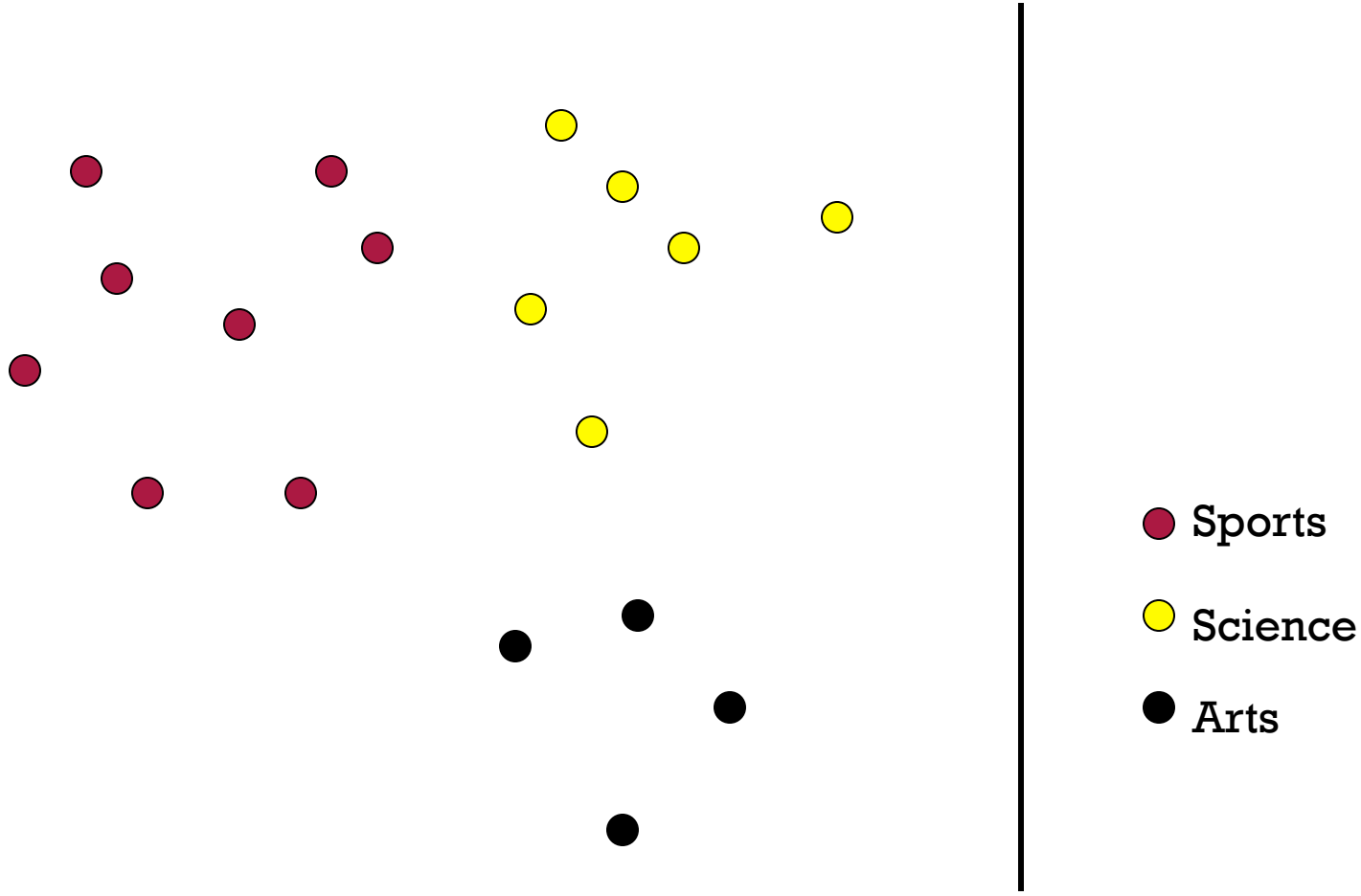
k-NN (k-Nearest Neighbor) Classifier

Aarti Singh

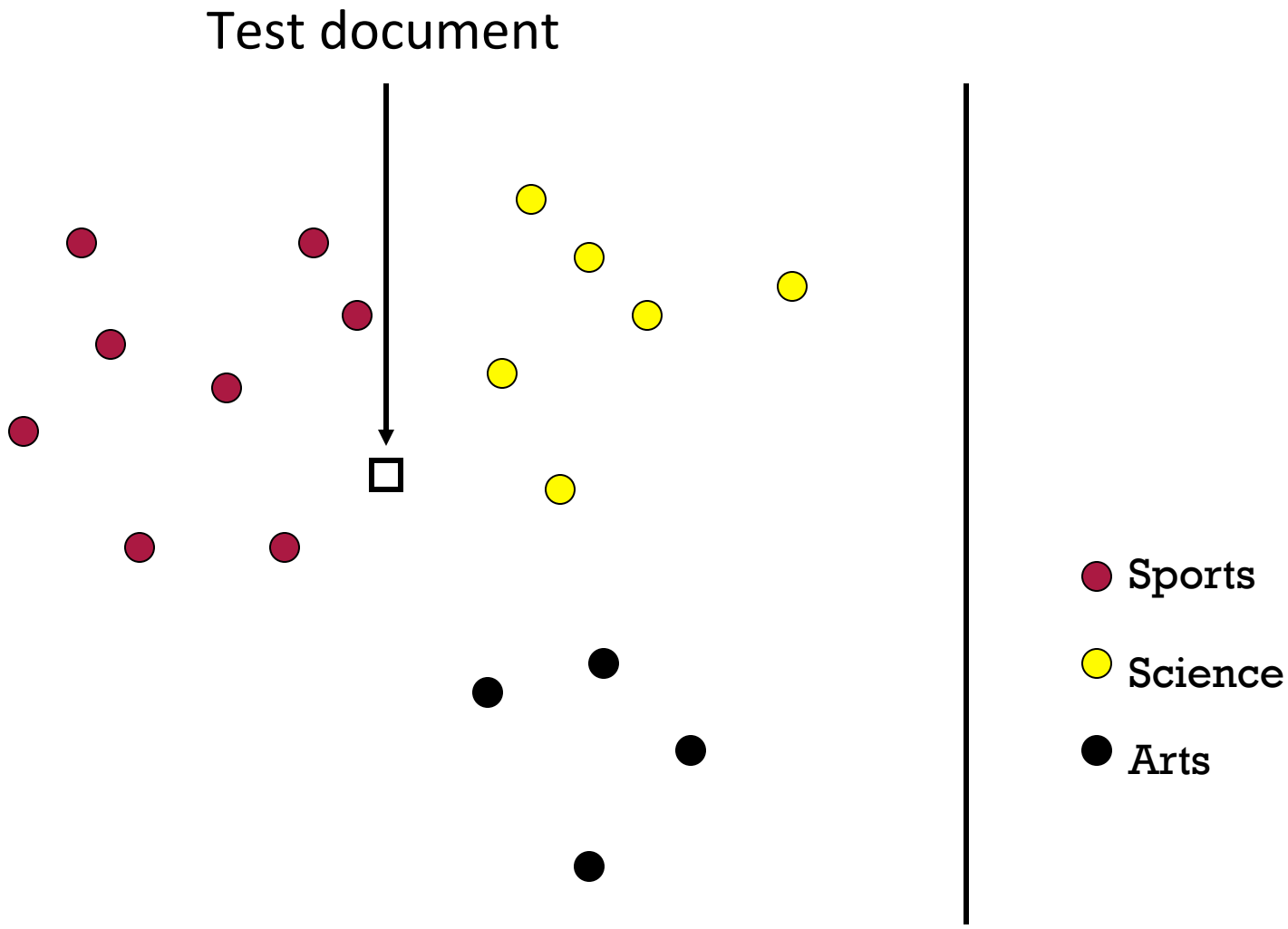
Co-instructor: Barnabas Poczos

Machine Learning 10-401
Feb 18 , 2016

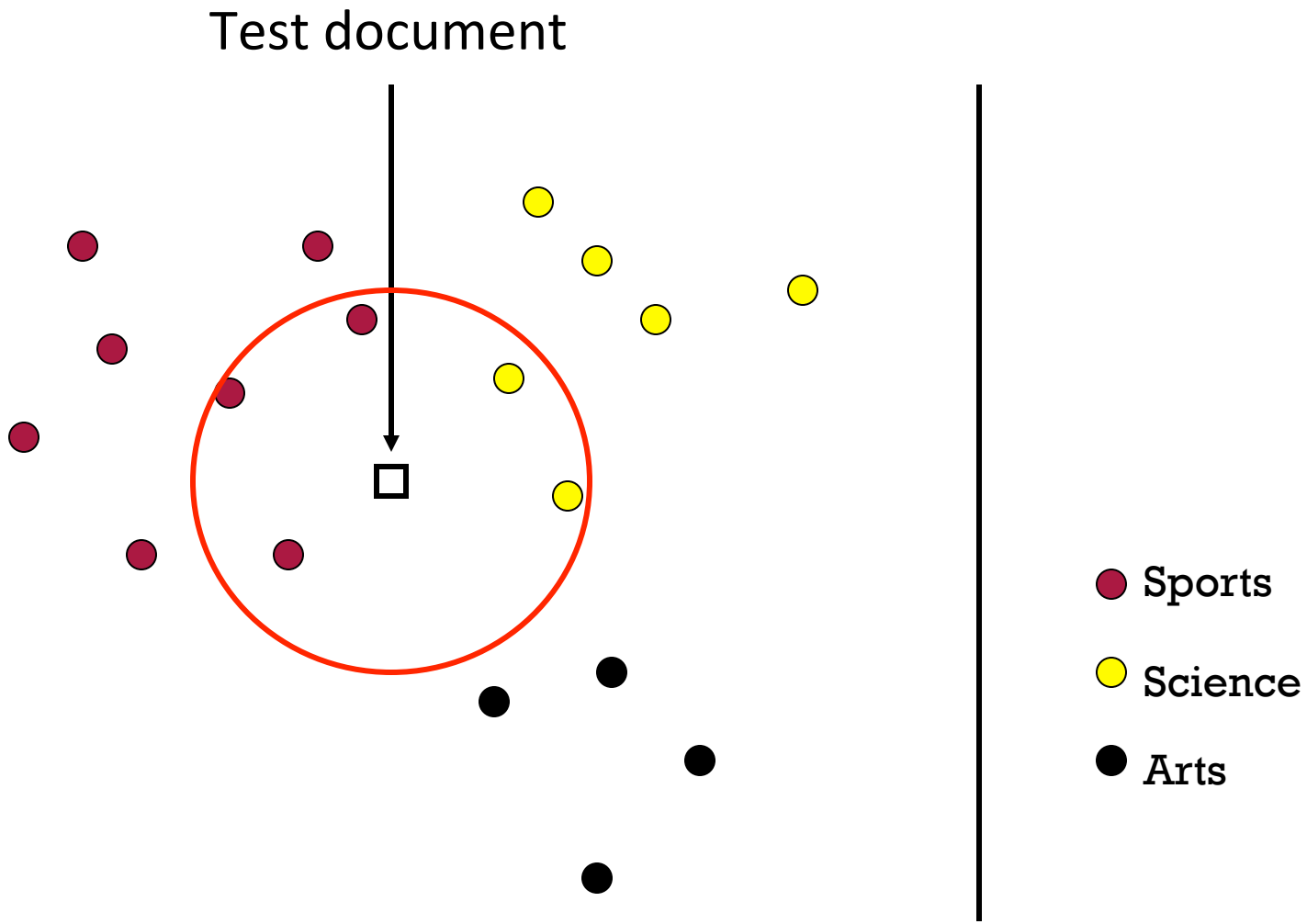
k-NN classifier



k-NN classifier



k-NN classifier (k=5)



What should we predict? … Average? Majority? Why?

k-NN classifier

- Optimal Classifier: $f^*(x) = \arg\max_y P(y|x)$
  $= \arg\max_y P(x|y)P(y)$

- k-NN Classifier: $\hat{f}_{kNN}(x) = \arg\max_y \hat{P}_{kNN}(x|y)\hat{P}(y)$
  $= \arg\max_y k_y$

$$\hat{P}_{kNN}(x|y) = \frac{k_y}{n_y}$$

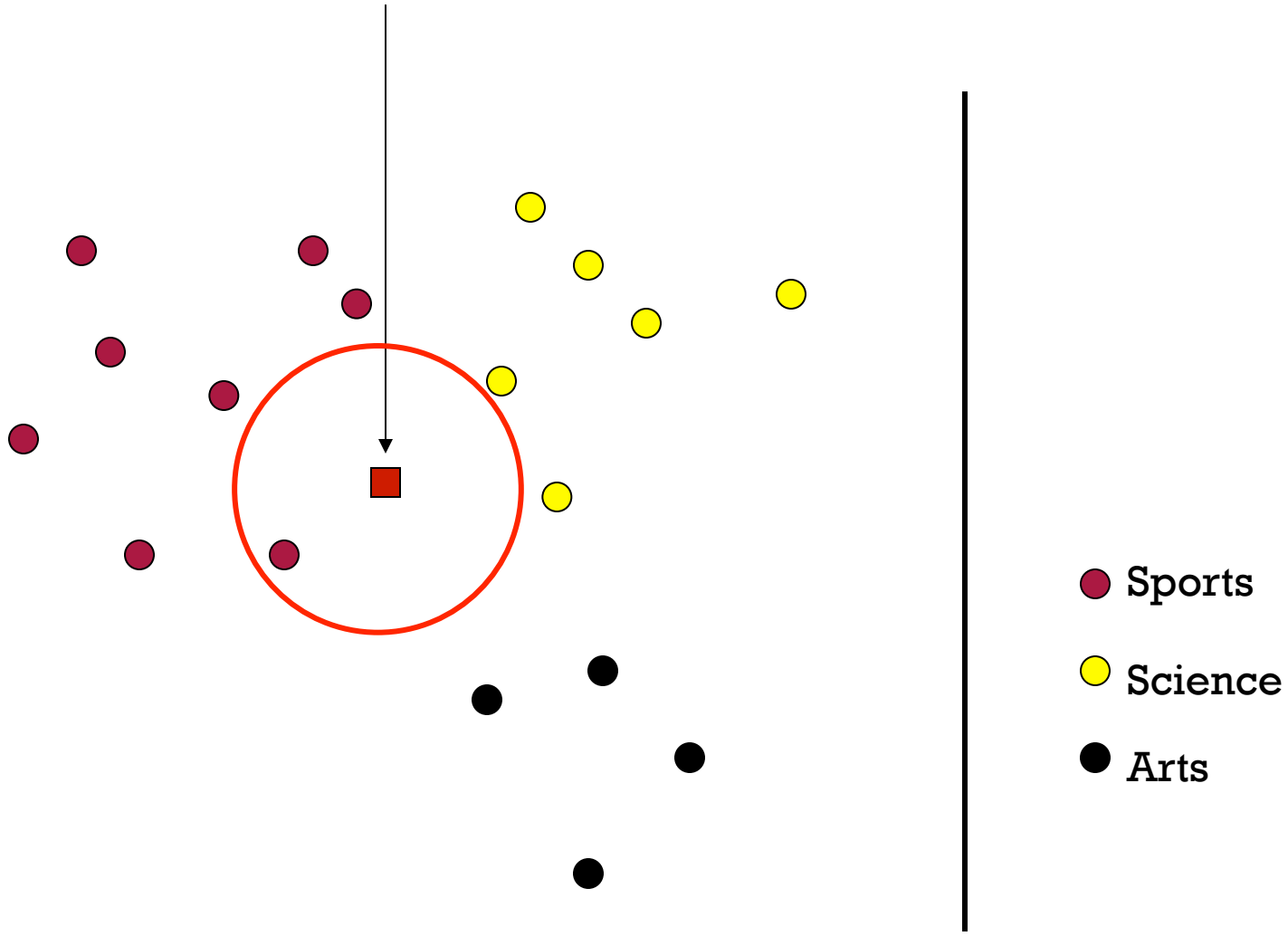
→ # training pts of class y amongst k NNs of x

→ # total training pts of class y

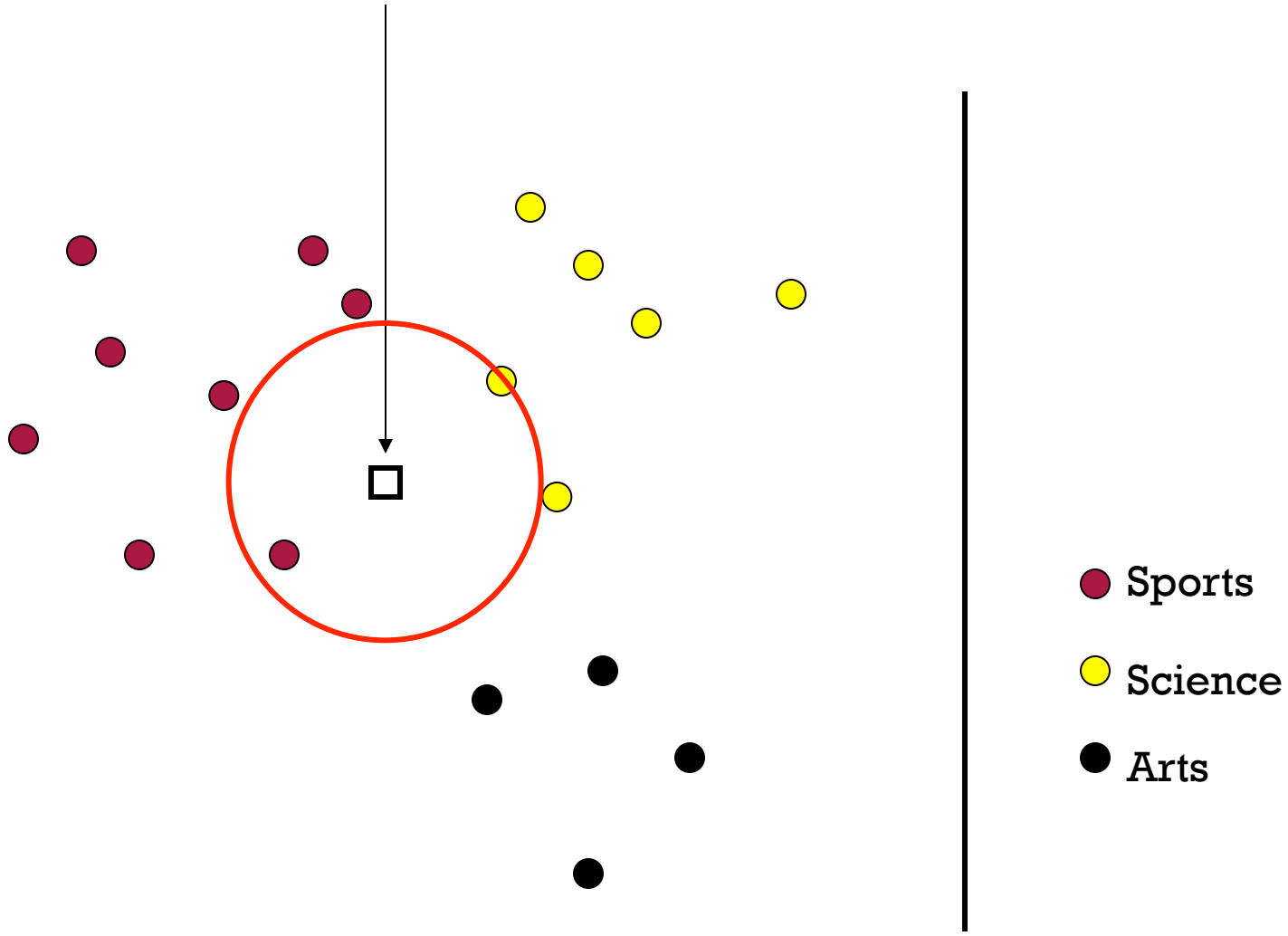
$$\sum_y k_y = k$$

$$\hat{P}(y) = \frac{n_y}{n}$$

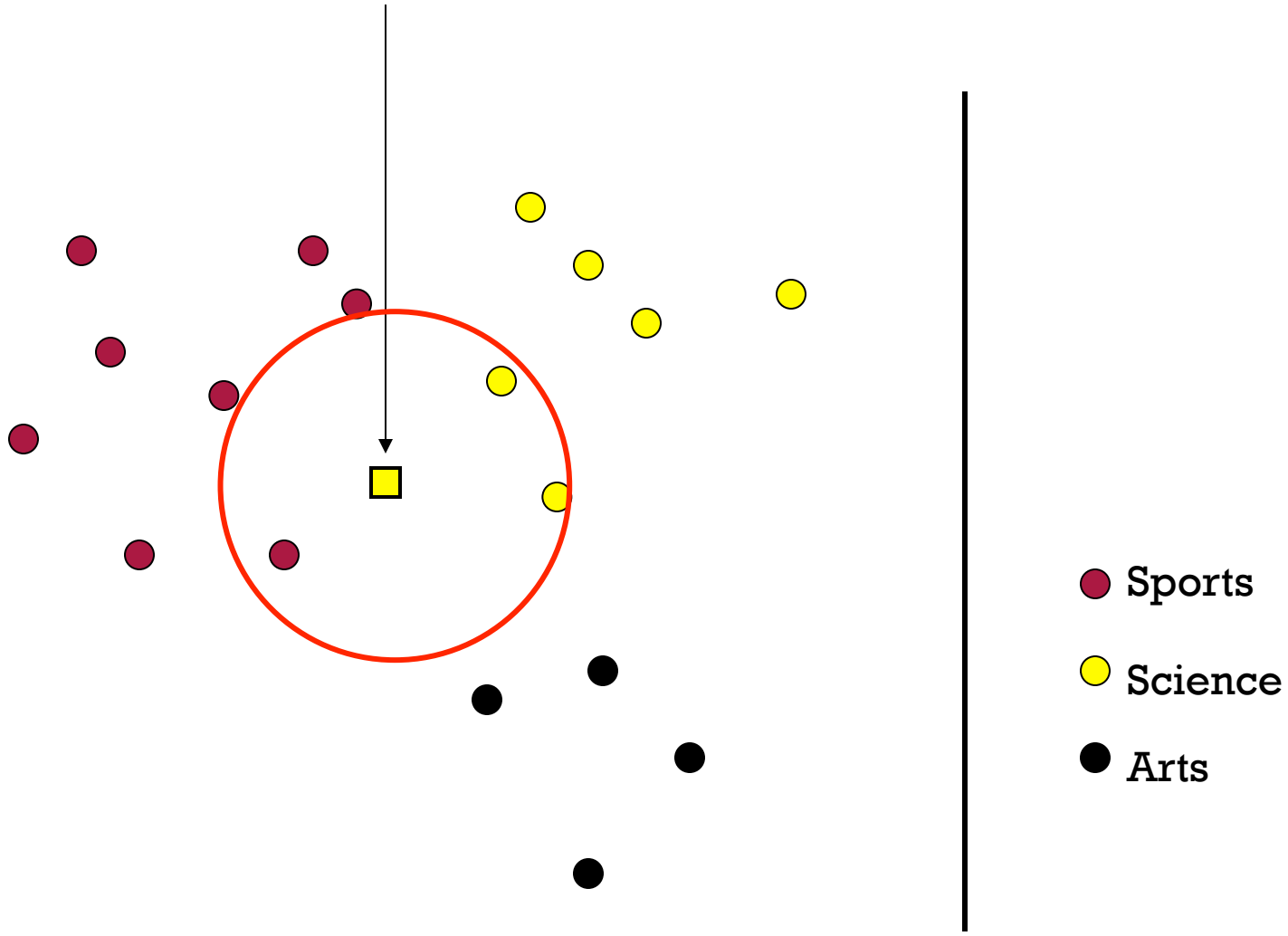
1-Nearest Neighbor (kNN) classifier



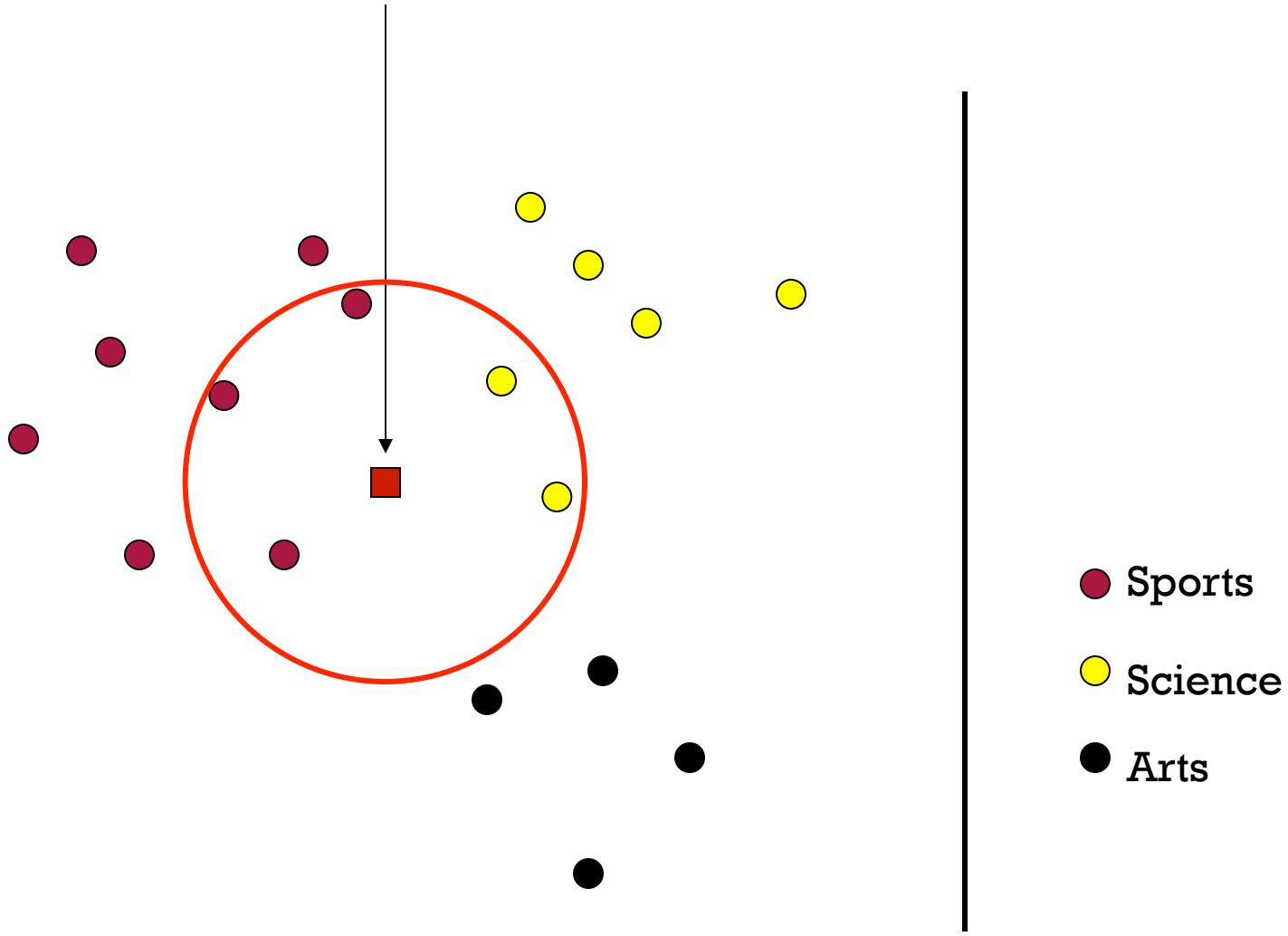
2-Nearest Neighbor (kNN) classifier



3-Nearest Neighbor (kNN) classifier

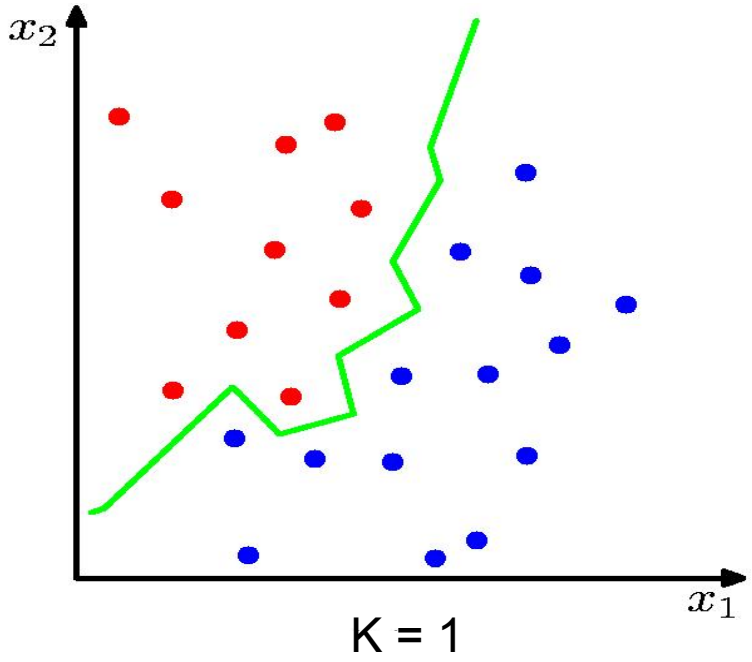


5-Nearest Neighbor (kNN) classifier



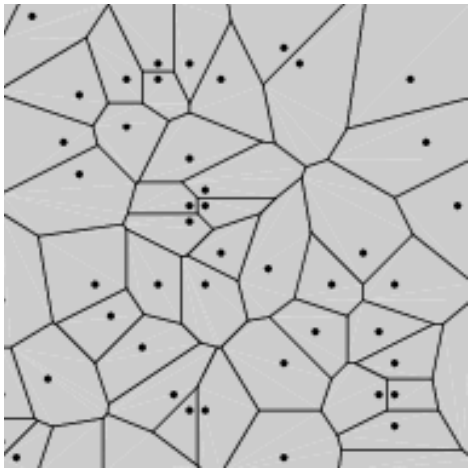
What is the best k?

1-NN classifier decision boundary



K = 1

Voronoi Diagram



As k increases, boundary becomes smoother (less jagged).

What is the best k?

Approximation vs. Stability Tradeoff

- Larger K => predicted label is more stable
- Smaller K => predicted label can approximate best classifier well

Non-parametric methods

Aka Instance-based/Memory-based learners

- Decision Trees
- k-Nearest Neighbors

Parametric methods

- Assume some model (Gaussian, Bernoulli, Multinomial, logistic, network of logistic units, Linear, Quadratic) with fixed number of parameters
  - Gaussian Bayes, Naïve Bayes, Logistic Regression, Perceptron, Neural Networks

- Estimate parameters ($\mu, \sigma^2, \theta, w, \beta$) using MLE/MAP and plug in

- **Pro** – need few data points to learn parameters
- **Con** – Strong distributional assumptions, not satisfied in practice

Non-Parametric methods

- Typically don't make any distributional assumptions
- As we have more data, we should be able to learn more complex models
- Let number of parameters scale with number of training data

- Some nonparametric methods
  - Decision Trees
  - k-NN (k-Nearest Neighbor) Classifier

Summary

- Parametric vs Nonparametric approaches
  - Nonparametric models place very mild assumptions on the data distribution and provide good models for complex data

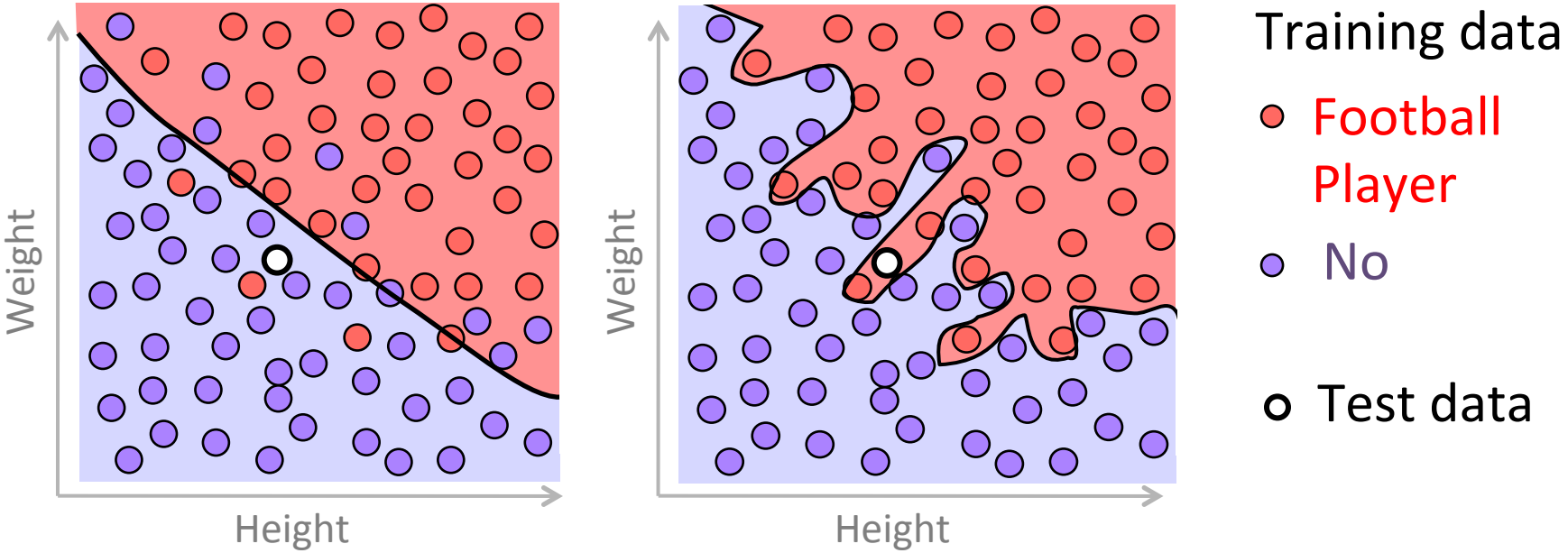
    Parametric models rely on very strong (simplistic) distributional assumptions

  - Nonparametric models requires storing and computing with the entire data set.

    Parametric models, once fitted, are much more efficient in terms of storage and computation.

Judging Overfitting

Training Data vs. Test Data



- A good machine learning algorithm
  - Does not **overfit** training data
  - **Generalizes** well to test data

Training error

- Training error of a classifier f

$$\frac{1}{n}\sum_{i=1}^{n} 1_{f(X_i)\neq Y_i}$$

Training Data
$\{X_i, Y_i\}_{i=1}^{n}$

- What about test error?

  Can't compute it.

- How can we know classifier is not overfitting?

  Hold-out or Cross-validation

Hold-out method

Can judge test error by using an independent sample of data.

Hold – out procedure:

n data points available $D \equiv \{X_i, Y_i\}_{i=1}^{n}$

1) Randomly split into two sets (preserving label proportion):

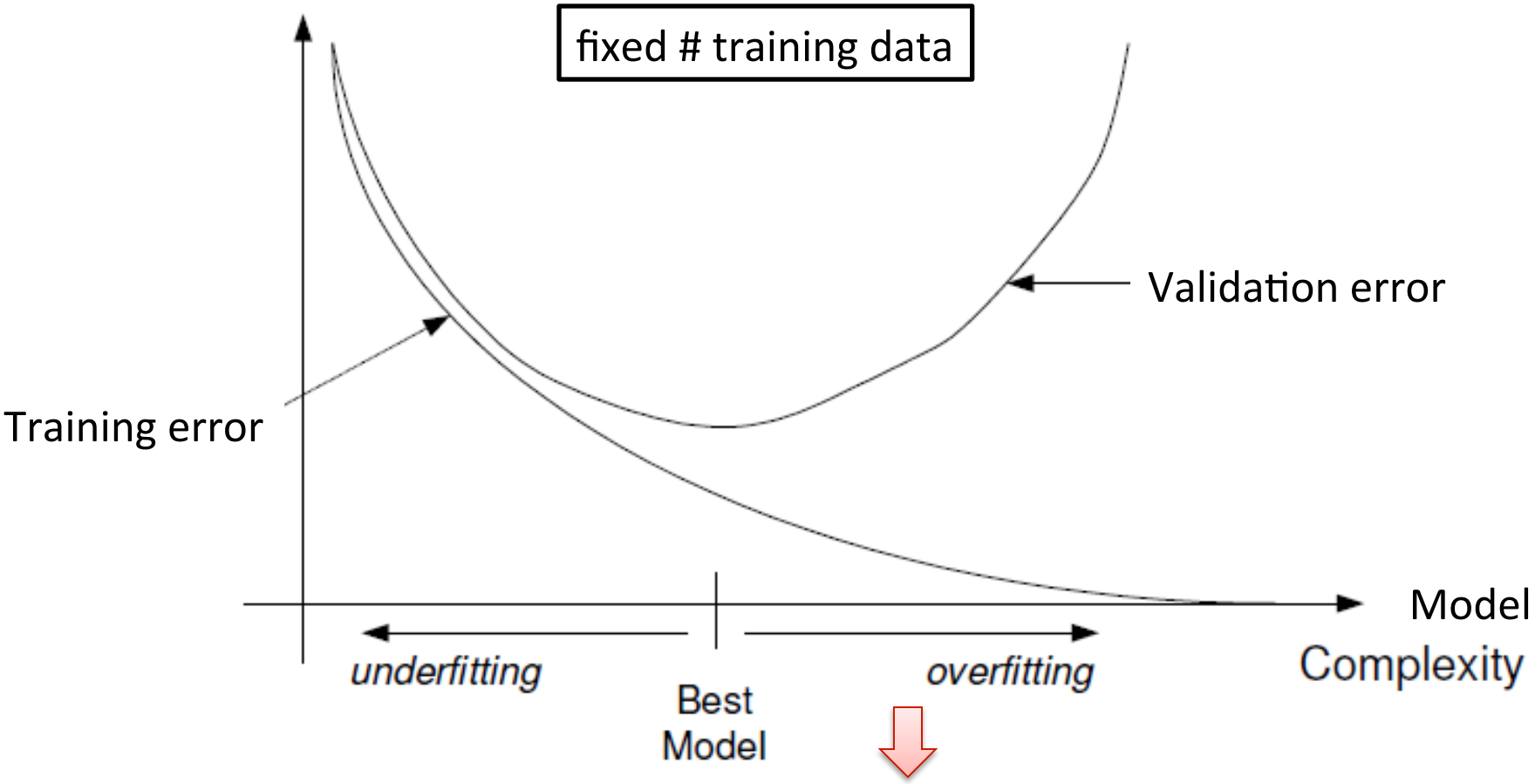
Training dataset Validation/Hold-out dataset

$$D_T = \{X_i, Y_i\}_{i=1}^{m} \qquad D_V = \{X_i, Y_i\}_{i=m+1}^{n}$$

often m = n/2

2) Train classifier on D_T. Report error on validation dataset D_V.
Overfitting if validation error is much larger than training error

Training vs. Validation Error



Training error is no longer a good indicator of validation or test error

Hold-out method

Drawbacks:

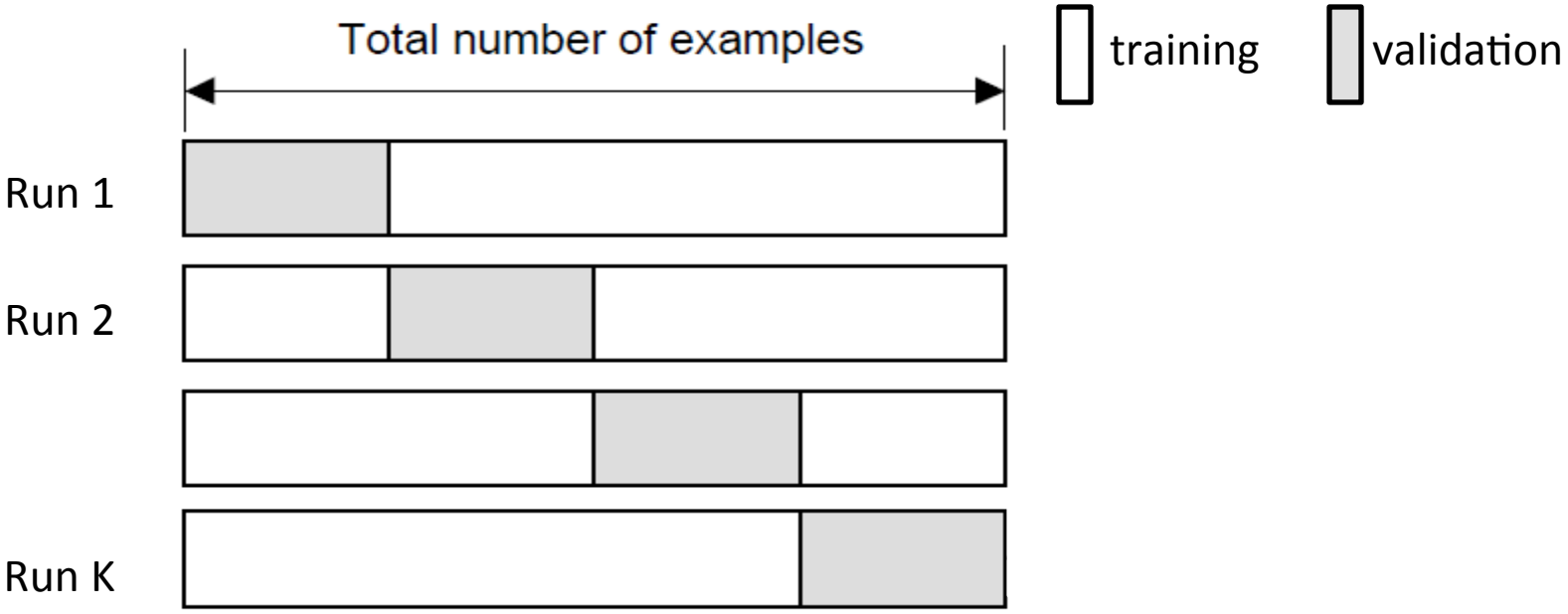
- May not have enough data to afford setting one subset aside for getting a sense of generalization abilities
- Validation error may be misleading (bad estimate of test error) if we get an "unfortunate" split

Limitations of hold-out can be overcome by a family of random sub-sampling methods at the expense of more computation.

Cross-validation

K-fold cross-validation

Create K-fold partition of the dataset.
Do K runs: train using K-1 partitions and calculate validation error on remaining partition (rotating validation partition on each run).
Report average validation error

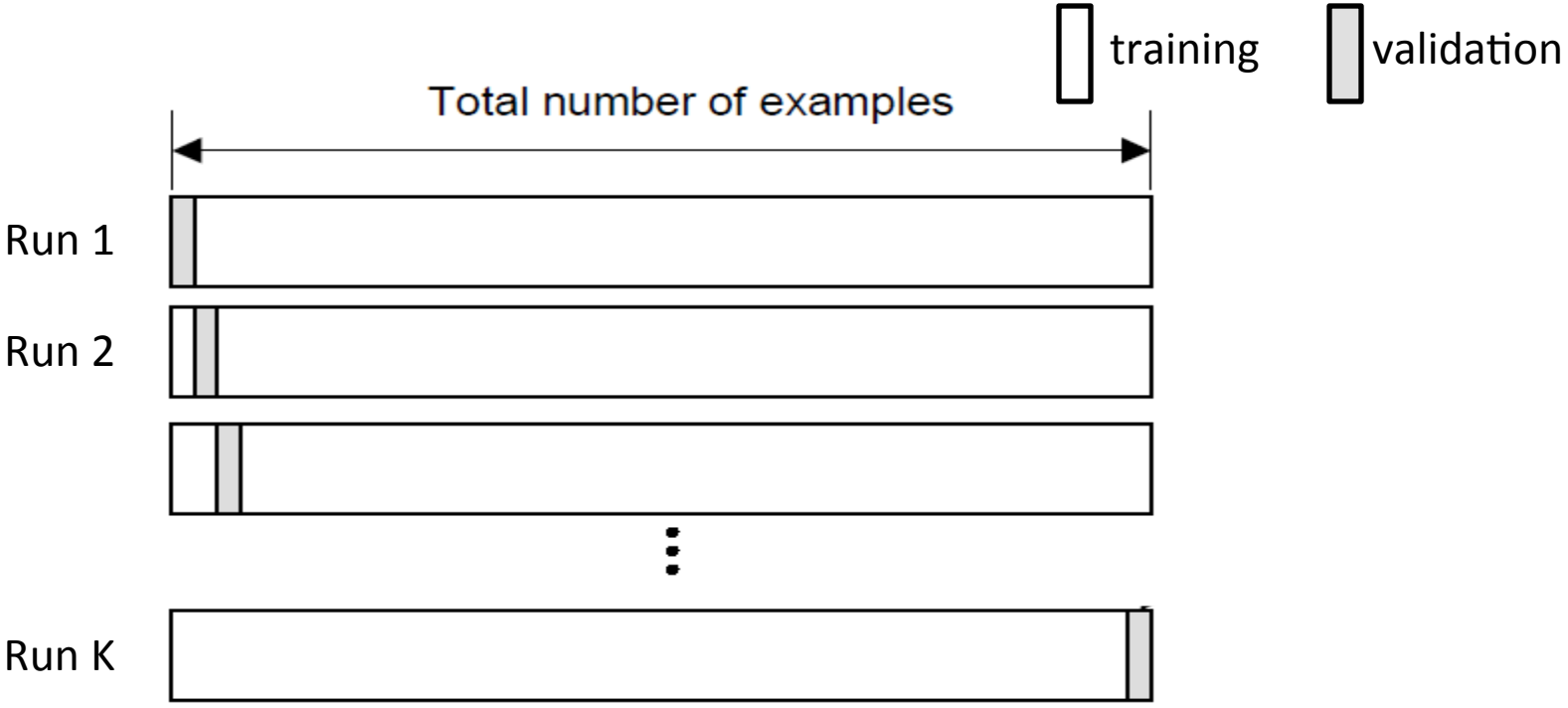


Cross-validation

Leave-one-out (LOO) cross-validation

Special case of K-fold with K=n partitions

Equivalently, train on n-1 samples and validate on only one sample per run for n runs



Cross-validation

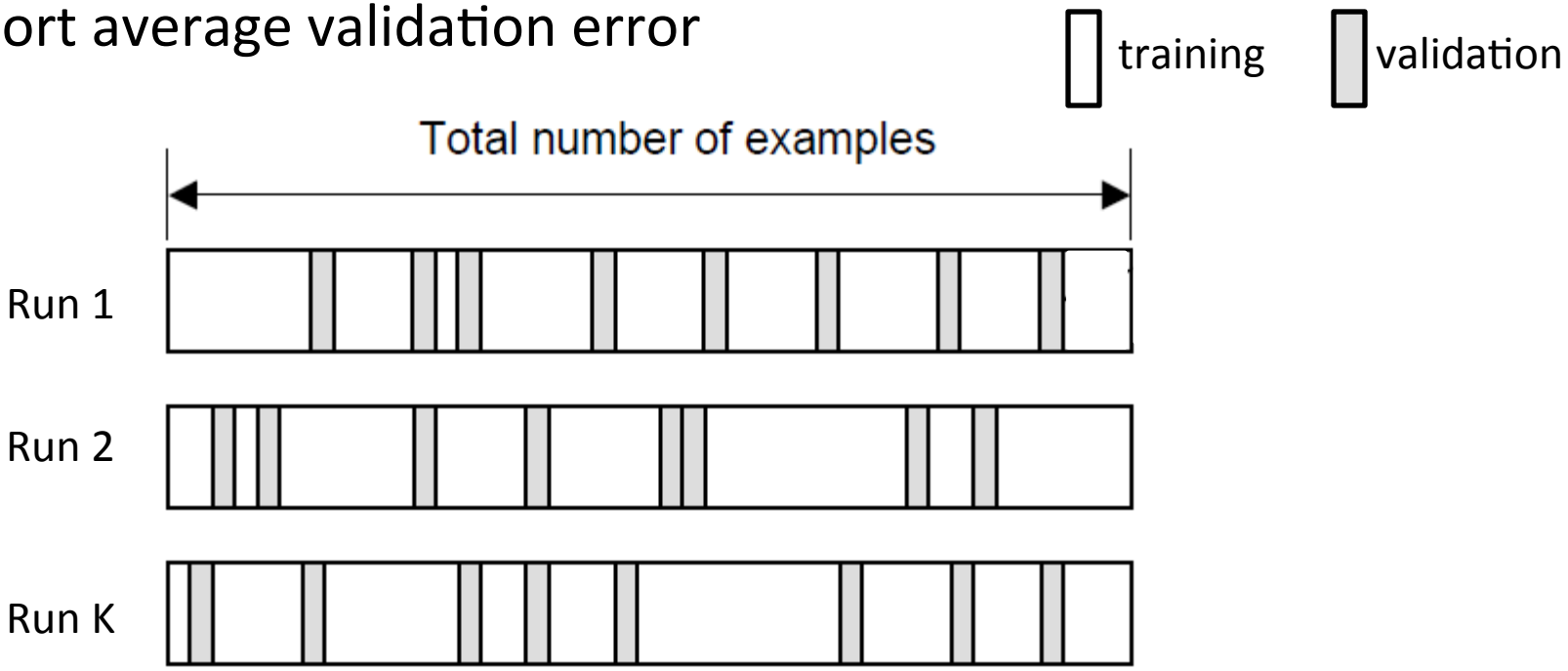
Random subsampling

Randomly subsample a fixed fraction αn ($0 < \alpha < 1$) of the dataset for validation.

Compute validation error with remaining data as training data.

Repeat K times

Report average validation error



Practical Issues in Cross-validation

How to decide the values for K and α ?

- Large K

  + Validation error can approximate test error well

  - Observed validation error will be unstable (few validation pts)

  - The computational time will be very large as well (many experiments)

- Small K

  + The # experiments and, therefore, computation time are reduced

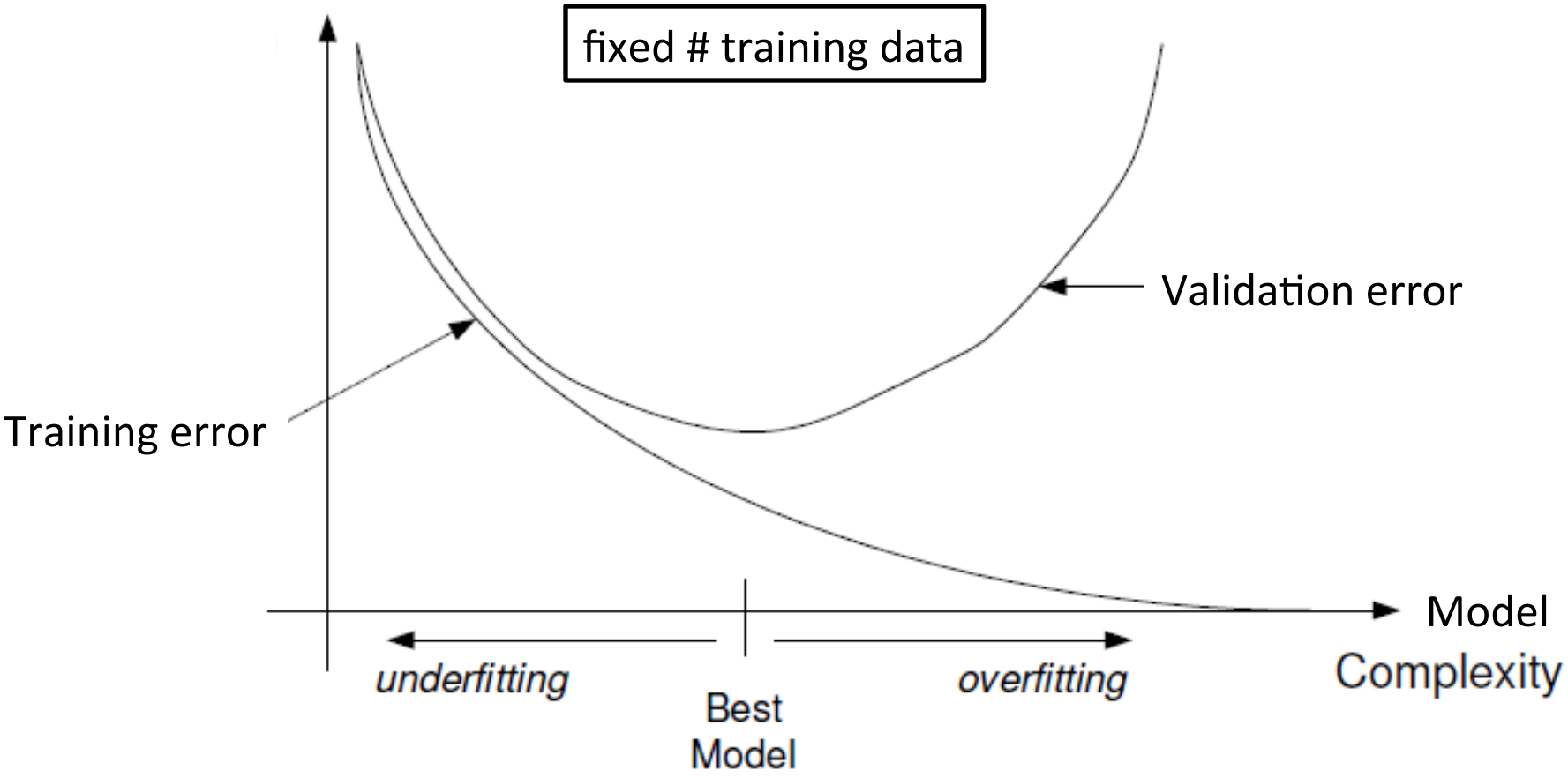
  + Observed validation error will be stable (many validation pts)

  - Validation error cannot approximate test error well

Common choice: K = 10, α = 0.1 ☺

Model selection

Effect of Model Complexity



Can we select good models using hold-out or cross-validation?

Examples of Model Spaces

Model Spaces with increasing complexity:

- Nearest-Neighbor classifiers with increasing neighborhood sizes k = 1,2,3,…

  Small neighborhood => Higher complexity

- Decision Trees with increasing depth k or with k leaves

  Higher depth/ More # leaves => Higher complexity

- Neural Networks with increasing layers or nodes per layer

  More layers/Nodes per layer => Higher complexity

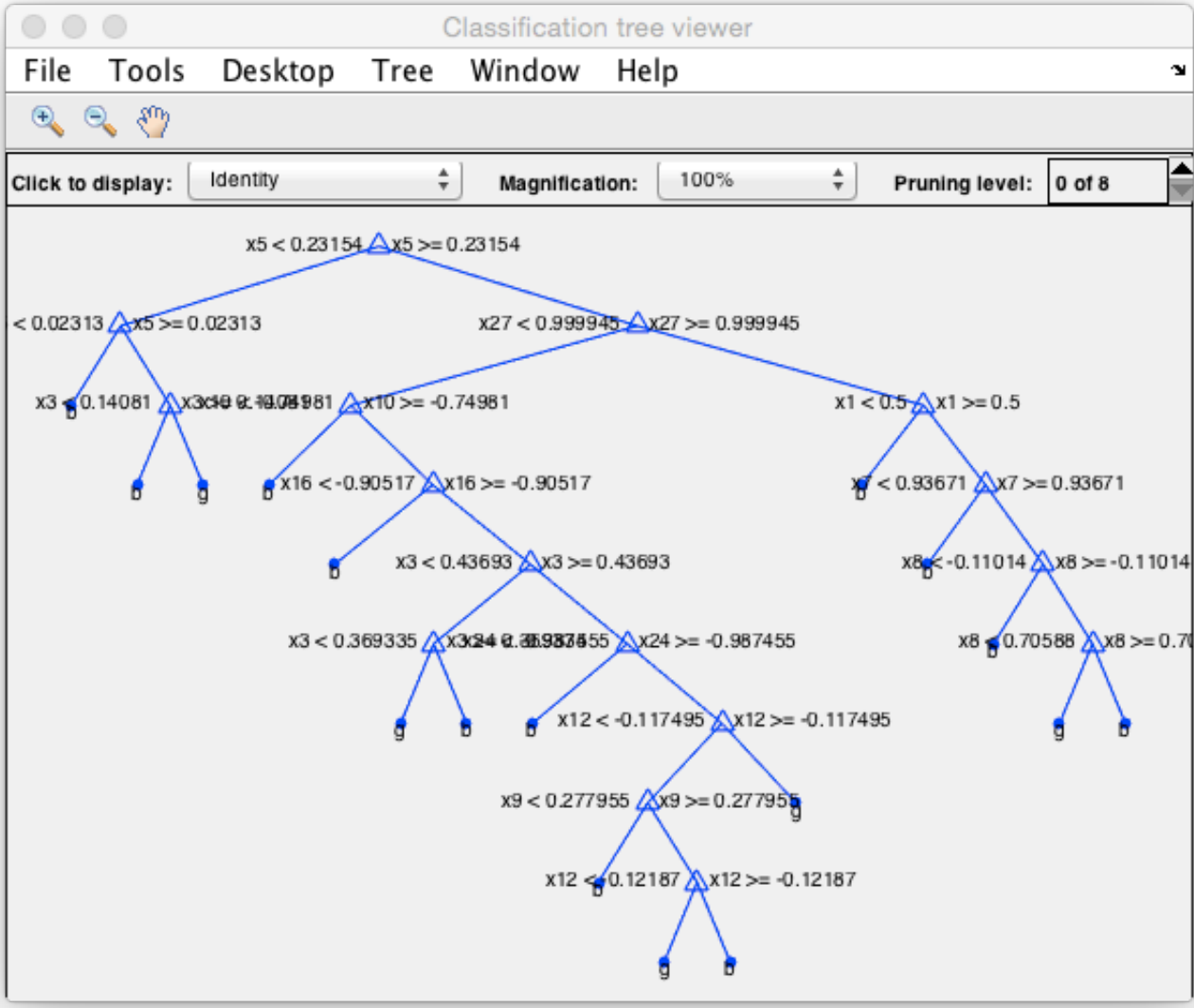
- MAP estimates with stronger priors (larger hyper-parameters β_H, β_T for Beta distribution or smaller variance for Gaussian prior)

How can we select the right complexity model ?

Model selection using Hold-out/ Cross-validation

- Train models of different complexities and evaluate their validation error using hold-out or cross-validation

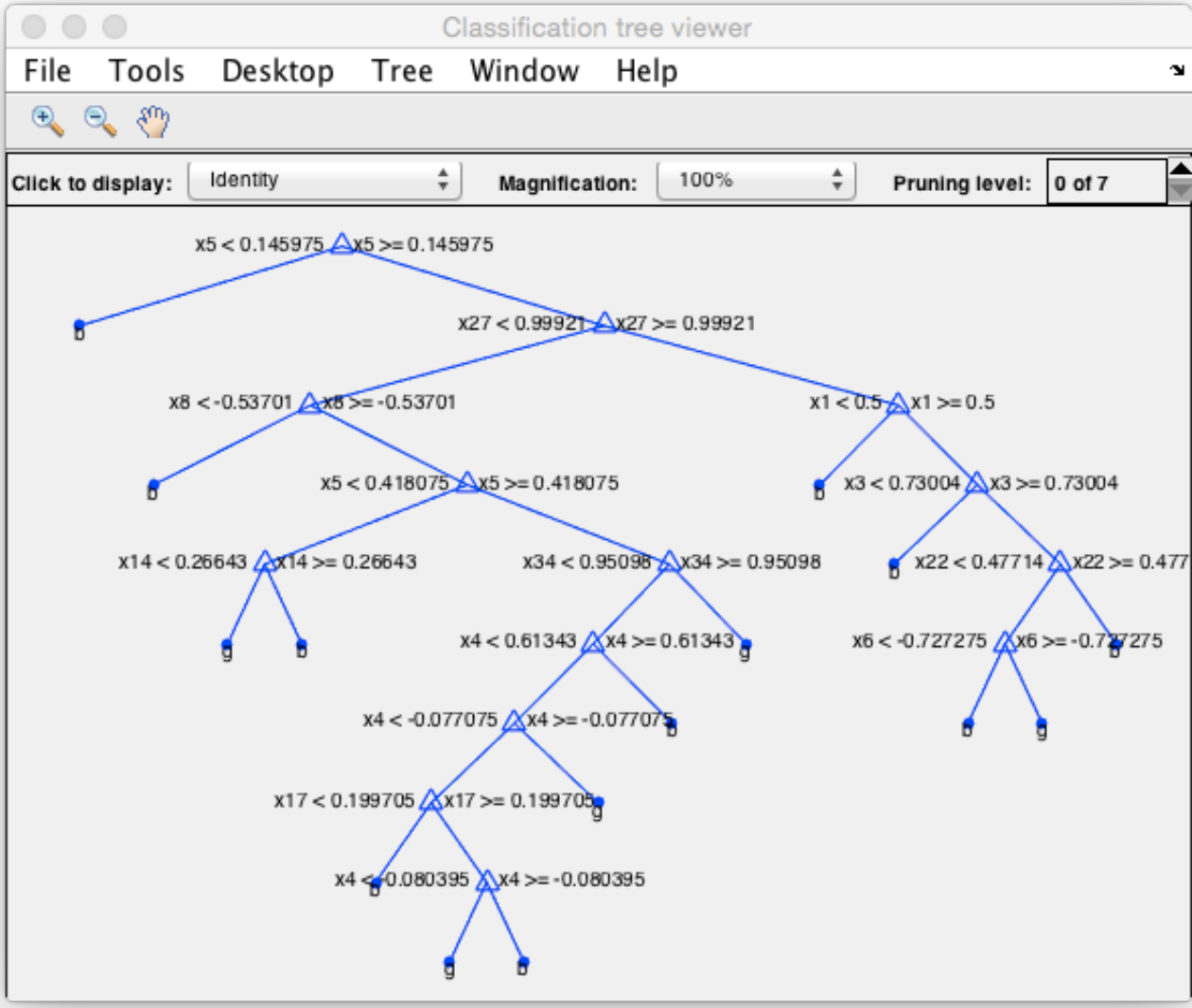
- Pick model with smallest validation error (averaged over different runs for cross-validation)




```
load ionosphere
% UCI dataset
% 34 features, 351 samples
% binary classification
rng(100)

%Defaulty MinLeafSize = 1
tc = fitctree(X,Y);
cvmodel = crossval(tc);
view(cvmodel.Trained{1},'Mode','graph')
kfoldLoss(cvmodel)
```

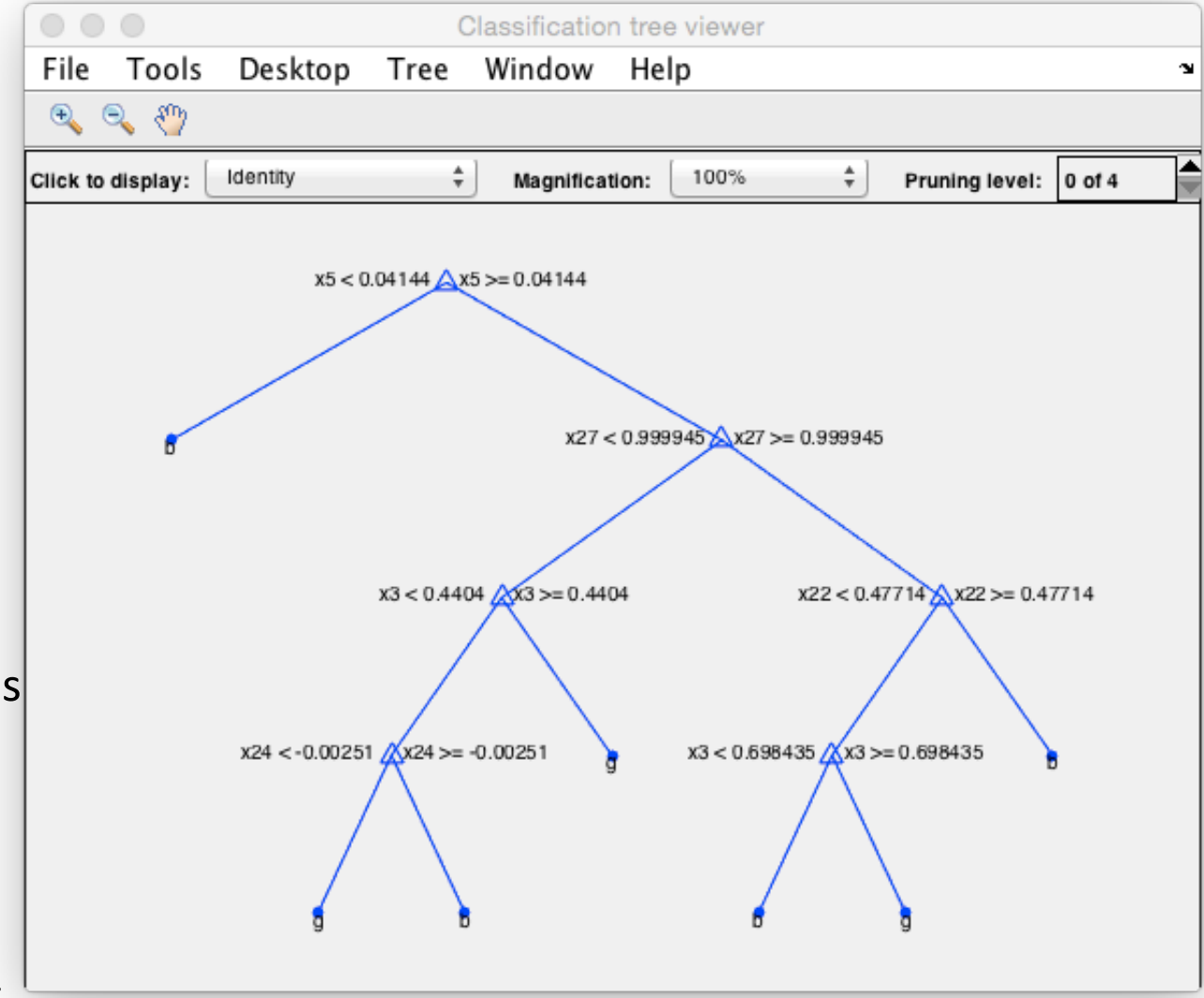
Validation error = 0.1254




```
load ionosphere
% UCI dataset
% 34 features, 351 samples
% binary classification
rng(100)

%Defaulty MinLeafSize = 1
tc = fitctree(X,Y, 'MinLeafSize',2);
cvmodel = crossval(tc);
view(cvmodel.Trained{1},'Mode','graph')
kfoldLoss(cvmodel)
```

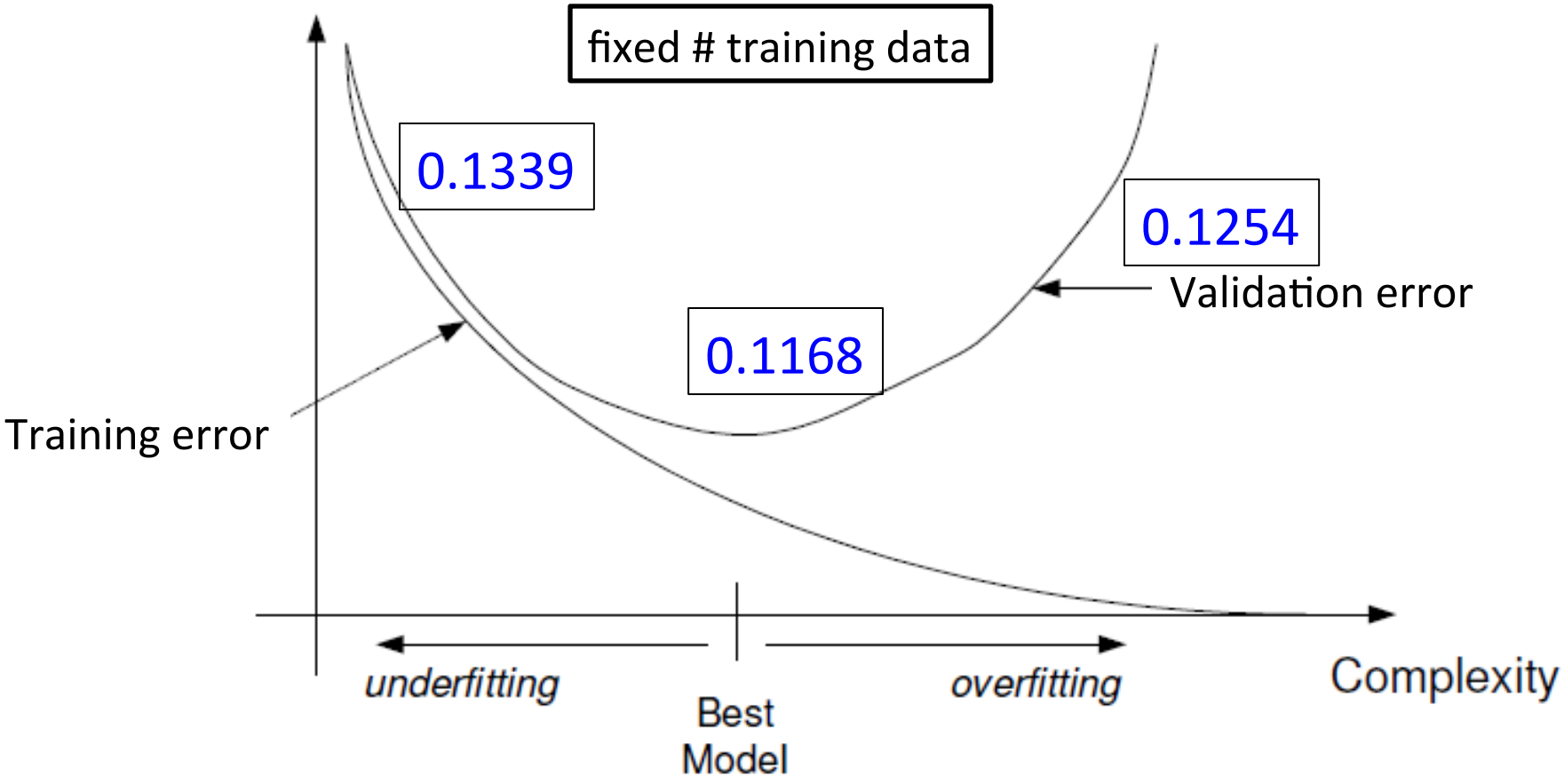
Validation error = 0.1168




```
load ionosphere
% UCI dataset
% 34 features, 351 samples
% binary classification
rng(100)

%Defaulty MinLeafSize = 1
tc = fitctree(X,Y, 'MinLeafSize',10);
cvmodel = crossval(tc);
view(cvmodel.Trained{1},'Mode','graph')
kfoldLoss(cvmodel)
```

Validation error = 0.1339



MinLeafSize 10 MinLeafSize 2 MinLeafSize 1