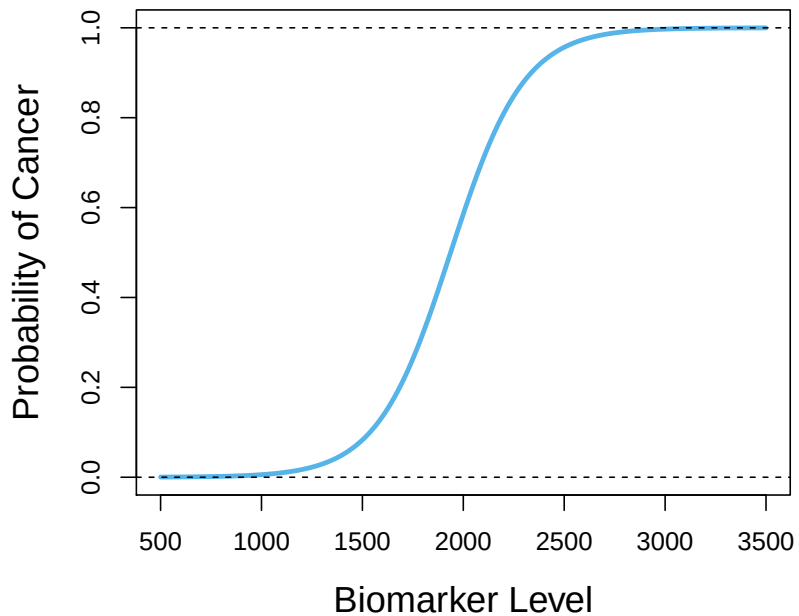
$$\ell\left(y, \eta\right) = \left(y - \eta\right)^2$$

Logistic regression just uses a different goodness-of-fit loss

$$\ell\left(y, \eta\right) = -y\eta + \log\left(1 - e^{\eta}\right)$$

Why Not Linear Regression?





- Left: linear regression.
- Right: logistic regression.

Multinomial Logistic Regression

- Suppose we have outcomes from K classes
- We now try to find vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$
- Where we model the probability of falling in class j as

$$p(y = j|x) = \frac{e^{\boldsymbol{\beta}_j^\top x}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_j^\top x}}$$

- As with logistic regression, this leads to a natural loss function to minimize to estimate $\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_K$

Ways to Extend Logistic Regression to High Dimensions

1. Variable Pre-Selection
2. Ridge Logistic Regression
3. Lasso Logistic Regression

How to decide which approach is best, and which tuning parameter value to use for each approach? Cross-validation or validation set approach.

Logistic Regression Extensions - I

We noted that logistic regression solves

$$\hat{\beta} \leftarrow \text{argmin}_\beta \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, x_t = i^\top \beta \right)$$

where

$$\ell (y, \eta) = -y\eta + \log (1 - e^\eta)$$

The lasso modifies things to

$$\hat{\beta} \leftarrow \text{argmin}_\beta \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, x_i^\top \beta \right) + \lambda \|\beta\|_1$$

Logistic Regression Extensions - II

Additionally, more flexible regression estimates (eg. splines), can be constructed with "feature engineering":

There, we predefine $Z(x)$ as some multivariate transformation of x, and solve

$$\hat{\beta} \leftarrow \text{argmin}_\beta \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, Z(x_i)^\top \beta \right)$$

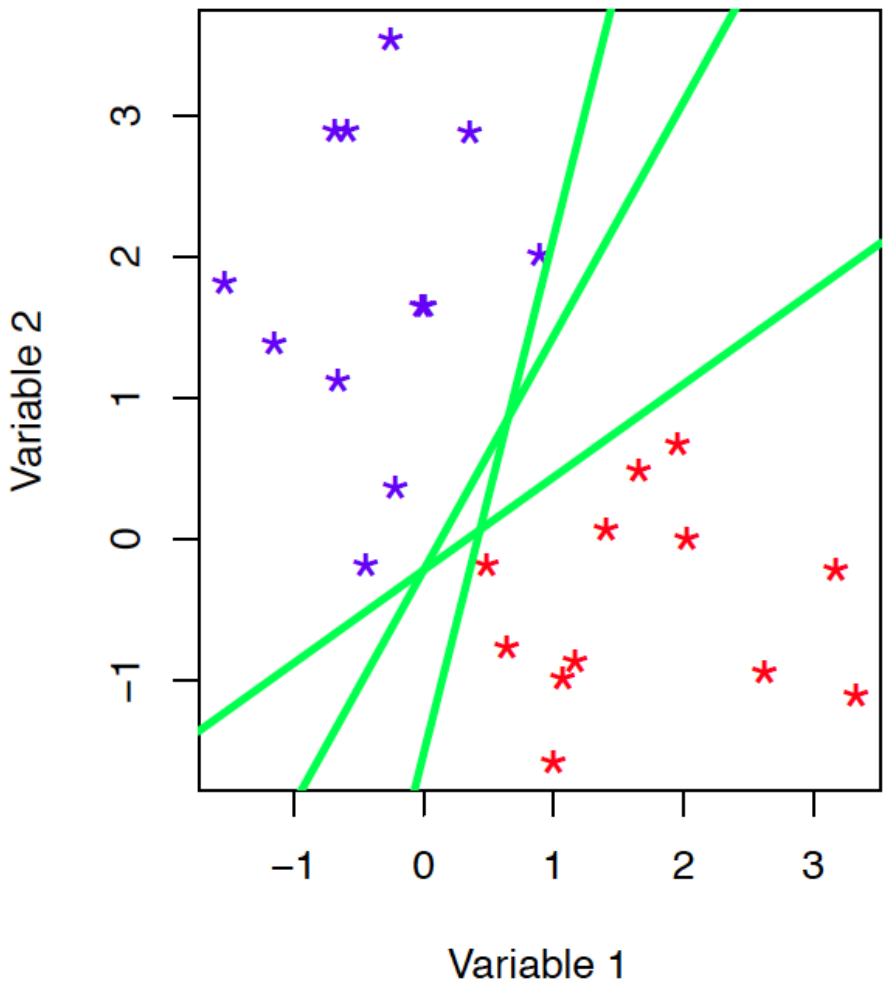
Here we use the logistic loss, as opposed to least squares.

This same idea holds quite broadly!

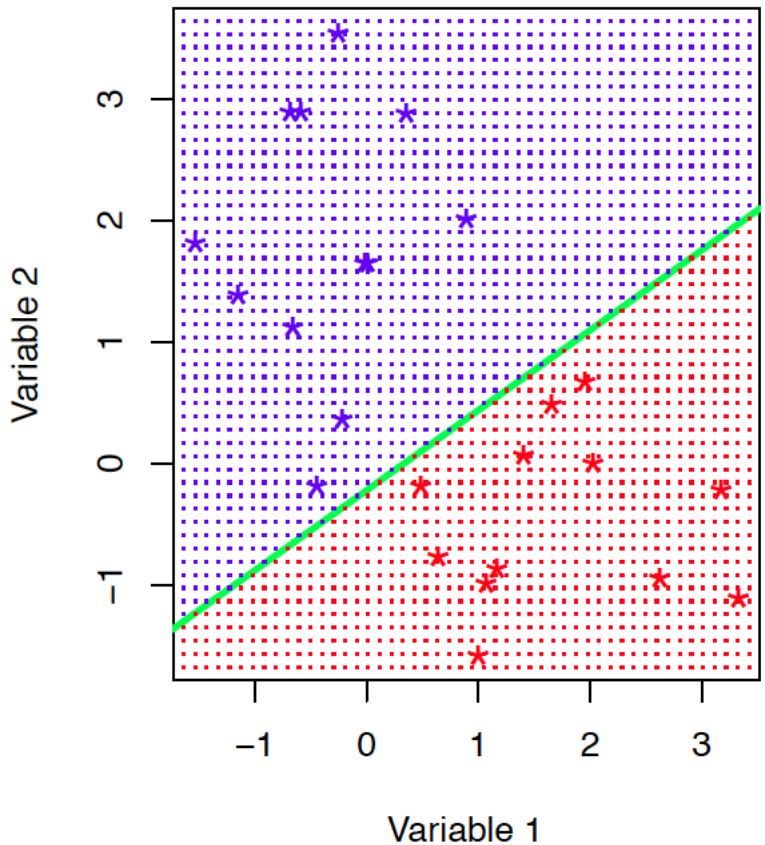
Support Vector Machines

- Developed in around 1995.
- Claimed to "overcome the curse of dimensionality."
- Does not (automatically) overcome the curse of dimensionality!
- Fundamentally and numerically very similar to logistic regression.
- But, it is a nice idea.

Separating Hyperplane

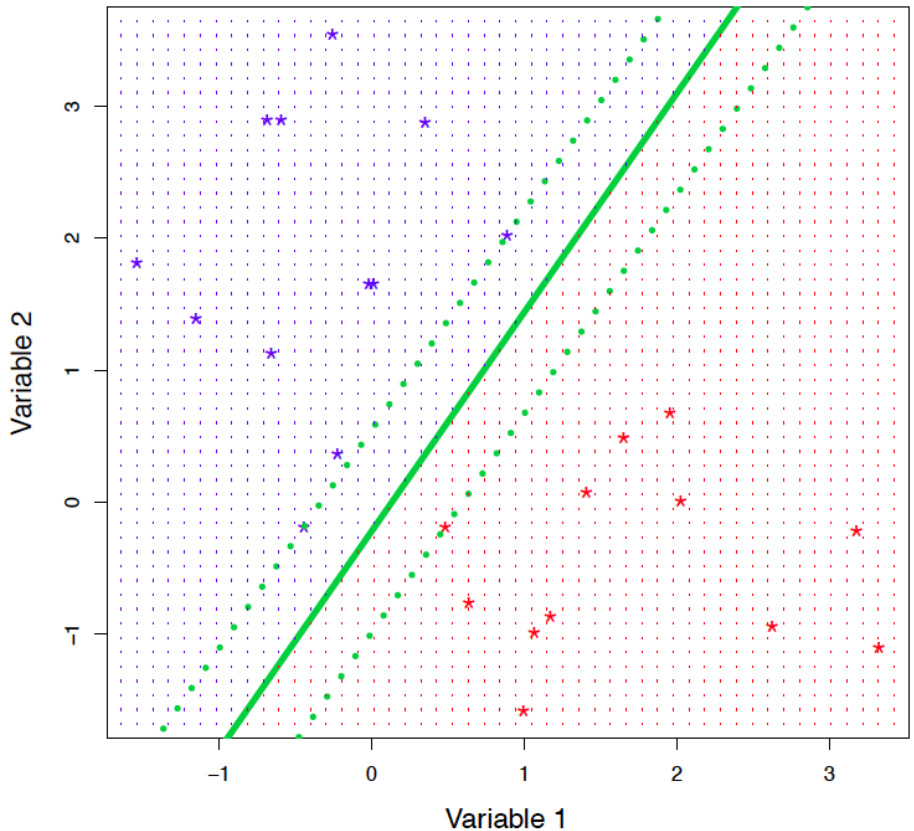


Classification Via a Separating Hyperplane



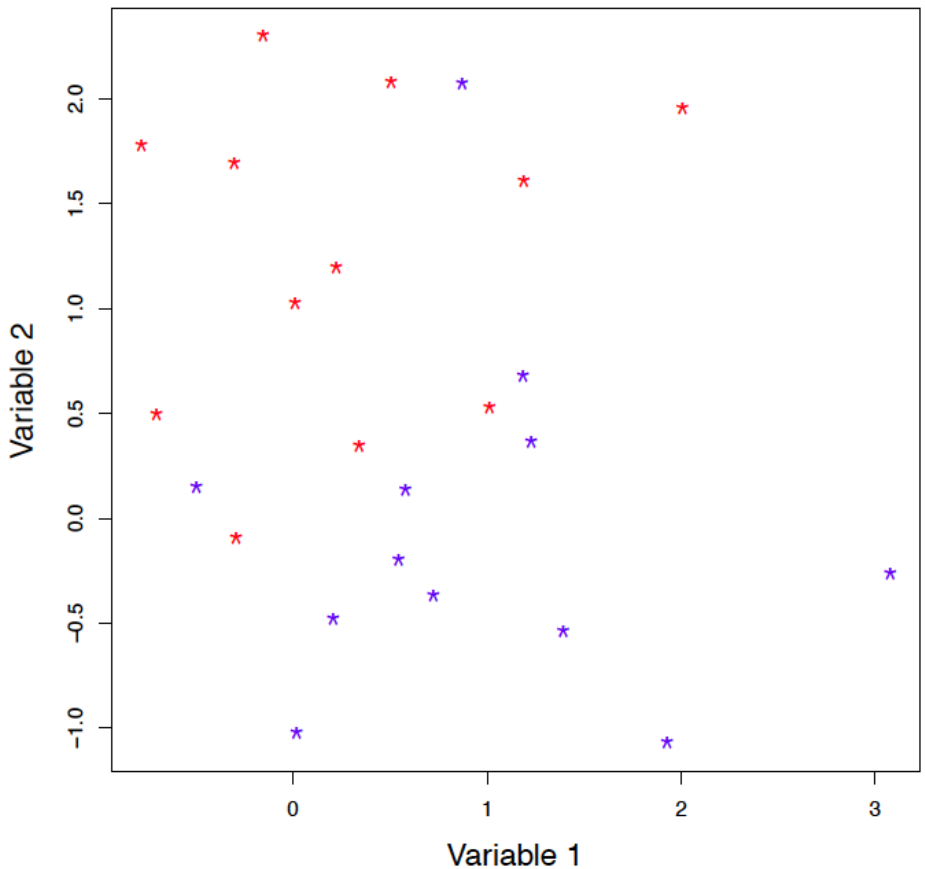
Blue class if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > c$; red class otherwise.

Maximal Separating Hyperplane

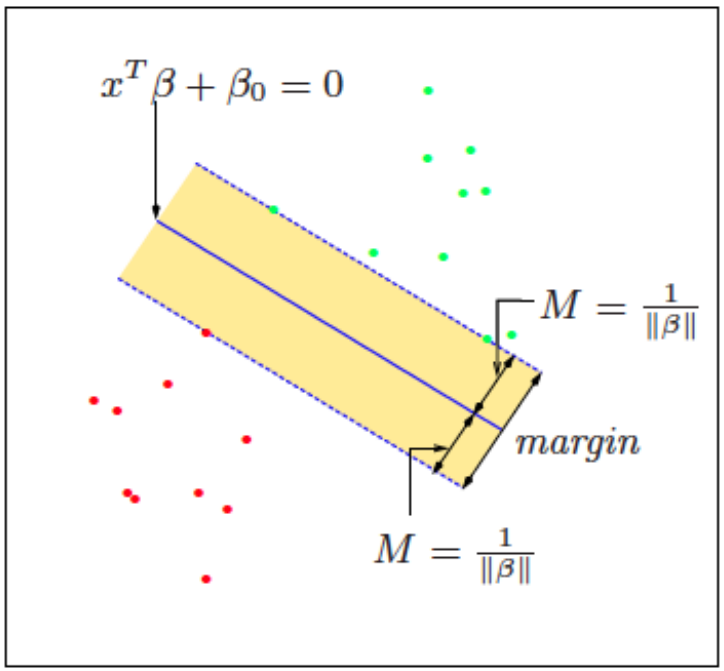


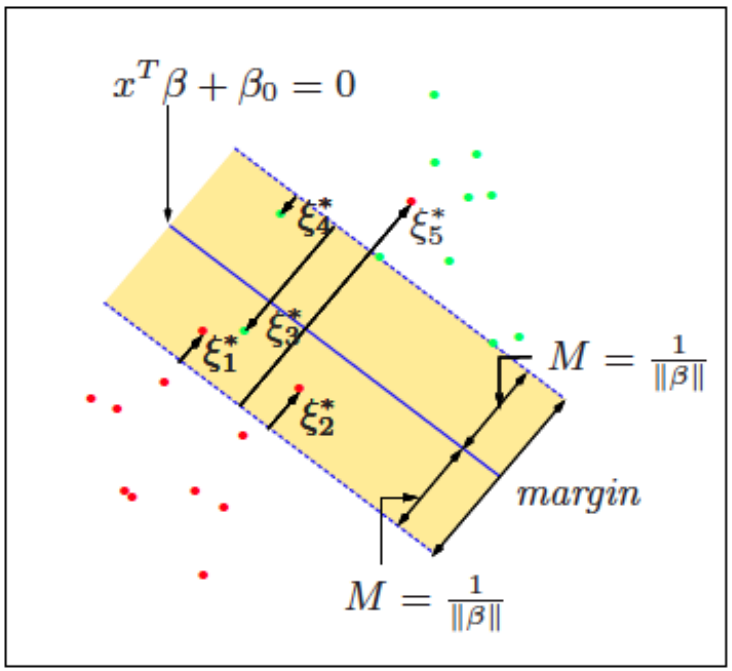
Note that only a few observations are on the margin: these are the support vectors.

What if There is No Separating Hyperplane?



Support Vector Classifier: Allow for Violations

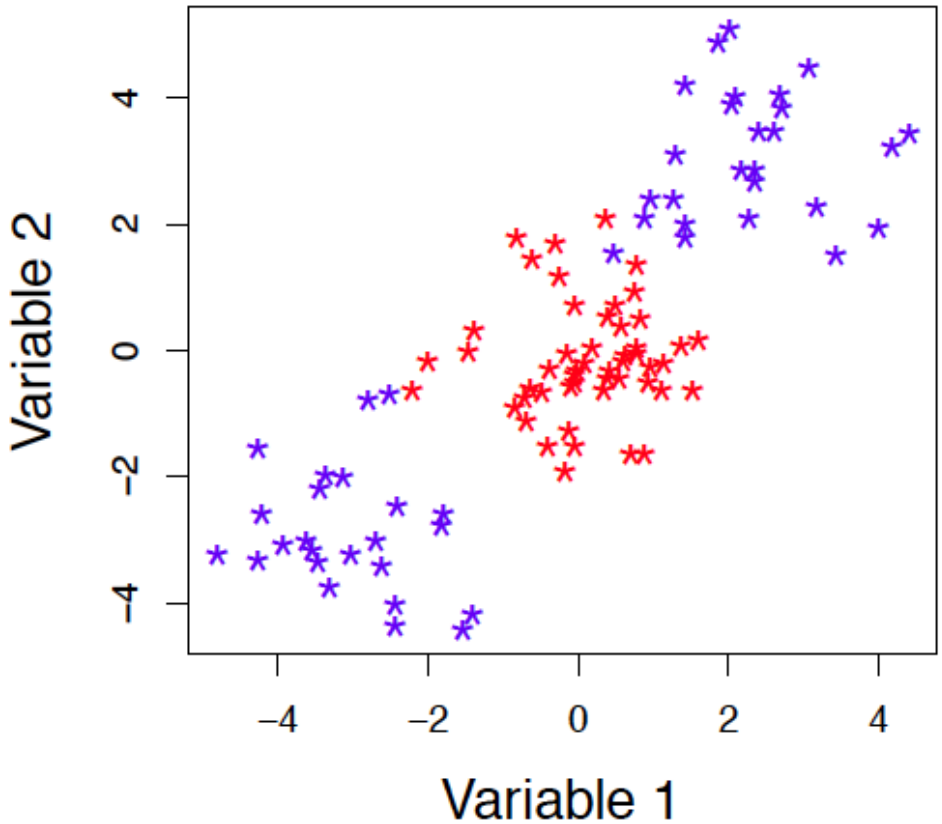




Support Vector Machine

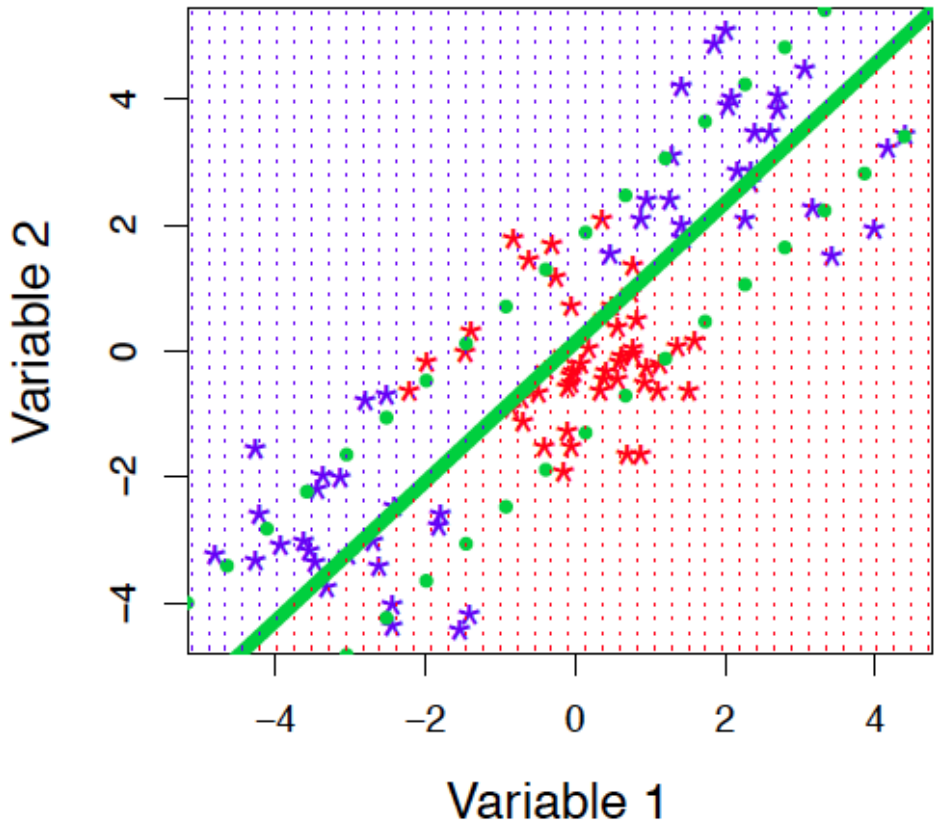
- The support vector machine is just like the support vector classifier, but it elegantly allows for non-linear expansions of the variables: "non-linear kernels".
- However, linear regression, logistic regression, and other classical statistical approaches can also be applied to non-linear functions of the variables.
- For historical reasons, SVMs are more frequently used with non-linear expansions as compared to other statistical approaches.

Non-Linear Class Structure



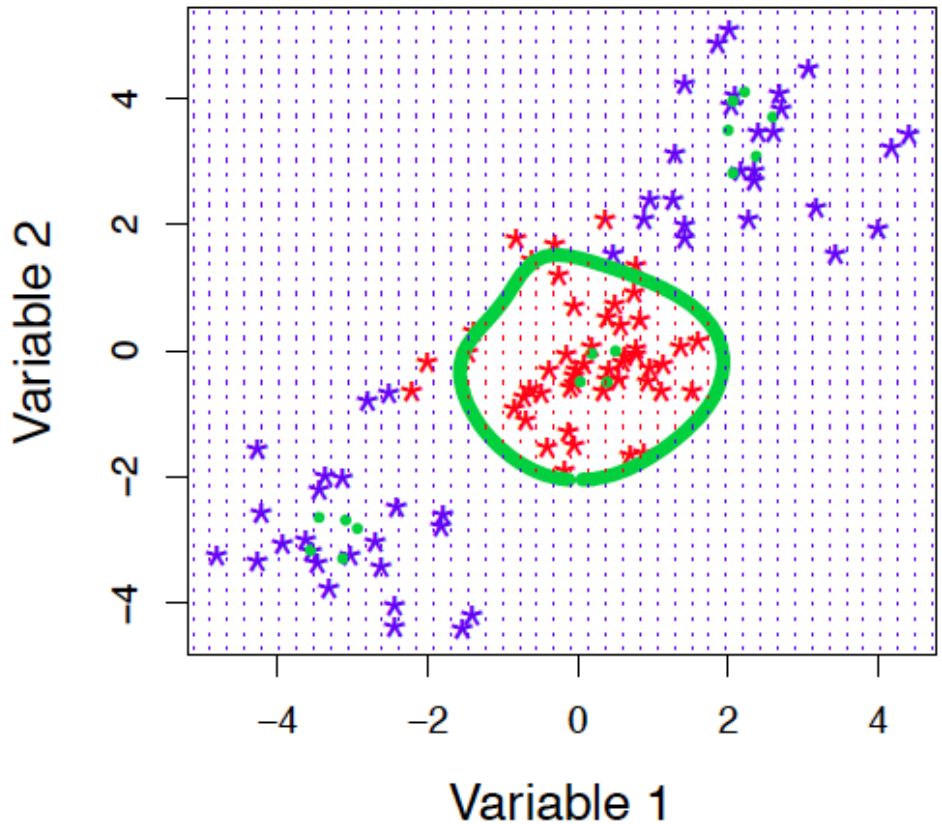
This will be hard for a linear classifier!

Try a Support Vector Classifier



Uh-oh!!

Support Vector Machine



Much Better.

Non-Linear Kernels

How do we think about "non-linear kernels"?

- Like with "basis expansion" idea; we create additional features from our original features.
- We then use support vector classifier in this higher dimensional space
- This allows for non-linear decision boundaries

Mostly a computational trick for doing this

Is A Non-Linear Kernel Better?

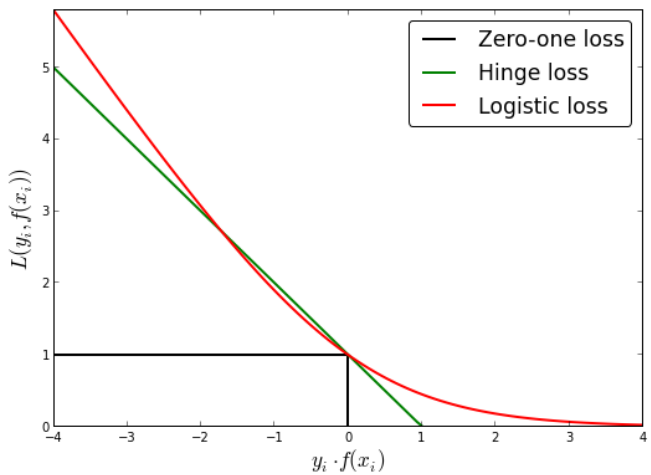
- Yes, if the true decision boundary between the classes is non-linear, and you have enough observations (relative to the number of features) to accurately estimate the decision boundary.
- No, if you are in a very high-dimensional setting such that estimating a non-linear decision boundary is hopeless.

SVM vs Other Classification Methods

- The main difference between SVM and logistic regression is the loss function used to assess the "fit":

$$\sum_{i=1}^{n} L(f(x_i), y_i)$$

- Zero-one loss: $I(f(x_i) = y_i)$, where $I()$ is the indicator function. Not continuous, so hard to work with!!
- Hinge loss: $\max(0, 1 - f(x_i)y_i)$
- Logistic loss: $\log\left(1 + e^{f(x_i)y_i}\right)$ [symmetric formulation]



SVM vs Logistic Regression

- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".
- The "kernel trick" makes things computationally easier, but it does not remove the danger of overfitting.
- SVM uses a non-linear kernel... but could do that with logistic or linear regression too!
- A disadvantage of SVM (compared to, e.g. logistic regression) is that it does not provide a measure of uncertainty: cases are "classified" to belong to one of the two classes.

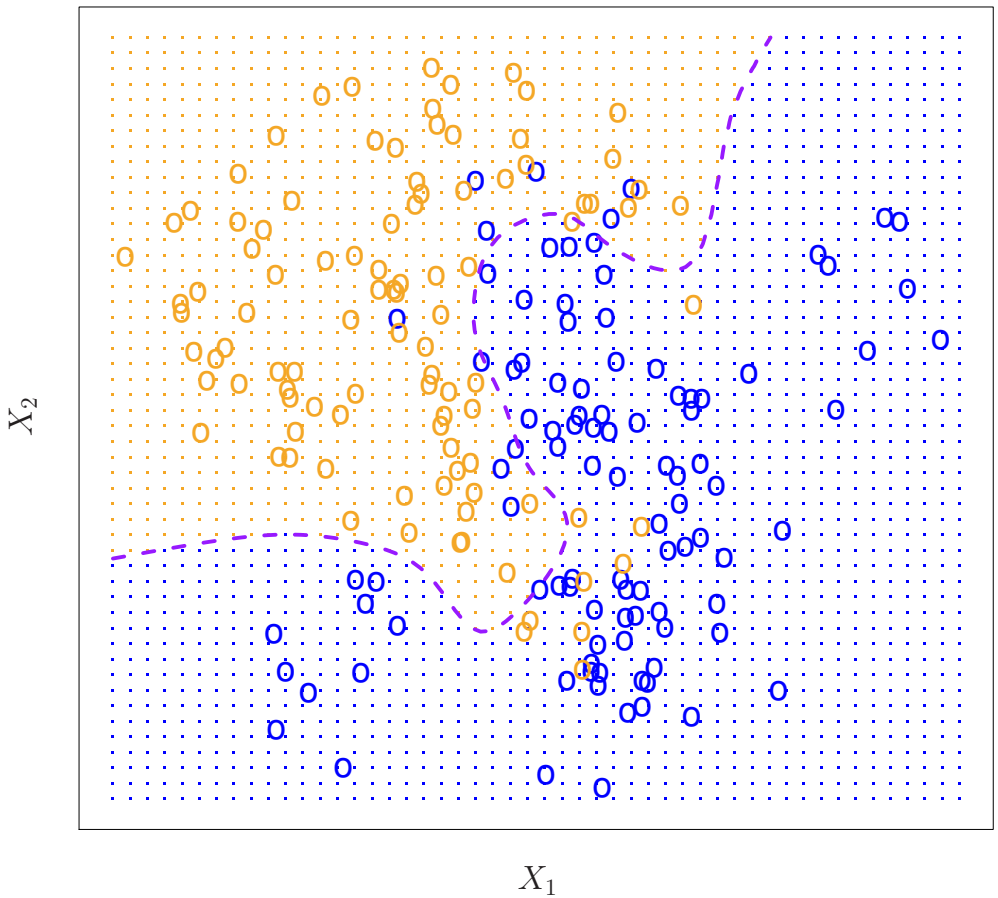
In High Dimensions...

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.
- This tuning parameter is like a ridge penalty, both mathematically and conceptually. The SVM decision rule involves all of the variables (the SVM problem can be written as a ridge problem but with the Hinge loss).
- Can get a sparse SVM using a lasso penalty; this yields a decision rule involving only a subset of the features.
- Logistic regression and other classical statistical approaches could be used with non-linear expansions of features. But this makes high-dimensionality issues worse.

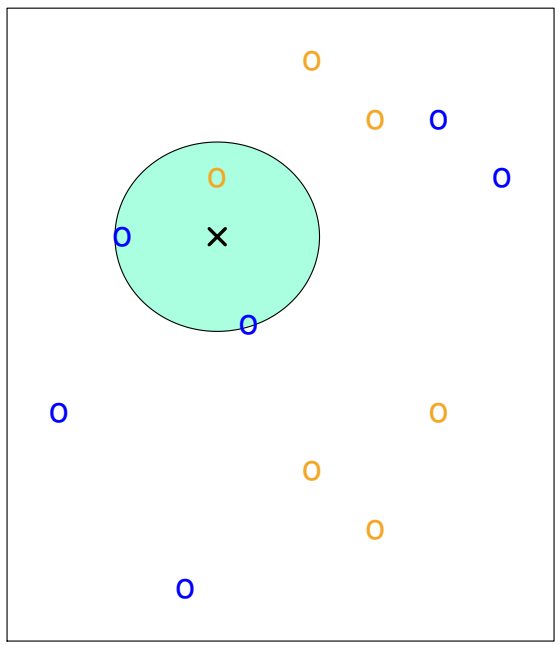
K-Nearest Neighbors

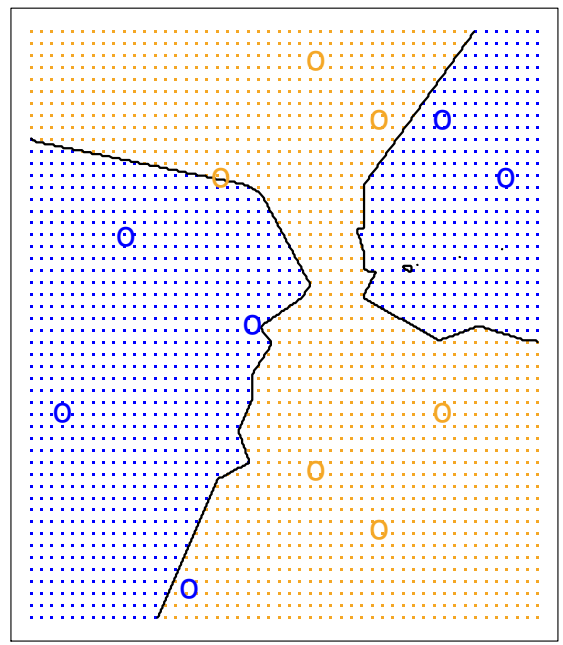
- Can I take a totally non-parametric (model-free) approach to classification?
- K-nearest neighbors:
  1. Identify the K observations whose X values are closest to the observation at which we want to make a prediction.
  2. Classify the observation of interest to the most frequent class label of those K nearest neighbors.

K-Nearest Neighbors

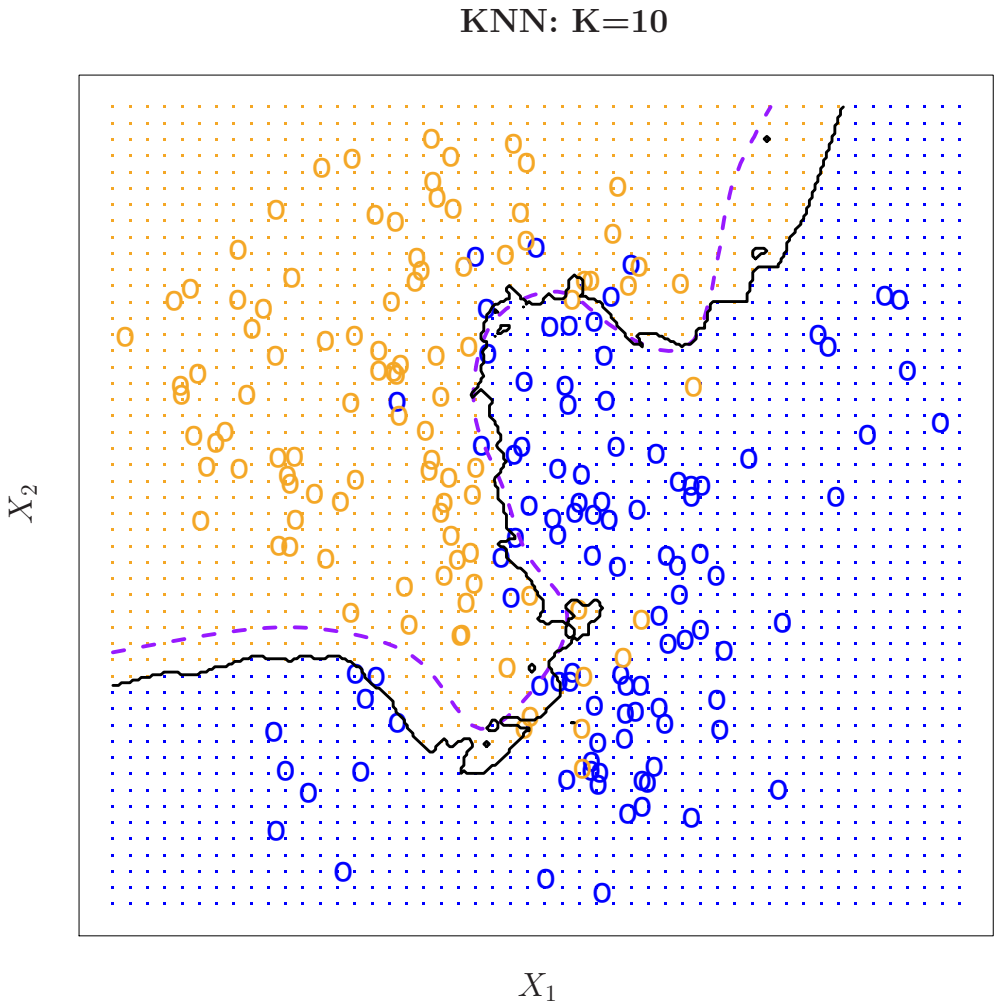


K-Nearest Neighbors



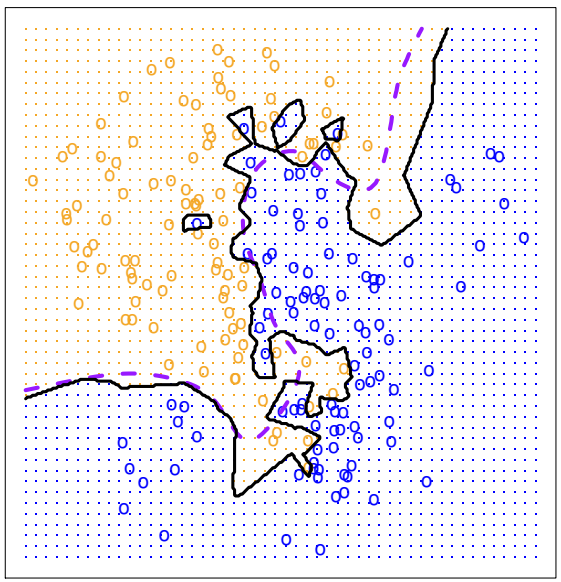


K-Nearest Neighbors

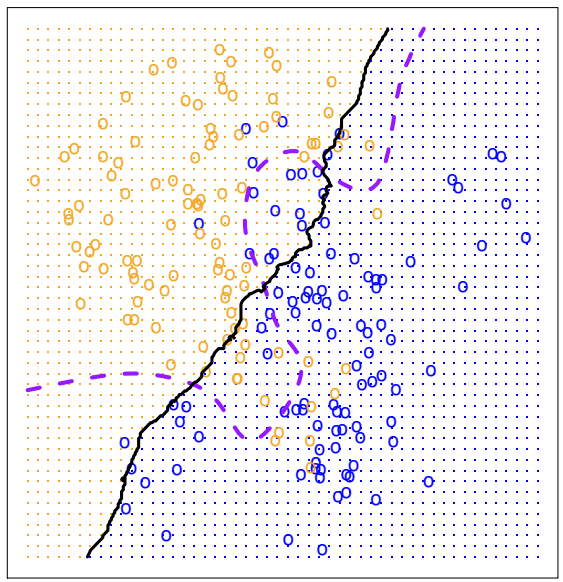


K-Nearest Neighbors

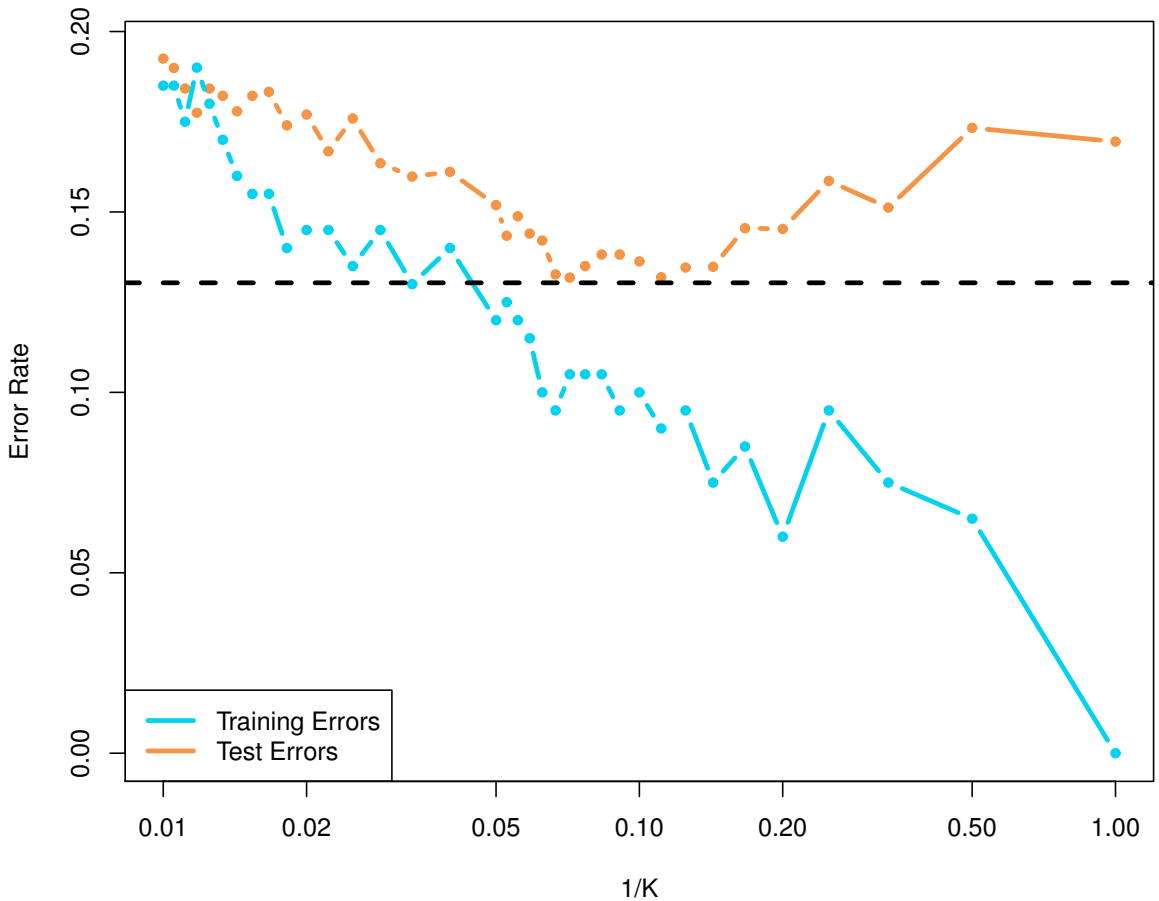
KNN: K=1



KNN: K=100



K-Nearest Neighbors



K-Nearest Neighbors

- Simple, intuitive, model-free.
- Good option when p is very small.
- Curse of dimensionality: when p is large, no neighbors are "near". All observations are close to the boundary.
- Do not use in high dimensions!

Assessing the Performance of Classifiers

Generally not assessed by squared error loss

Common choices include

- Misclassification Rate
- ROC-based metric (eg. AUC)
- Predictive (log)Likelihood

Test/Validation Performance of Classifiers

As with continuous outcome...

Must use separate observations for training and evaluation

Ideally want the same training/validation/test split

Cross-Validation ROC

When using CV, rather than getting an ROC curve per fold...

Build a model $\hat{f}_{-k}$ without fold k

and get scores $\hat{f}_{-k}(x_{k(i)})$ for all obs in fold k

---

After cycling through all folds, will have a complete set of n scores.

Use these scores and the observed outcomes to evaluate ROC

This idea is known as pre-validation; it can be used more generally

Calibration

Often want more from our model than

- *most likely class*
- or uninterpretable *score*

Would like a well calibrated probability

Calibration

If a doctor says...

"you have a 70% chance of responding to treatment" ...

What should that mean?

If it is well calibrated, then...

if a doctor said that to 100 patients, roughly 70 would respond.

Poor Calibration

Outputs from probabilistic models are rarely well calibrated...

Many algorithms actually only give scores (eg. SVM).

Identical method for

- "recalibrating", and
- turning scores into probabilities.

Evaluating Calibration

If we overfit, will tend to find predicted probs near 0 and 1...

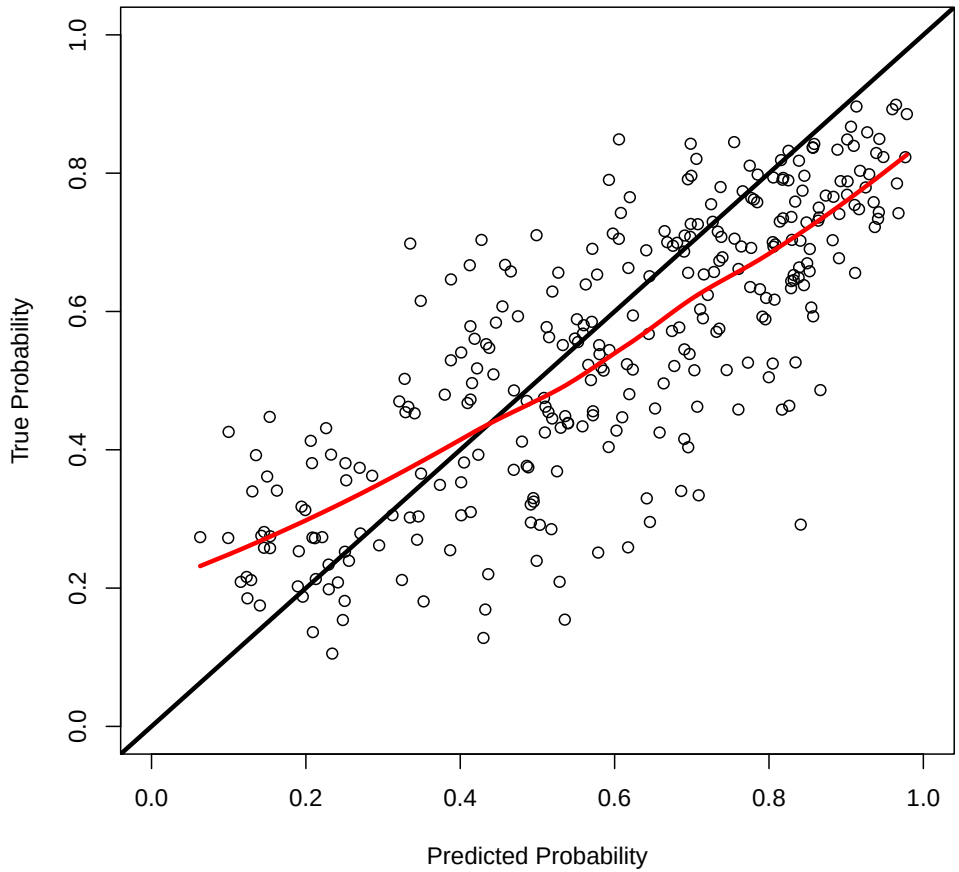
This can happen with eg. logistic regression even if misclassification error is good.

Can evaluate calibration by regressing y on predicted prob...

With good calibration, this should look like a straight line

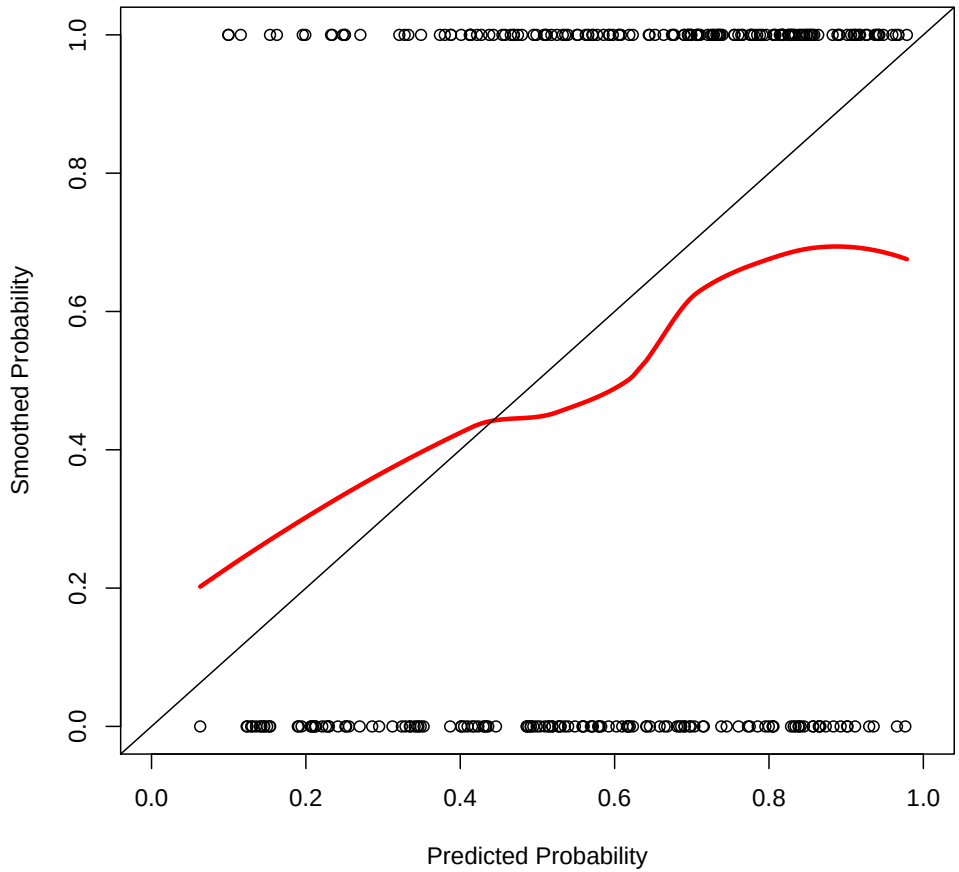
Poor Calibration - II

For example, evaluating logistic regression on test data...



Poor Calibration - III

However, we don't get to see the actual probs. Instead we see...



Turning Scores into probabilities

A score is just a derived feature...

uses response, so requires carefulness!

Suppose we have 2 datasets (with outcome measured).

Imagine we use the first to develop a score $S(x)$...

And would like to turn the score into a probability with the second.

Recalibration - I

Can calibrate a score by...

using regression of y onto $S(x)$ as a calibrating transformation

This should be done on the second dataset.

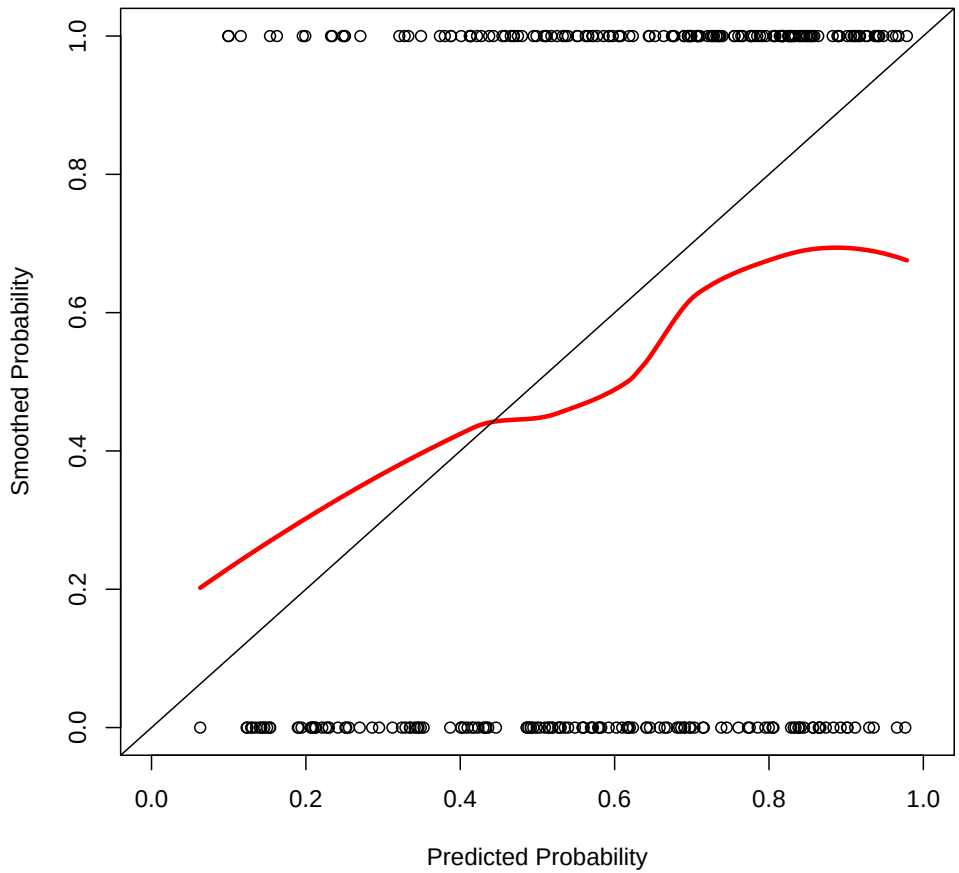
Recalibration - II

In practice this would look like

- On dataset 1, use ML method to predict y with x
- Let $S(x)$ denote the output of this model
- On dataset 2, use a univariate smoother to regress y on $z = S(x)$
- Let $T(z)$ denote the output of this model
- Our final calibrated probabilistic predictive model is $T(S(x))$

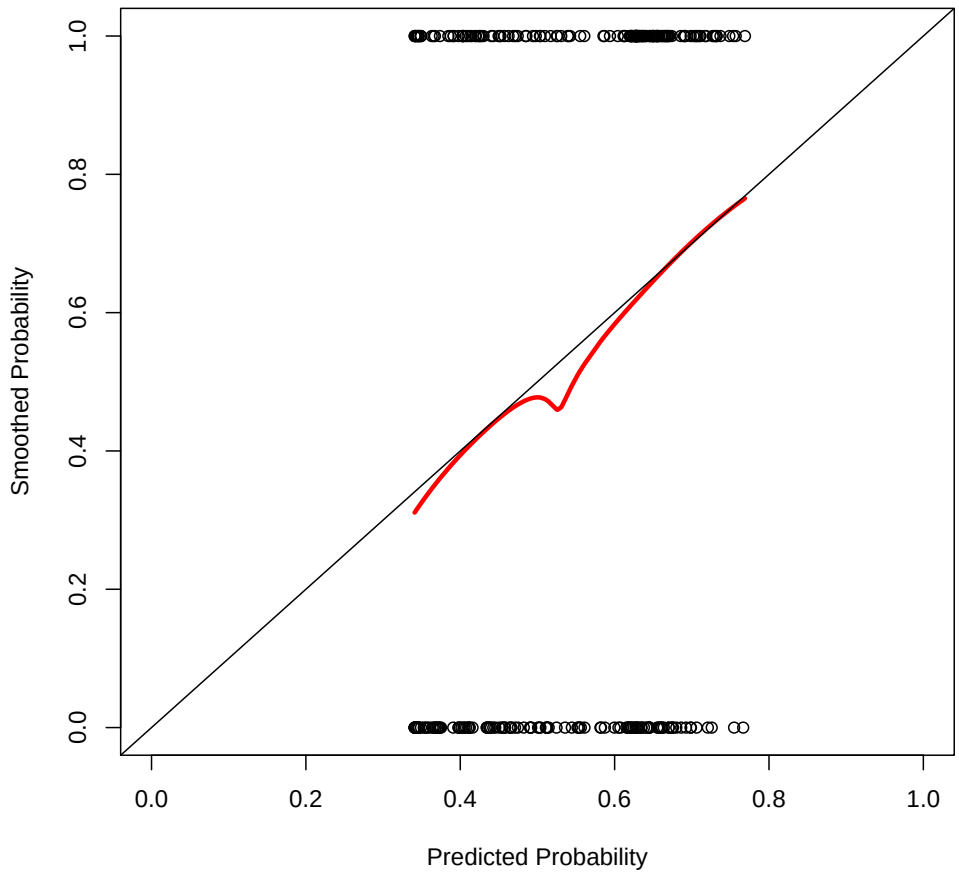
Recalibration Example

This both evaluates calibration, and gives the recalibrating transform



Recalibration Example

And this is an evaluation of the recalibrated model on a third dataset



Univariate Smoothingg for Calibration

This can be done with any univariate regression method

- Loess
- Spline regression
- Isotonic regression

Used for evaluating calibration and recalibrating...

With only 1 dataset, can build 2-stage procedure using pre-validation

Discussion Question

Suppose we want to classify patients as having cancer/not having cancer using methylation on cf-dna fragments

In particular, say we initially consider 10000 cpg sites, and try to build a classification model that uses proportion of methylated fragments at each of those sites as features.

Would it make sense to run an SVM with a non-linear kernel here?

If we used cross-validation to select between both that SVM, and a LASSO-logistic regression, which do you think would be selected?