CS6716 Pattern Recognition

Model selection

Aaron Bobick

School of Interactive Computing

Administrivia

- This lecture is really Chapter 6 of the Hastie book.
- Slides brought to you buy Bibhas Chakraborty and friends

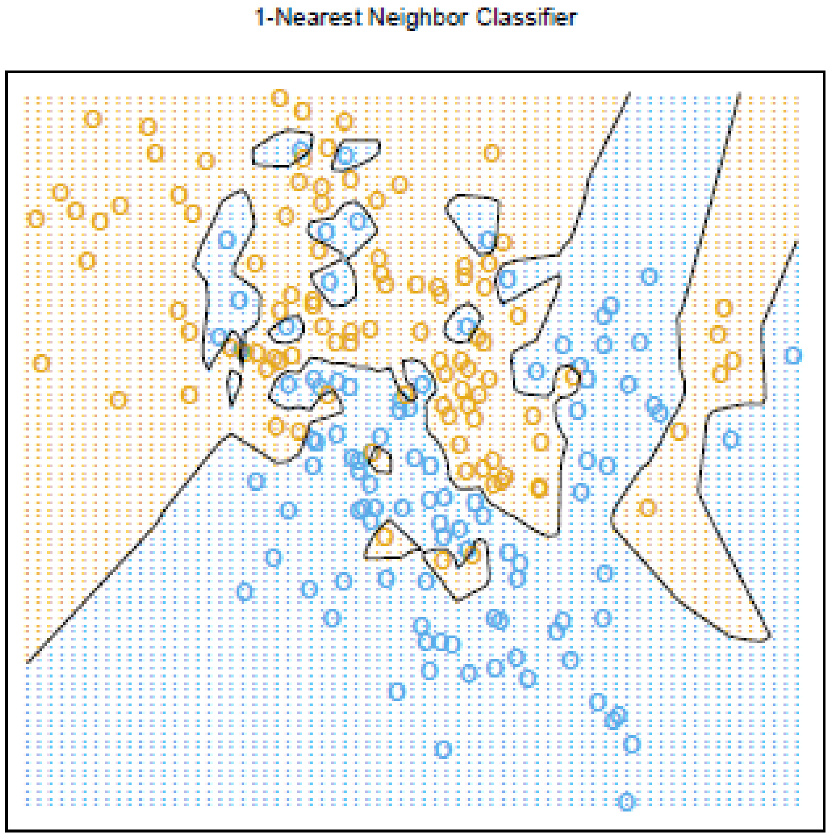
1-Nearest Neighbor

- Define a distance d(x1,x2) between any 2 examples
  - examples are feature vectors
  - so could just use Euclidean distance …

- Training: Index the training examples for fast lookup.
- Test: Given a new x, find the closest x1 from training. Classify x the same as x1 (positive or negative)

- Can learn complex decision boundaries
- As training size $\rightarrow \infty$, error rate is at most 2x the Bayes-optimal rate (i.e., the error rate you'd get from knowing the true model that generated the data – whatever it is!)

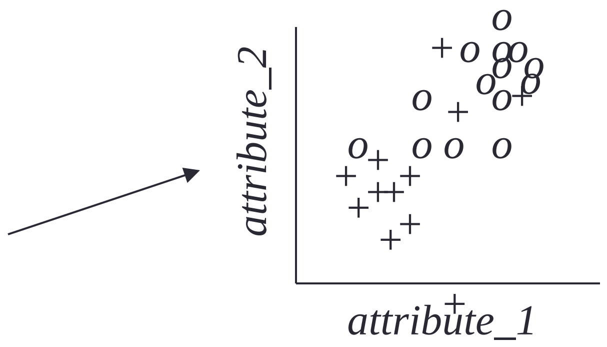
1-Nearest Neighbor – decision boundary



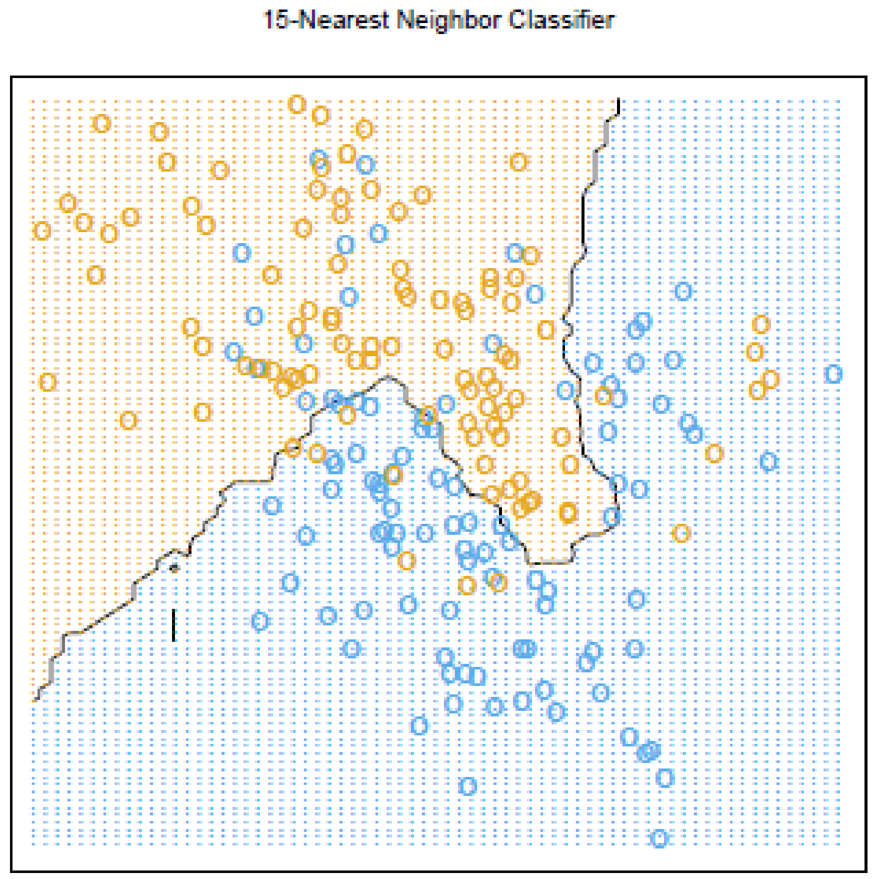
k-Nearest Neighbor

Instead of picking just the single nearest neighbor, pick the k nearest neighbors and have them vote

- Average of k points more reliable when:
  - noise in training vectors x
  - noise in training labels y
  - classes partially overlap



15 Nearest Neighbors – it's smoother…



How to choose "k"

- Odd k (often 1, 3, or 5):
  - Avoids problem of breaking ties (in a binary classifier)
- Large k:
  - less sensitive to noise (particularly class noise)
  - better probability estimates for discrete classes
  - larger training sets allow larger values of k
- Small k:
  - captures fine structure of problem space better
  - may be necessary with small training sets
- Balance between large and small k
  - What does this remind you of?
- As training set approaches infinity, and k grows large, kNN becomes Bayes optimal
- But with finite N, how to choose K?

Performance Assessment: Loss Function

- Typical choices for quantitative response Y:

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{(squared error)} \\ |Y - \hat{f}(X)| & \text{(absolute error)} \end{cases}$$

- Typical choices for categorical response G:

$$L(G, \hat{G}(X)) = I(G \neq \hat{G}(X)) \qquad \text{(0-1 loss function)}$$

$$L(G, \hat{p}(X)) = -2\sum_{k=1}^{K} I(G = k)\log \hat{p}_k$$

$$= -2\log \hat{p}_G(X) \qquad \text{(log likelihood)}$$

Training Error $\overline{err}$

- Training error is the *average* loss over the training sample.
- For the quantitative response variable Y:

$$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

- For the categorical response variable G:

$$\overline{err} = \tfrac{1}{N}\sum_{i=1}^{N} I(g_i \neq \hat{G}(x_i))$$

$$\overline{err} = -\frac{2}{N}\sum_{i=1}^{N} \log \hat{p}_{g_i}(x_i)$$

Prediction Error vs Test (Generalization) Error

- *Test* or *generalization* error an *independent* test sample is conditioned on the training set T
- *Expected prediction error* is expectation over training sets. We often ignore these differences (until CV).
- For quantitative response Y:

$$Err = E[L(Y, \hat{f}(X))]$$

- For categorical response G:

$$Err = E[L(G, \hat{G}(X))]$$

$$Err = E[L(G, \hat{p}(X))]$$

Bias, Variance and Model Complexity

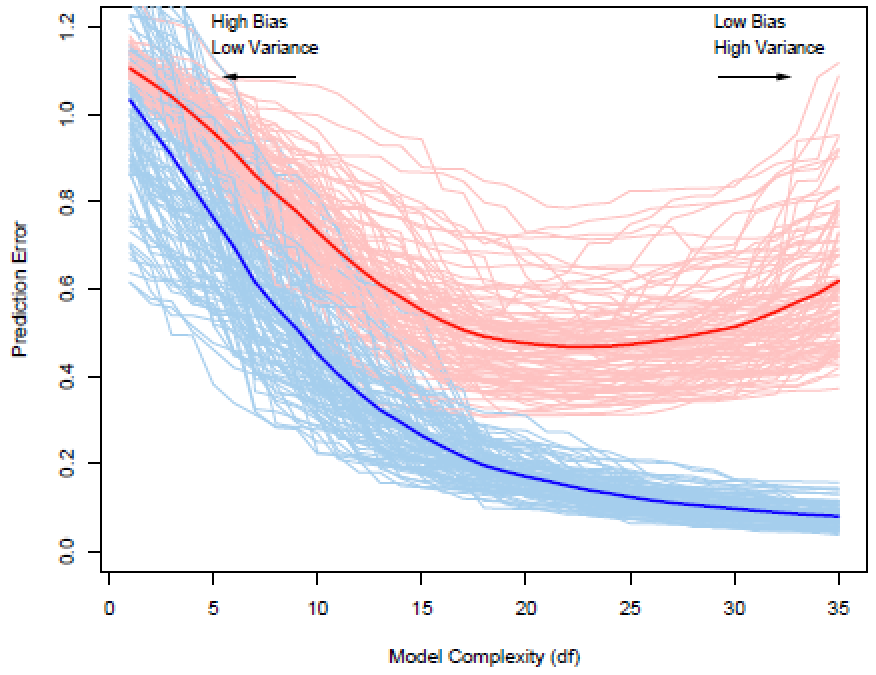


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\mathrm{err}}$*, while the light red curves show the conditional test error* $\mathrm{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\mathrm{E}[\overline{\mathrm{err}}]$.

What do we see from the preceding figure?

- There is an optimal model complexity that gives minimum test error.
- *Training error is not a good estimate of the test error.*
- There is a bias-variance tradeoff in choosing the appropriate complexity of the model.

Goals

- **Model Selection**: estimating the performance of different models in order to choose the best one.
- **Model Assessment**: having chosen a final model, estimating its generalization error on new data.
- *Model Averaging:* averaging the predictions from different models to achieve improved performance.

Splitting the data

- "In a data rich situation" split the dataset into three parts:

Training set: used to fit the models.

Validation set: used to estimate prediction error for model selection.

Test set: used to assess the generalization error for the final chosen model.

- But in reality we are not so clean.

The Bias-Variance Decomposition

- Using regression as model, assume that $Y = f(X) + \varepsilon$ where $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma_\varepsilon^2$. Then at an input point $X = x_0$:

$$
\begin{aligned}
Err(x_0) &= E[(Y - \hat{f}(x_0))^2 \mid X = x_0] \\
&= \sigma_\varepsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \sigma_\varepsilon^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0))
\end{aligned}
$$

= Irreducible Error + Bias^2 + Variance

k-NN regression example

- Assume average of *k* nearest neighbors:

$$Err(x_0) = E[(Y - \hat{f}(x_0))^2 \mid X = x_0]$$
$$= \sigma_\varepsilon^2 + [f(x_o) - \frac{1}{k}\sum_{l=1}^{k} f(x_l))]^2 + \frac{\sigma_\varepsilon^2}{k}$$

- For small k, good fit (small bias), larger variance. For big k, more bias, less variance.
- This is a model selection problem.

In-sample and Extra-sample Error

- In-sample error is the average prediction error, conditioned on the training sample ***x***'s. It is obtained when ***new responses*** are observed for the training set features.

$$Err_{in} = \frac{1}{N}\sum_{i=1}^{N} Err(x_i) = \frac{1}{N}\sum_{i=1}^{N} E_{Y^{New}} L(Y_i^{New}, \hat{f}(x_i)).$$

- Extra-sample error is the average prediction error when both features and responses are new.

Optimism of the Training Error Rate

- Typically, the training error rate will be less than the true test error. Why?
- Define the **optimism** as the expected difference between *Err*$_{in}$ and the training error:

$$op \equiv Err_{in} - E_y(\overline{err})$$

- Can define an expected optimism over training sets but we won't here.

Optimism (cont'd)

- For squared error, 0-1, and other loss function, "it can be shown" generally that

$$op = \frac{2}{N}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

The more influence y_i has on its own prediction the more optimistic you are.

- Therefore

$$Err_{in} = E_y(\overline{err}) + \frac{2}{N}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

- Can be simplified as $Err_{in} = E_y(\overline{err}) + 2\cdot\frac{d}{N}\sigma_\varepsilon^{\,2}$ for the model $Y = f(X) + \varepsilon$ by a linear fit with d inputs.

How to estimate prediction error?

- Estimate the optimism and then add it to the training error rate.
  - Methods such as AIC, BIC work in this way for a special class of estimates that are linear in their parameters.
- Estimating in-sample error is used for model selection.

- Methods like cross-validation and bootstrap:

  - direct estimates of the extra-sample error.

  - can be used with any loss function.

  - used for model assessment.

Estimates of In-Sample Prediction Error

- General form $Est(Err_{in}) = \overline{err} + Est(op)$
- C_p statistic (when *d* parameters are fitted under squared error loss):

$$C_p = \overline{err} + 2 \cdot \frac{d}{N} \hat{\sigma}_{\varepsilon}{}^2$$

- AIC (Akaike information criterion), a more generally applicable estimate of $\mathrm{Err}_{\mathrm{in}}$ when a log-likelihood loss function is used:

$$-2E[\log \Pr_{\hat{\theta}}(Y)] \approx -\frac{2}{N} E[\mathrm{loglik}] + 2\frac{d}{N}$$

$$\mathrm{loglik} = \sum_{i-1}^{N} \log \Pr_{\theta}(y_i)$$

More on AIC

- Four Gaussian, AIC is identical to C_p

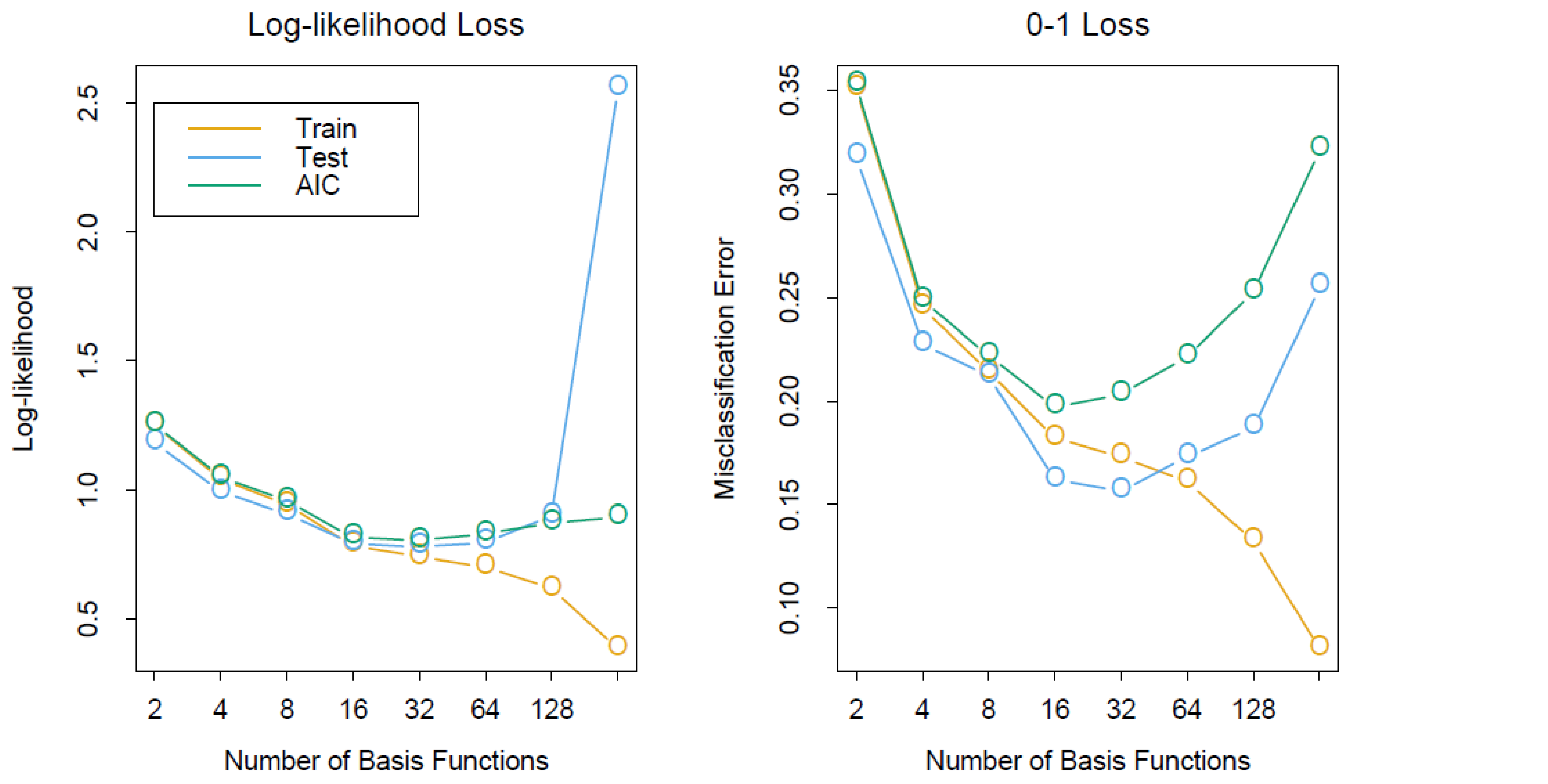
- Given a set of models $f_\alpha(x)$ indexed by a tuning parameter α, define

$$AIC(\alpha) = \overline{err}(\alpha) + 2\frac{d(\alpha)}{N}\hat{\sigma}_\varepsilon{}^2$$

- Find the tuning parameter $\hat{\alpha}$ that minimizes the function, and the final chosen model is $f_{\hat{\alpha}}(x)$

Bias, Variance and prediction error

- Phoneme example of Hastie using logistic regression



Bayesian Information Criterion (BIC)

- Model selection tool applicable in settings where the fitting is carried out by maximization of a log-likelihood.
- Motivation from Bayesian point of view.
- BIC tends to penalize complex models more heavily, giving preference to simpler models in selection.
- Its generic form is: $BIC = -2 \cdot (\log lik) + (\log N) \cdot d.$
- If Gaussian (Hastie 233):

$$\mathrm{BIC} = \frac{N}{\sigma_\varepsilon^2}\left[\overline{\mathrm{err}} + (\log N) \cdot \frac{d}{N}\sigma_\varepsilon^2\right]$$

Bayesian Model Selection

- Suppose we have candidate models $M_m, m = 1,\ldots,M$
  with corresponding model parameters θ_m.
- Prior distribution: $\Pr(\theta_m \mid M_m), m = 1,\ldots,M$.
- Posterior probability:

$$\Pr(M_m \mid Z) \propto \Pr(M_m) \cdot \Pr(Z \mid M_m).$$

- Compare two models via posterior odds:

$$\frac{\Pr(M_m \mid Z)}{\Pr(M_l \mid Z)} = \frac{\Pr(M_m)}{\Pr(M_l)} \cdot \frac{\Pr(Z \mid M_m)}{\Pr(Z \mid M_l)}$$

- The second factor on the RHS is called the Bayes factor and describes the contribution of the data towards posterior odds.

Bayes and BIC (cont)

- Using Laplace approximation (see Murphy), one can establish a simple (but approximate) relationship between posterior model probability and the BIC.

$$\log \Pr(Z \mid M_m) = \log Pr(Z \mid \hat{\theta}_m, M_m) - \frac{d_m}{2} \cdot \log N + K$$

- If we define the loss function $-2 \log Pr(Z \mid \hat{\theta}_m, M_m)$ then for Gaussian:

$$\mathrm{BIC} = \frac{N}{\sigma_\varepsilon^2}\left[\overline{\mathrm{err}} + (\log N) \cdot \frac{d}{N}\sigma_\varepsilon^2\right]$$

- So BIC is Bayes!

Digression…. Josh Tennenbaum

- I have a model to produce numbers between 0 and 100.
- I tell you four of my numbers are 8, 32, 2, 64

- Do you guess the evens? The numbers between 2 and 64? Other guesses? Which seems best?

Bayesian Approach Continued

- Unless strong evidence to the contrary, we typically assume that prior over models is uniform (non-informative prior).

- Lower BIC implies higher posterior probability of the model. Use of BIC as model selection criterion is thus justified.

AIC or BIC?

- BIC is asymptotically consistent as a selection criterion. That means, given a family of models including the true model, the probability that BIC will select the correct one approaches one as the sample size becomes large.
- AIC does not have the above property. Instead, it tends to choose more complex models as $N \to \infty.$
- For small or moderate samples, BIC often chooses models that are too simple, because of its heavy penalty on complexity.

Cross-Validation

- The simplest and most widely used method for estimating prediction error.

- The idea is to directly estimate the extra sample error $Err = E\left[L\left(Y, \hat{f}(X)\right)\right]$, when the method $\hat{f}(x)$ is applied to an independent test sample.

- In K-fold cross-validation, we split the data into roughly equal-size parts. For the *k*-th part, fit the model to the other K-1 parts and calculate the prediction error of the fitted model when predicting the *k*-th part of the data.

Cross-Validation (Cont'd)

- The cross-validation estimate of prediction error is

$$CV(\alpha) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-k(i)}(x_i, \alpha)).$$

- This $CV(\alpha)$ provides an estimate of the test error, and we find the tuning parameter $\hat{\alpha}$ minimizes it.
- Our final chosen model will be $f(x, \hat{\alpha})$ which we fit to all the data.

Value of K?

- If $K = N$ CV is approximately unbiased, but has high variance. The computational burden is also high.
- On the other hand, with, say, $K = 5$ CV has low variance but more bias.
- If the learning curve has a considerable slope at the given training set size, 5-fold, 10-fold CV will overestimate the true prediction error…

The Learning Curve

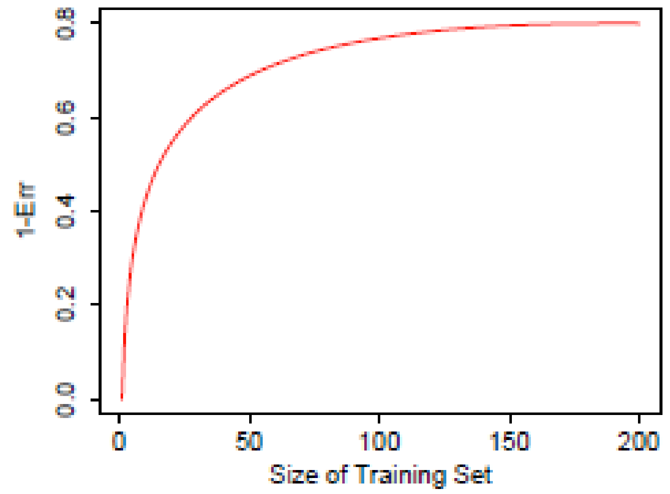


FIGURE 7.8. *Hypothetical learning curve for a classifier on a given task: a plot of* $1 - \text{Err}$ *versus the size of the training set* N*. With a dataset of* 200 *observations,* 5*-fold cross-validation would use training sets of size* 160*, which would behave much like the full set. However, with a dataset of* 50 *observations fivefold cross–validation would use training sets of size* 40*, and this would result in a considerable overestimate of prediction error.*

Some funny things…

- Simulation: N=50 samples of two classes, 2000 predictor variables
- Screen the predictors: find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels
- 2. Using just this subset of predictors, build a multivariate classifier (say 1-NN).
- 3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

What happened?

- Actually the predictors were uncorrelated to label (error should be 50%)
- Using 1-NN showed a CV error of 3%!!!
- How did that happen?
- Step 1 – already saw the labels!!! Not a real CV. You must remove the k-th part completely.

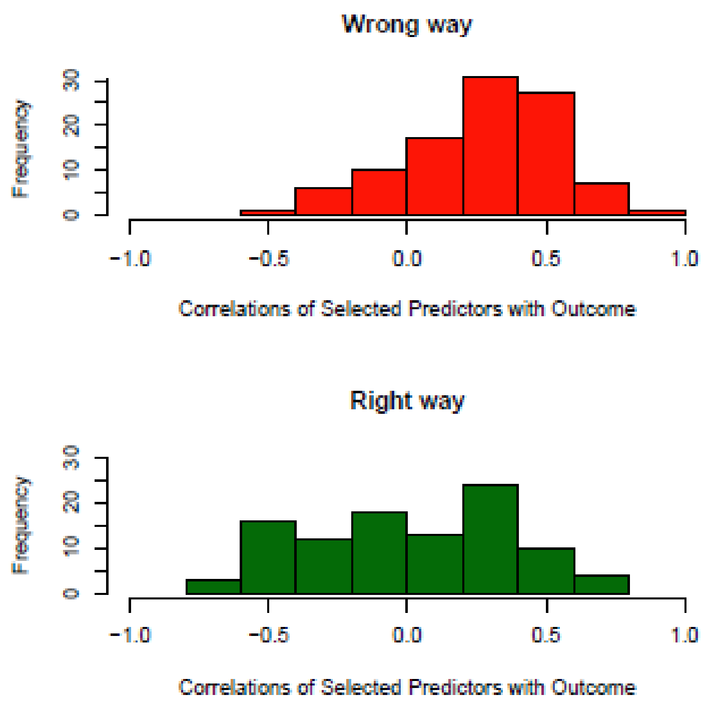
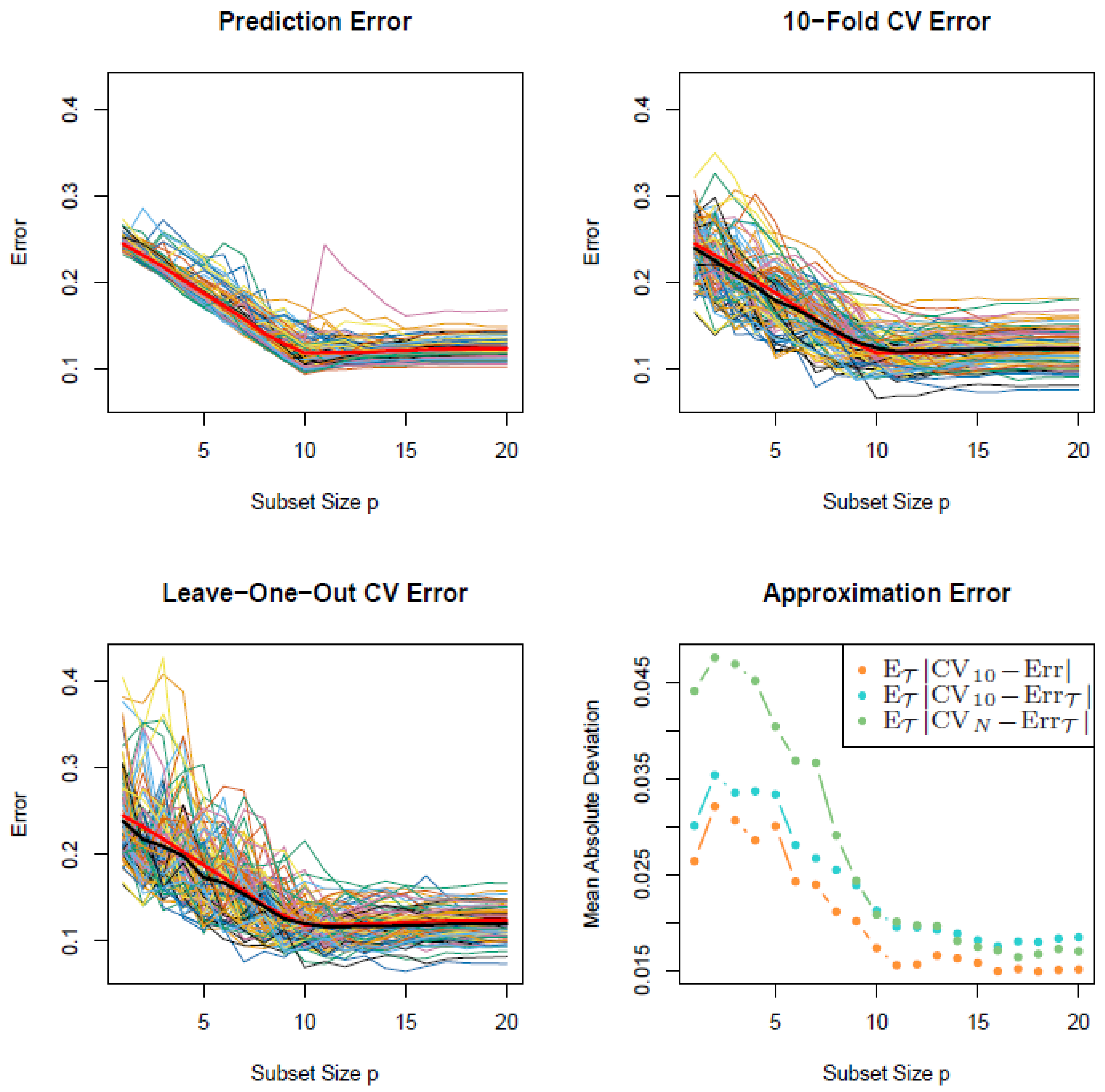


FIGURE 7.10. *Cross-validation the wrong and right way: histograms shows the correlation of class labels, in* 10 *randomly chosen samples, with the* 100 *predictors chosen using the incorrect (upper red) and correct (lower green) versions of cross-validation.*

CV behavior as function of K

X is 20 dim vector on $[0\ 1]^{20}$.Y is 1 if sum of the first 10 elements is great than 5, otherwise 0. Use best subset linear regression of size p.



End

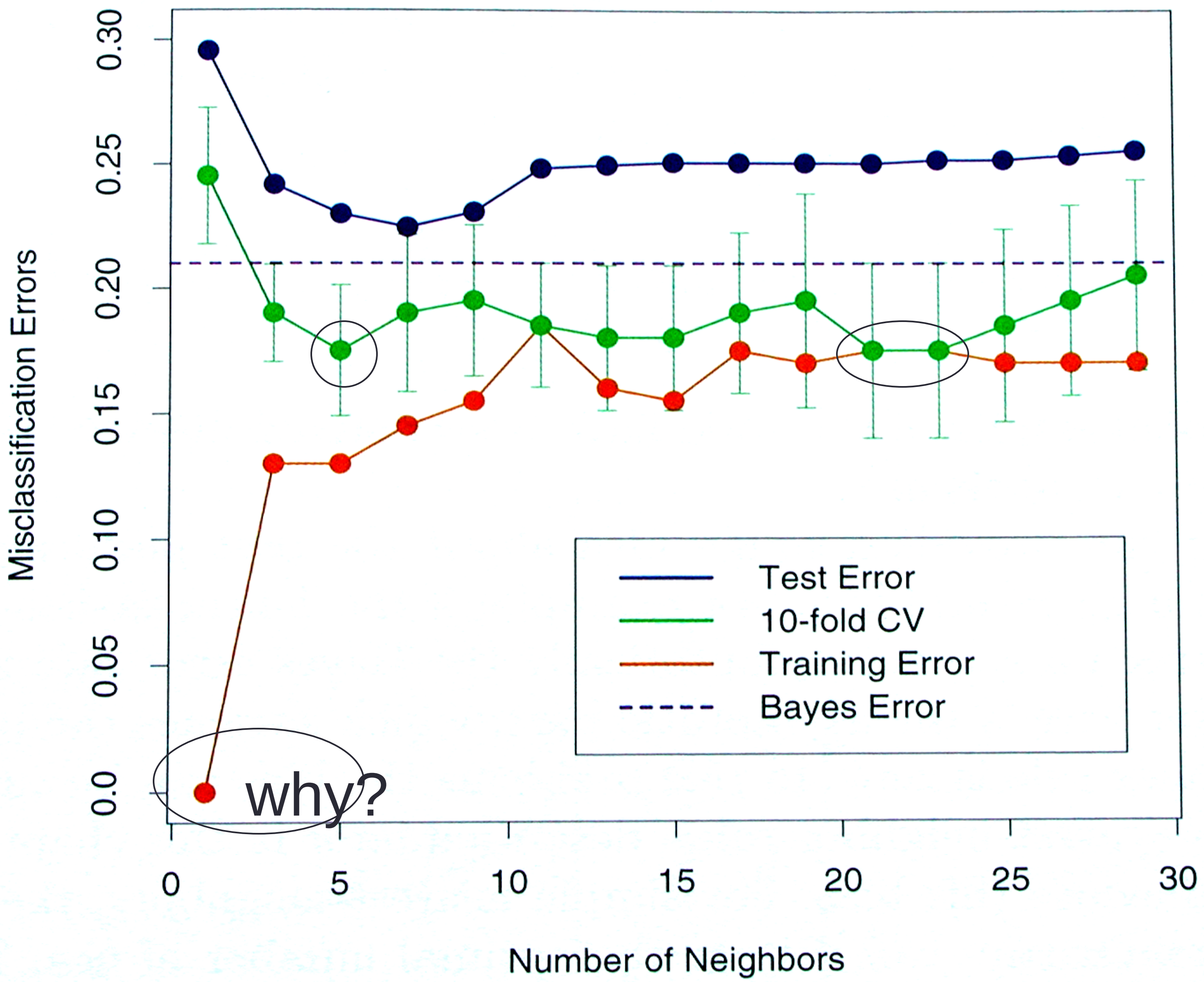
How to estimate prediction error?

- Estimate the **optimism** and then add it to the **training error rate**.

  -- Methods such as AIC, BIC work in this way for a special class of estimates that are **linear** in their **parameters**.
- Estimating **in-sample error** is used for **model selection**.
- Methods like **cross-validation** and **bootstrap:**

  - direct estimates of the **extra-sample error.**
  - can be used with any loss function.
  - used for **model assessment**.

From Hastie, Tibshirani, Friedman 2001 p419



Cross-Validation

- Models usually perform better on training data than on future test cases
- 1-NN is 100% accurate on training data!
- “Leave-one-out” cross validation:
  - “remove” each case one-at-a-time
  - use as test case with remaining cases as train set
  - average performance over all test cases
- LOOCV is impractical with most learning methods, but extremely efficient with MBL!

Advantages of Memory-Based Methods

- Lazy learning: don’t do any work until you know what you want to predict (and from what variables!)
  - never need to learn a global model
  - many simple local models taken together can represent a more complex global model
- Learns arbitrarily complicated decision boundaries
- Very efficient cross-validation
- Easy to explain to users how it works
  - … and why it made a particular decision!
- Can use ***any*** distance metric: string-edit distance, …
  - handles missing values, time-varying distributions, ...

Bootstrap Method

- General tool for assessing statistical accuracy.
- Suppose we have a model to fit the training data

$$Z = \{(x_i, y_i), i = 1, \dots, N\}.$$

- The idea is to draw random samples with replacement of size N from the training data. This process is repeated B times to get B bootstrap datasets.
- Refit the model to each of the bootstrap datasets and examine the behavior of the fits over B replications.

Bootstrap (Cont'd)

- Here $S(Z)$ is any quantity computed from the data Z. From the bootstrap sampling, we can estimate any aspect of the distribution of $S(Z)$.

  For example, its variance is estimated by

$$\hat{Var}(S(Z)) = \frac{1}{B-1}\sum_{b=1}^{B}(S(Z^{*b}) - \bar{S}^{*})^2,$$

  where $\bar{S}^{*} = \sum_b S(Z^{*b}) / B.$

Bootstrap used to estimate prediction error: Mimic CV

- Fit the model on a set of bootstrap samples keeping track of predictions from bootstrap samples not containing that observation.

- The leave-one-out bootstrap estimate of prediction error is

$$\hat{Err}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)).$$

- $\hat{Err}^{(1)}$ solves the over-fitting problem suffered by $\hat{Err}_{boot}$, but has training-set-size bias, mentioned in the discussion of CV.

The "0.632 Estimator"

- Average number of distinct observations in each bootstrap sample is approximately $0.632 \cdot N$.
- Bias will roughly behave like that of two-fold cross-validation (biased upwards).
- The "0.632 estimator" is designed to get rid of this bias.

$$E\hat{r}r^{(0.632)} = 0.368 \cdot \overline{err} + 0.632 \cdot E\hat{r}r^{(1)}.$$

Bagging

- Introduced by Breiman (Machine Learning, 1996).
- Acronym for ‘Bootstrap aggregation’ .
- It averages the prediction over a collection of bootstrap samples, thus reducing the variance in prediction.

Bagging (Cont'd)

- Consider the regression problem with training data $Z=\{(x_1,y_1),\ldots,(x_N,y_N)\}.$
- Fit a model and get a prediction $\hat{f}(x)$ at the input x.
- For each bootstrap sample $Z^{*b}, b=1,\ldots,B,$

  fit the model, get the prediction $\hat{f}^{*b}(x)$. Then the bagging (or, bagged) estimate is:

$$\hat{f}_{bag}(x)=\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

Bagging (extended to classification)

- Let $\hat{G}$ be a classifier for a K-class response. Consider an underlying indicator vector function

$$\hat{f}(x) = (0,\ldots,0,1,0,\ldots,0),$$

the entry in the i-th place is 1 if the prediction for x is the i-th class, such that

$$\hat{G}(x) = \arg\max_k \hat{f}(x).$$

- Then the bagged estimate $\hat{f}_{bag}(x) = (p_1,\ldots,p_K),$ where p_k is the proportion of base classifiers predicting class k at x where $k = 1,\ldots,K.$
- Finally, $\hat{G}_{bag}(x) = \arg\max_k \hat{f}_{bag}(x).$

Bagging Example

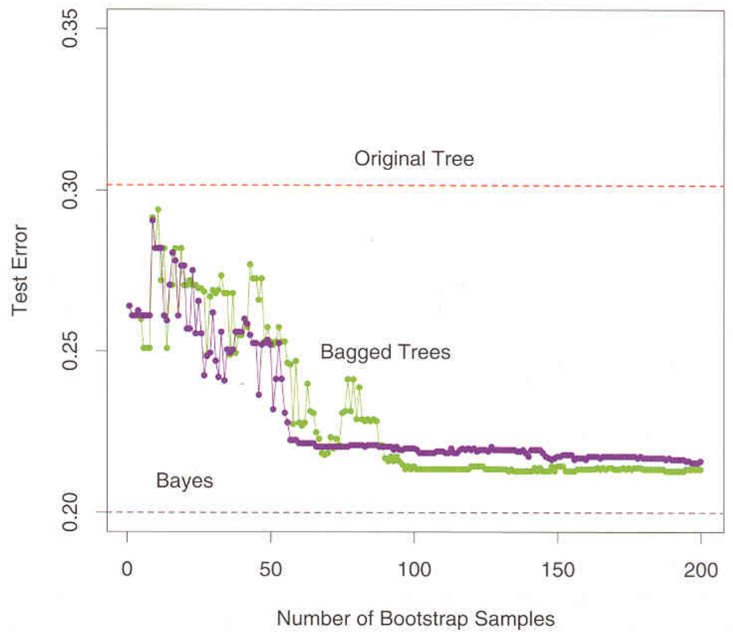


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The green points correspond to majority vote, while the purple points average the probabilities.*

- The figure is taken from Pg 249 of the book *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman.

Bayesian Model Averaging

- Candidate models: $M_m, m = 1, \dots, M.$
- Posterior distribution and mean:

$$\Pr(\zeta \mid Z) = \sum_{m=1}^{M} \Pr(\zeta \mid M_m, Z) \Pr(M_m \mid Z),$$

$$E(\zeta \mid Z) = \sum_{m=1}^{M} E(\zeta \mid M_m, Z) \Pr(M_m \mid Z).$$

- Bayesian prediction (posterior mean) is a weighted average of individual predictions, with weights proportional to posterior probability of each model.
- Posterior model probabilities can be estimated by BIC.

Frequentist Model Averaging

- Given predictions $\hat{f}_1(x), \ldots, \hat{f}_M(x),$ under squared error loss, we can seek the weights such that

$$\hat{w} = \arg\min_{w} E_P[Y - \sum_{m=1}^{M} w_m \hat{f}_m(x)]^2 .$$

- The solution is the population linear regression of on Y

$$\hat{F}(x)^T \equiv [\hat{f}_1(x), \ldots, \hat{f}_M(x)]:$$

$$\hat{w} = E_P[\hat{F}(x)\hat{F}(x)^T]^{-1} E_P[\hat{F}(x)Y].$$

- Combining models never makes things worse, at the population level. As population regression is not available, it is replaced by regression over the training set, which sometimes doesn't work well.

Stacking

- *Stacked generalization* , or *stacking* is a way to get around the problem.
- The stacking weights are given by

$$\hat{w}^{st} = \arg\min_{w} \sum_{i=1}^{N} [y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x_i)]^2.$$

- The final stacking prediction is: $\sum_{m=1}^{M} w_m^{st} \hat{f}_m(x).$
- Close connection with leave-out-one-cross-validation.
- Better prediction, less interpretability.

References

- Hastie,T., Tibshirani, R. and Friedman, J.-*The Elements of Statistical Learning* (ch. 7 and 8)

Memory-Based Learning

E.g., k-Nearest Neighbor

Also known as “case-based” or “example-based” learning

Intuition behind memory-based learning

- Similar inputs map to similar outputs
  - If not true ➔ learning is impossible
  - If true ➔ learning reduces to defining "*similar*"
- Not all similarities created equal
  - guess J. D. Salinger's weight
    - who are the similar people?
    - similar occupation, age, diet, genes, climate, …
  - guess J. D. Salinger's IQ
    - similar occupation, writing style, fame, SAT score, …
- Superficial vs. deep similarities?
  - B. F. Skinner and the behaviorism movement

what do brains actually do?

parts of slide thanks to Rich Caruana

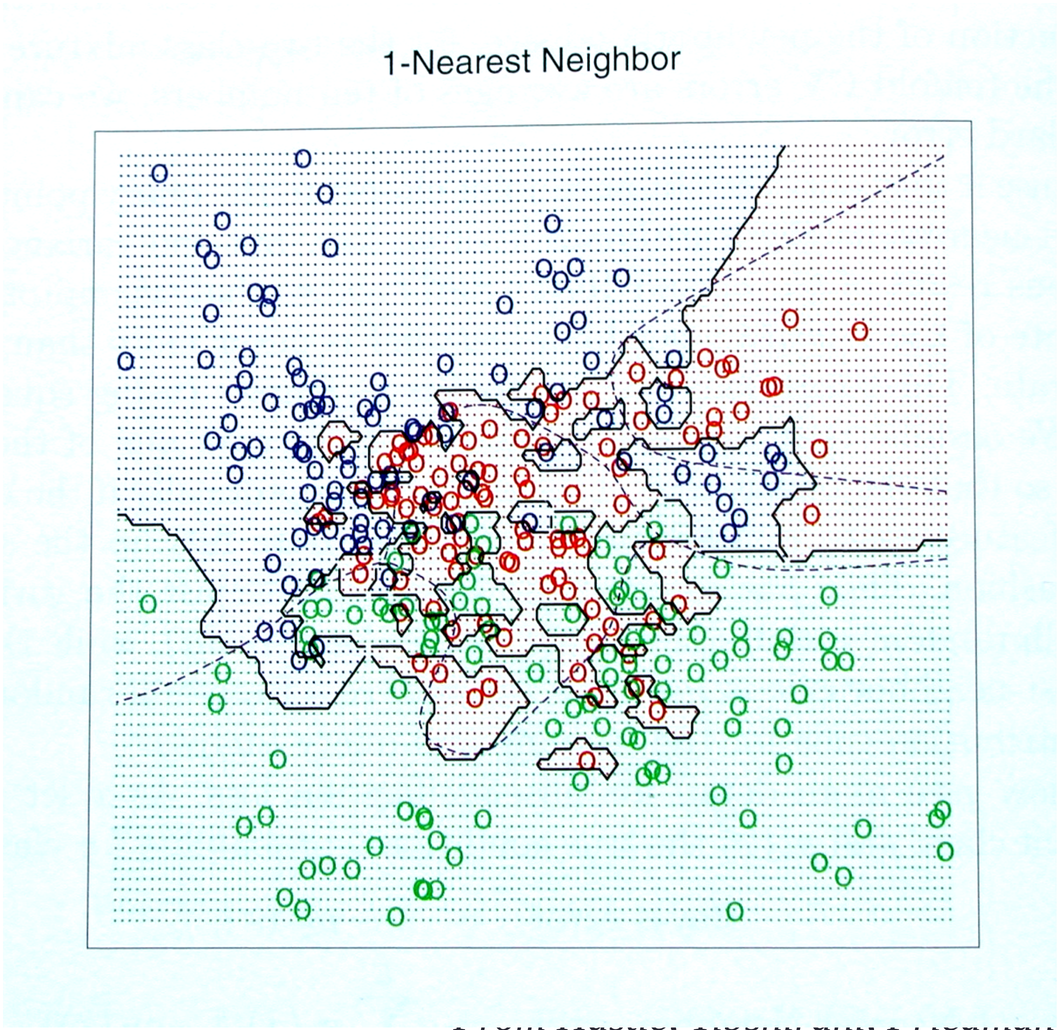
1-Nearest Neighbor

- Define a distance d(x1,x2) between any 2 examples
  - examples are feature vectors
  - so could just use Euclidean distance …

- Training: Index the training examples for fast lookup.
- Test: Given a new x, find the closest x1 from training. Classify x the same as x1 (positive or negative)

- Can learn complex decision boundaries
- As training size → ∞, error rate is at most 2x the Bayes-optimal rate (i.e., the error rate you'd get from knowing the true model that generated the data – whatever it is!)

1-Nearest Neighbor – decision boundary

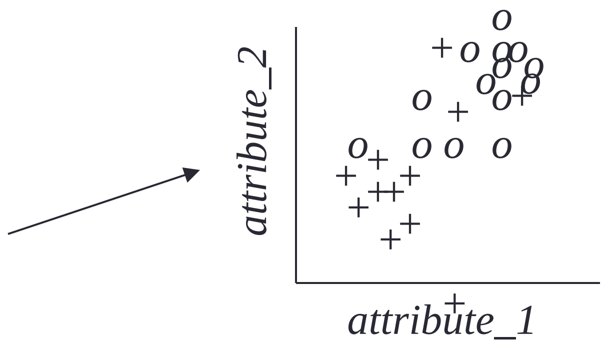


From Hastie, Tibshirani, Friedman 2001 p418

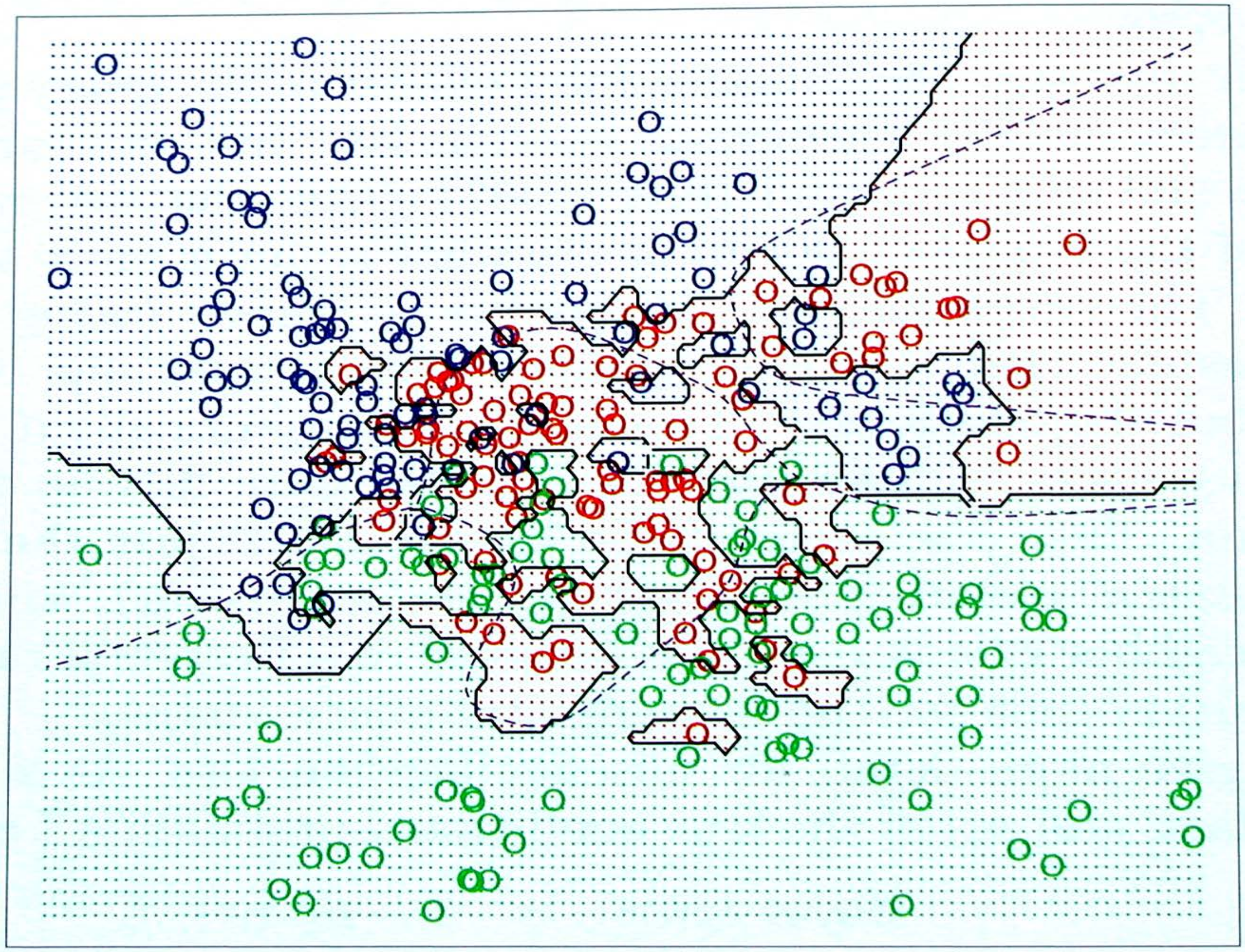
k-Nearest Neighbor

- Instead of picking just the single nearest neighbor, pick the k nearest neighbors and have them vote

- Average of k points more reliable when:
  - noise in training vectors x
  - noise in training labels y
  - classes partially overlap

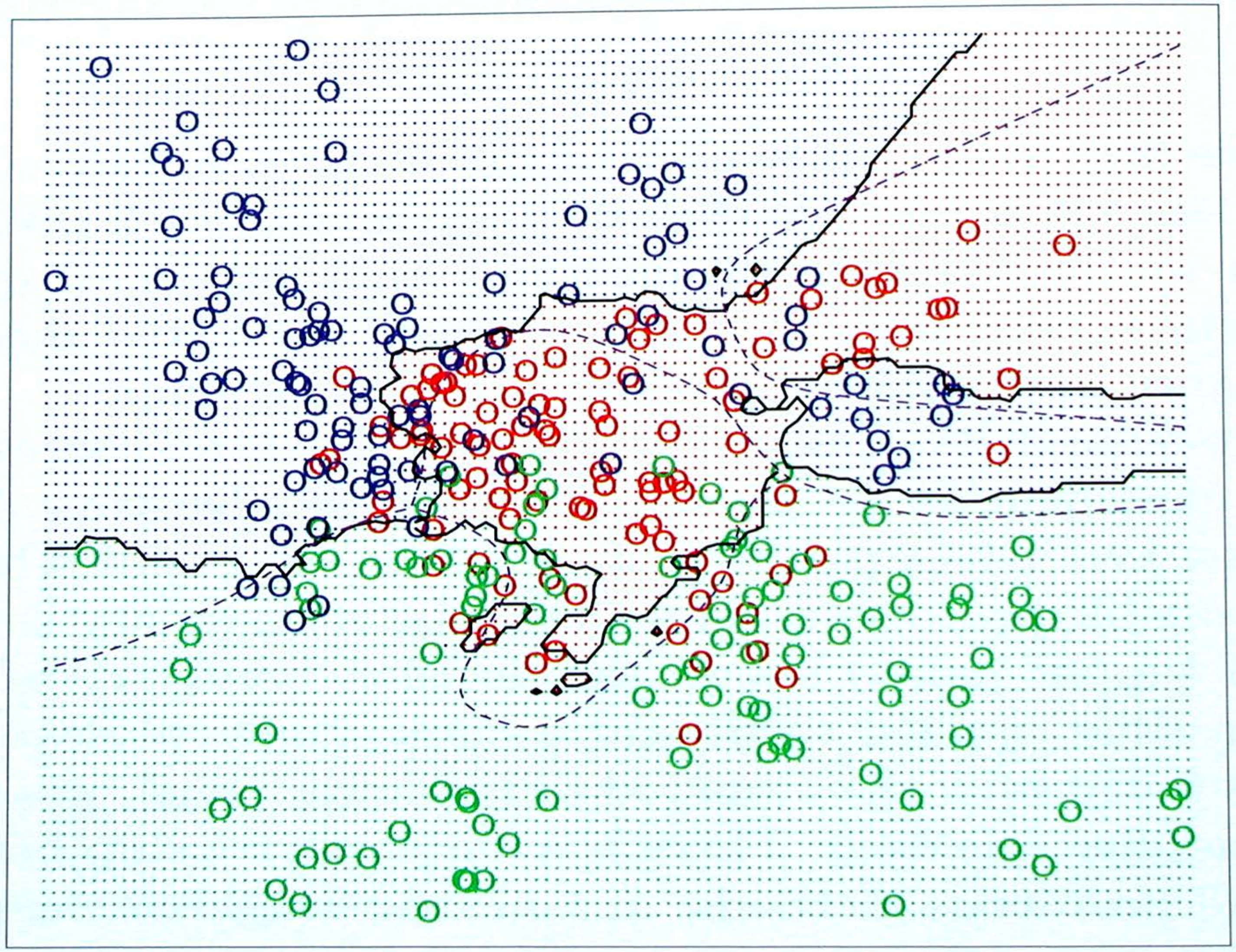


1 Nearest Neighbor – decision boundary



From Hastie, Tibshirani, Friedman 2001 p418

15 Nearest Neighbors – it's smoother!



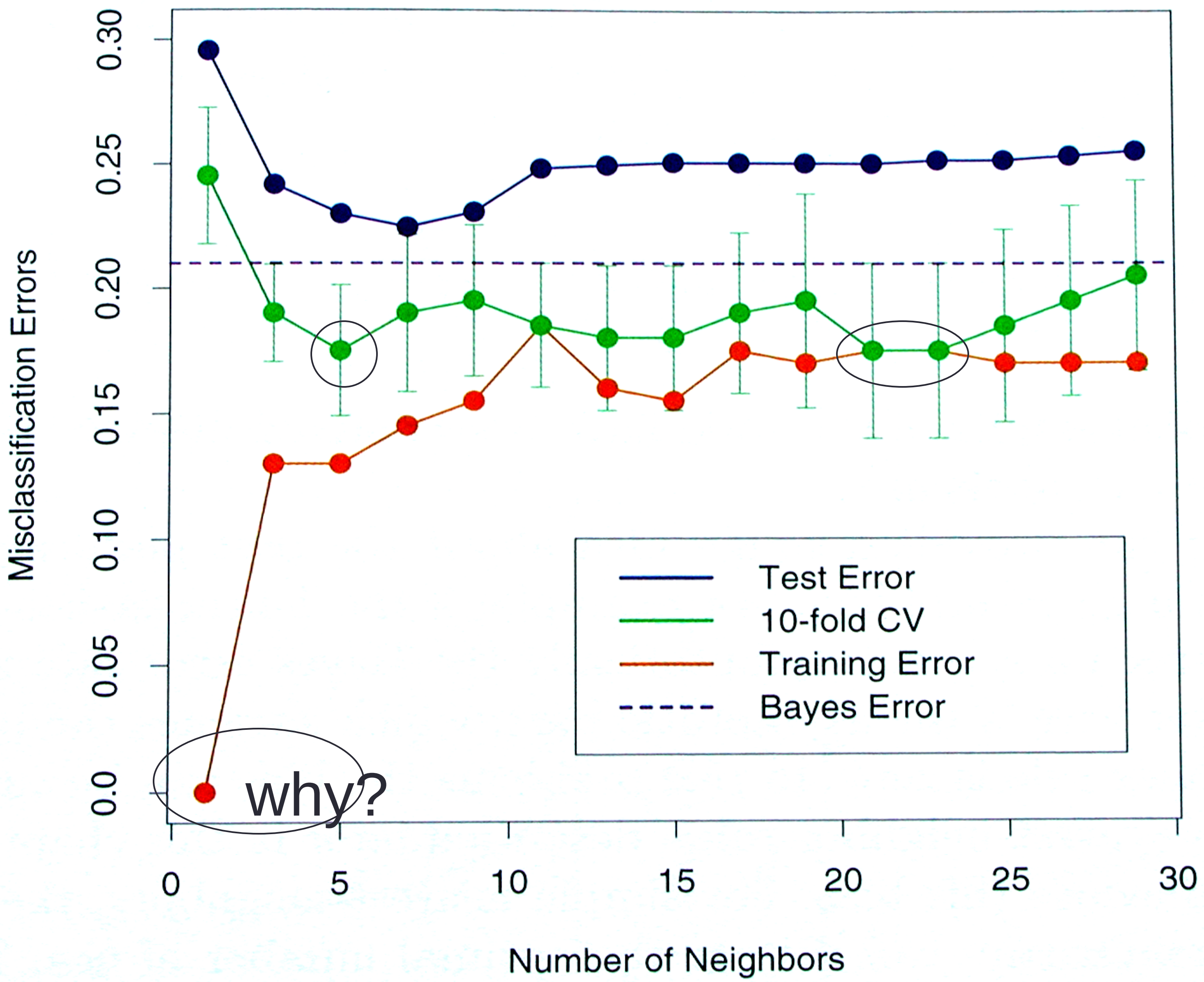
From Hastie, Tibshirani, Friedman 2001 p418

How to choose "k"

- Odd k (often 1, 3, or 5):
  - Avoids problem of breaking ties (in a binary classifier)
- Large k:
  - less sensitive to noise (particularly class noise)
  - better probability estimates for discrete classes
  - larger training sets allow larger values of k
- Small k:
  - captures fine structure of problem space better
  - may be necessary with small training sets
- Balance between large and small k
  - What does this remind you of?
- As training set approaches infinity, and k grows large, kNN becomes Bayes optimal

slide thanks to Rich Caruana (modified)

From Hastie, Tibshirani, Friedman 2001 p419



Cross-Validation

- Models usually perform better on training data than on future test cases
- 1-NN is 100% accurate on training data!
- "Leave-one-out" cross validation:
  - "remove" each case one-at-a-time
  - use as test case with remaining cases as train set
  - average performance over all test cases
- LOOCV is impractical with most learning methods, but extremely efficient with MBL!

slide thanks to Rich Caruana

Distance-Weighted kNN

- hard to pick large vs. small k
  - may not even want k to be constant
- use large k, but more emphasis on <u>nearer</u> neighbors?

$$prediction(x) = \frac{\sum_{i=1}^{k} w_i \cdot y_i}{\sum_{i=1}^{k} w_i}$$

where $x_1, \dots x_k$ are the k-NN and $y_1, \dots y_k$ their labels

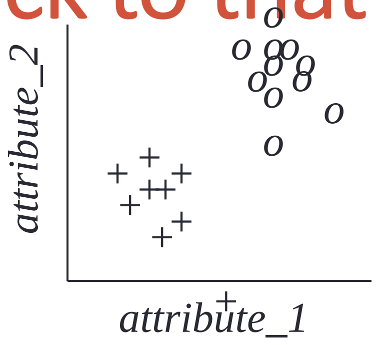
We define relative weights for the k-NN:

$$w_i = \frac{1}{Dist(x_i, x)} \text{ or maybe } \frac{1}{Dist(x_i, x)^{\beta}} \text{ or often } \frac{1}{\exp \beta \cdot Dist(x_i, x)}$$

Combining k-NN with other methods, #1

- Instead of having the k-NN simply vote, put them into a little machine learner!
- To classify x, train a "local" classifier on its k nearest neighbors (maybe weighted).
  - polynomial, neural network, …

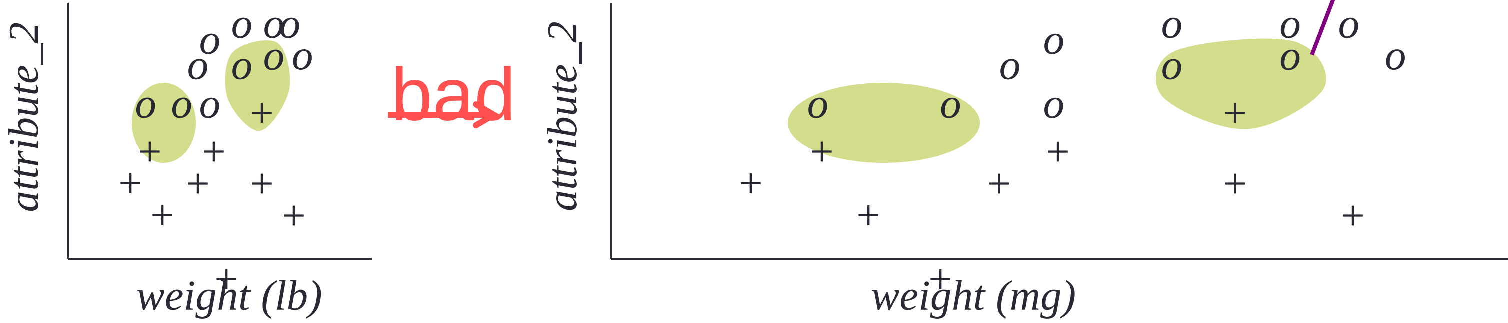
Now back to that distance function



- Euclidean distance treats all of the input dimensions as equally important

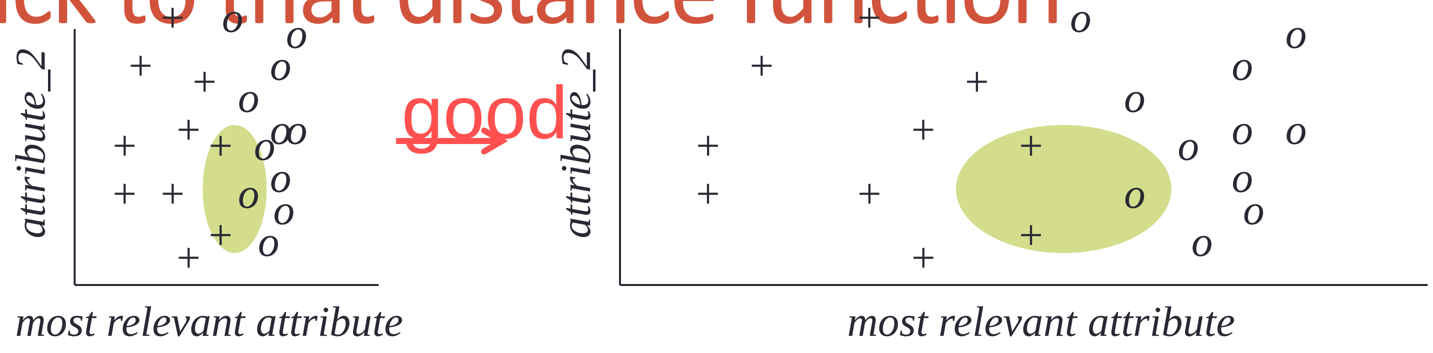
These o's are now "closer" to + than to each other ☹

Now back to that distance function



- Euclidean distance treats all of the input dimensions as equally important
- Problem #1:
  - What if the input represents physical weight not in pounds but in milligrams?
    - Then small differences in physical weight dimension have a huge effect on distances, overwhelming other features.
  - Should really correct for these arbitrary "scaling" issues.
    - One simple idea: rescale weights so that standard deviation = 1.

Now back to that distance function



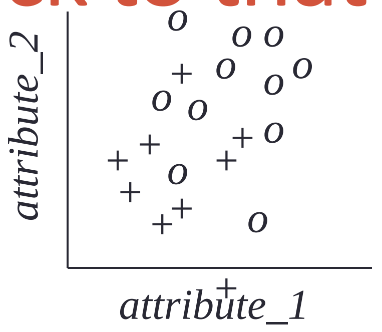
- Euclidean distance treats all of the input dimensions as equally important
- Problem #2:
  - What if some dimensions more correlated with true label?
    - (more relevant, or less noisy)
  - Stretch those dimensions out so that they are more important in determining distance.
    - One common technique is called "information gain."

Weighted Euclidean Distance

$$d(x, x') = \sqrt{\sum_{i=1}^{N} s_i \cdot \left(x_i - x'_i\right)^2}$$

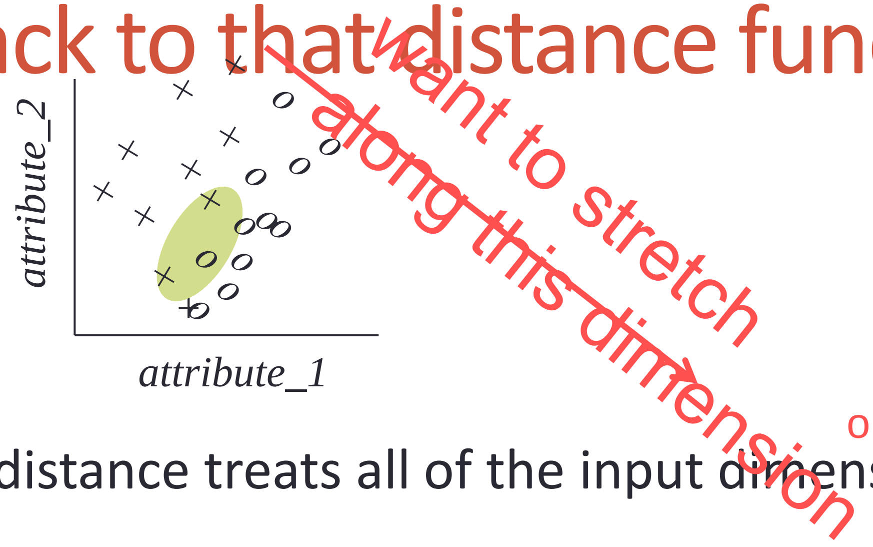
- large weight s_i $\rightarrow$ attribute i is more important
- small weight s_i $\rightarrow$ attribute i is less important
- zero weight s_i $\rightarrow$ attribute i doesn't matter

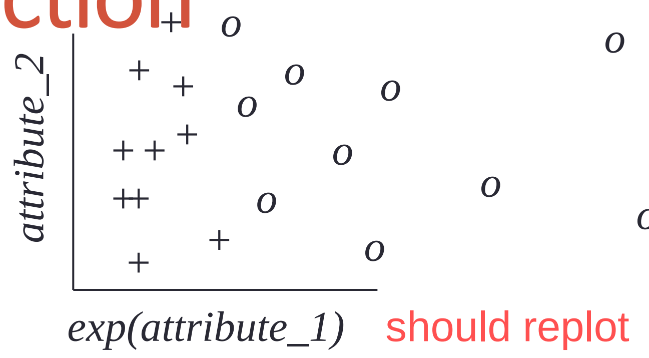
Now back to that distance function



- Euclidean distance treats all of the input dimensions as equally important
- Problem #3:
  - Do we really want to decide <u>separately</u> and <u>theoretically</u> how to scale each dimension?
  - Could simply pick dimension scaling factors to maximize performance on development data. (maybe do leave-one-out)
    - Similarly, pick number of neighbors k and how to weight them.
  - Especially useful if performance measurement is complicated (e.g., 3 classes and differing misclassification costs).

Now back to that distance function



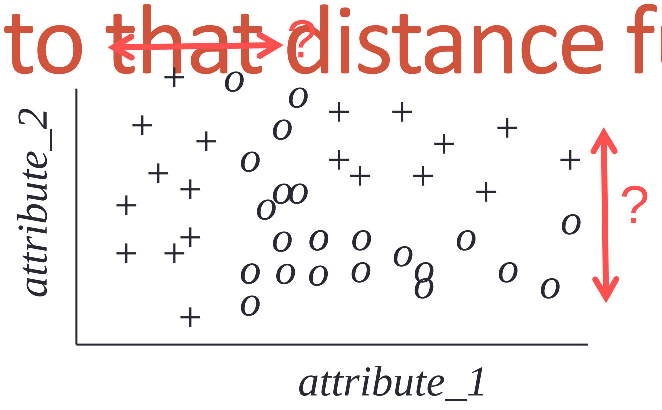


should replot on log scale before measuring dist

- Euclidean distance treats all of the input dimensions as equally important
- Problem #4:
  - Is it the original input dimensions that we want to scale?
  - What if the true clusters run diagonally? Or curve?
  - We can transform the data first by extracting a different, useful set of features from it:
    - Linear discriminant analysis
    - Hidden layer of a neural network

} i.e., redescribe the data by how a **different** type of learned classifier

Now back to that distance function



- Euclidean distance treats all of the input dimensions as equally important
- Problem #5:
  - Do we really want to transform the data globally?
  - What if different regions of the data space behave differently?
  - Could find 300 “nearest” neighbors (using *global* transform), then *locally* transform that subset of the data to redefine “near”
  - Maybe could use decision trees to split up the data space first

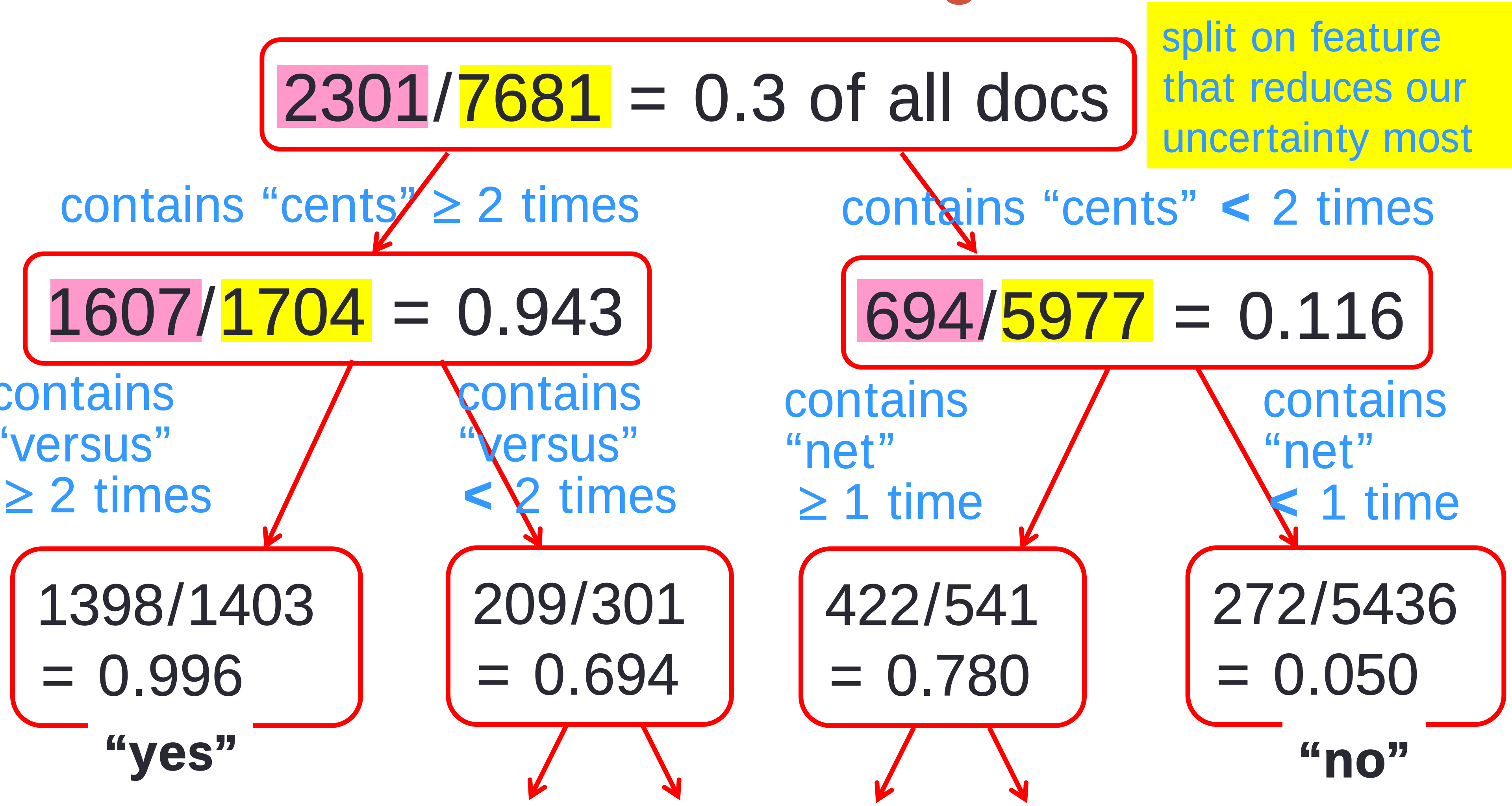
Why are we doing all this preprocessing?

- Shouldn't the user figure out a smart way to transform the data <u>before</u> giving it to k-NN?
- Sure, that's always good, but what will the user try?
  - Probably a lot of the same things we're discussing.
  - She'll stare at the training data and try to figure out how to transform it so that close neighbors tend to have the same label.
  - To be nice to her, we're trying to <u>automate</u> the most common parts of that process – like scaling the dimensions appropriately.
  - We may still miss patterns that her visual system or expertise can find. So she may still want to transform the data.
  - On the other hand, we may find patterns that would be hard for her to see.

Tangent: Decision Trees

(a different simple method)

Is this Reuters article an Earnings Announcement?


2301/7681 = 0.3 of all docs
split on feature that reduces our uncertainty most
contains "cents" ≥ 2 times
contains "cents" < 2 times
1607/1704 = 0.943
694/5977 = 0.116
contains "versus" ≥ 2 times
contains "versus" < 2 times
contains "net" ≥ 1 time
contains "net" < 1 time
1398/1403 = 0.996
"yes"
209/301 = 0.694
422/541 = 0.780
272/5436 = 0.050
"no"

Booleans, Nominals, Ordinals, and Reals

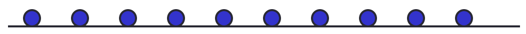
- Consider attribute value differences:

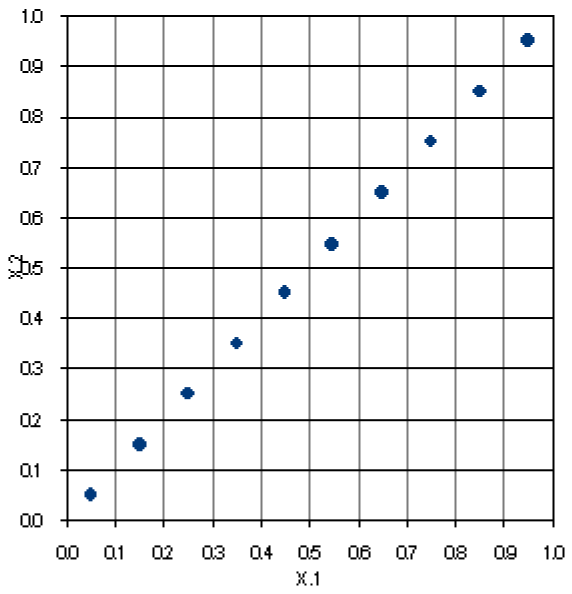
  $(x_i - x'_i)$: what does subtraction do?

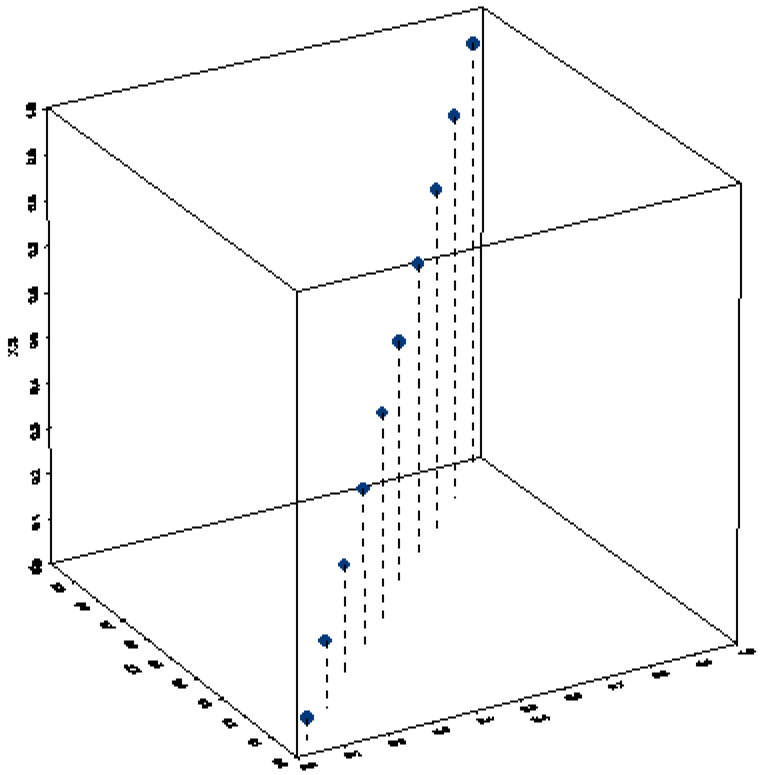
- Reals: easy! full continuum of differences
- Integers: not bad: discrete set of differences
- Ordinals: not bad: discrete set of differences
- Booleans: awkward: hamming distances 0 or 1
- Nominals? not good! recode as Booleans?

"Curse of Dimensionality""

- Pictures on previous slides showed 2-dimensional data
- What happens with lots of dimensions?
- 10 training samples cover the space less & less well …

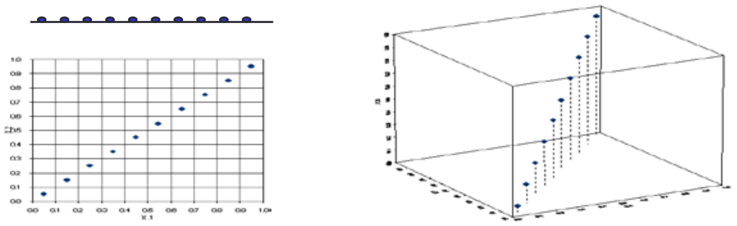






"Curse of Dimensionality""

- Pictures on previous slides showed 2-dimensional data
- What happens with lots of dimensions?
- 10 training samples cover the space less & less well …



- A deeper perspective on this:
  - Random points chosen in a high-dimensional space tend to all be pretty much equidistant from one another!
    - (Proof: in 1000 dimensions, the squared distance between two random points is a sample variance of 1000 coordinate distances. Since 1000 is large, this sample variance is usually close to the true variance.)
  - So each test example is about equally close to most training examples.
  - We need a lot of training examples to expect one that is unusually close to the test example.

images thanks to Charles Annis

"Curse of Dimensionality""

- also, with lots of dimensions/attributes/features, the irrelevant ones may overwhelm the relevant ones:

$$d(x,x') = \sqrt{\sum_{i=1}^{relevant} (x_i - x'_i)^2 + \sum_{j=1}^{irrelevant} (x_j - x'_j)^2}$$

- So the ideas from previous slides grow in importance:
  - feature weights (scaling)
    - feature selection (try to identify & discard irrelevant features)
    - but with lots of features, some irrelevant ones will probably accidentally look relevant on the training data
  - smooth by allowing more neighbors to vote (e.g., larger k)

slide thanks to Rich Caruana (modified)

Advantages of Memory-Based Methods

- Lazy learning: don’t do any work until you know what you want to predict (and from what variables!)
  - never need to learn a global model
  - many simple local models taken together can represent a more complex global model
- Learns arbitrarily complicated decision boundaries
- Very efficient cross-validation
- Easy to explain to users how it works
  - … and why it made a particular decision!
- Can use ***any*** distance metric: string-edit distance, …
  - handles missing values, time-varying distributions, ...

Weaknesses of Memory-Based Methods

- Curse of Dimensionality
  - often works best with 25 or fewer dimensions
- Classification runtime scales with training set size
  - clever indexing may help (K-D trees? locality-sensitive hash?)
  - large training sets will not fit in memory
- Sometimes you wish NN stood for "neural net" instead of "nearest neighbor" ☺
  - Simply averaging nearby training points isn't very subtle
  - Naive distance functions are overly respectful of the input encoding
- For regression (predict a number rather than a class), the extrapolated surface has discontinuities

Current Research in MBL

- Condensed representations to reduce memory requirements and speed-up neighbor finding to scale to 10^6–10^{12} cases
- Learn better distance metrics
- Feature selection
- Overfitting, VC-dimension, ...
- MBL in higher dimensions
- MBL in non-numeric domains:
  - Case-Based or Example-Based Reasoning
  - Reasoning by Analogy

References

- *Locally Weighted Learning* by Atkeson, Moore, Schaal
- *Tuning Locally Weighted Learning* by Schaal, Atkeson, Moore

Closing Thought

- In many supervised learning problems, all the information you ever have about the problem is in the training set.
- Why do most learning methods discard the training data after doing learning?
- Do neural nets, decision trees, and Bayes nets capture *all* the information in the training set when they are trained?
- Need more methods that combine MBL with these other learning methods.
  - to improve accuracy
  - for better explanation
  - for increased flexibility