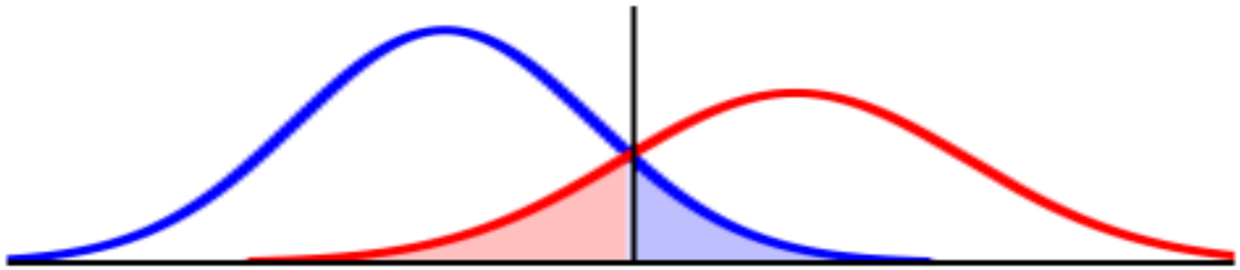
CS178:
Machine Learning & Data Mining



Prof. Alexander Ihler

Fall 2023

Outline

Optimal Decisions (in theory)

Bayes Classifiers

Types of Errors

Training & Validation Data

K-Nearest Neighbor Models

Outline

- Optimal Decisions (in theory)
- Bayes Classifiers
- Types of Errors
- Training & Validation Data
- K-Nearest Neighbor Models

A simple, optimal classifier

- Classifier f(x ; θ)
  - maps observations x to predicted target values
- Simple example
  - Discrete feature x: f(x ; θ) is a contingency table
  - Ex: spam filtering: observe just X_1 = sender in contact list?
- Suppose we knew the true conditional probabilities:
- Best prediction is the most likely target!

"Bayes error rate"

Pr[X=0] * Pr[wrong | X=0] + Pr[X=1] * Pr[wrong | X=1]

= Pr[X=0] * (1- Pr[Y=S | X=0]) + Pr[X=1] * (1-Pr[Y=K | X=1])

Can't do better than this without more information: e.g., more features (email header, body text, etc.)

Feature	spam	keep
X=0	0.6	0.4
X=1	0.1	0.9

A simple classifier from data

- Training data $D=\{x^{(i)},y^{(i)}\}$, Classifier $f(x\ ;\ D)$
  - Discrete feature vector x
  - f(x ; D) is a contingency table
- Ex: Fisher Iris data, one feature
  - X_1 = sepal length (different ranges)
  - How should we make our predictions?
  - One method: just estimate the probabilities?

Sepal length	Iris setosa	Iris virsicolor	Iris virginica
X < 5	21	30	5
5 < X < 6	23	21	30
6 < X < 7	0	16	35
7 < X	0	1	10

$\Rightarrow$

Sepal length	Iris setosa	Iris virsicolor	Iris virginica
X < 5	0.375	0.536	0.089
5 < X < 6	0.311	0.284	0.405
6 < X < 7	0.	0.314	0.686
7 < X	0.	0.091	0.909

(empirically estimated)

Estimating $p(y|X=x)$: "probabilistic" learning
Gives a prediction *and* an (estimated) notion of confidence in that prediction

A simple classifier from data

- Training data $D=\{x^{(i)},y^{(i)}\}$, Classifier $f(x ; D)$
  - Discrete feature vector x
  - f(x ; D) is a contingency table
- Ex: Fisher Iris data, one feature
  - What if we give more information?
  - Let's reduce the ranges of the table entries:

Two sources of error!

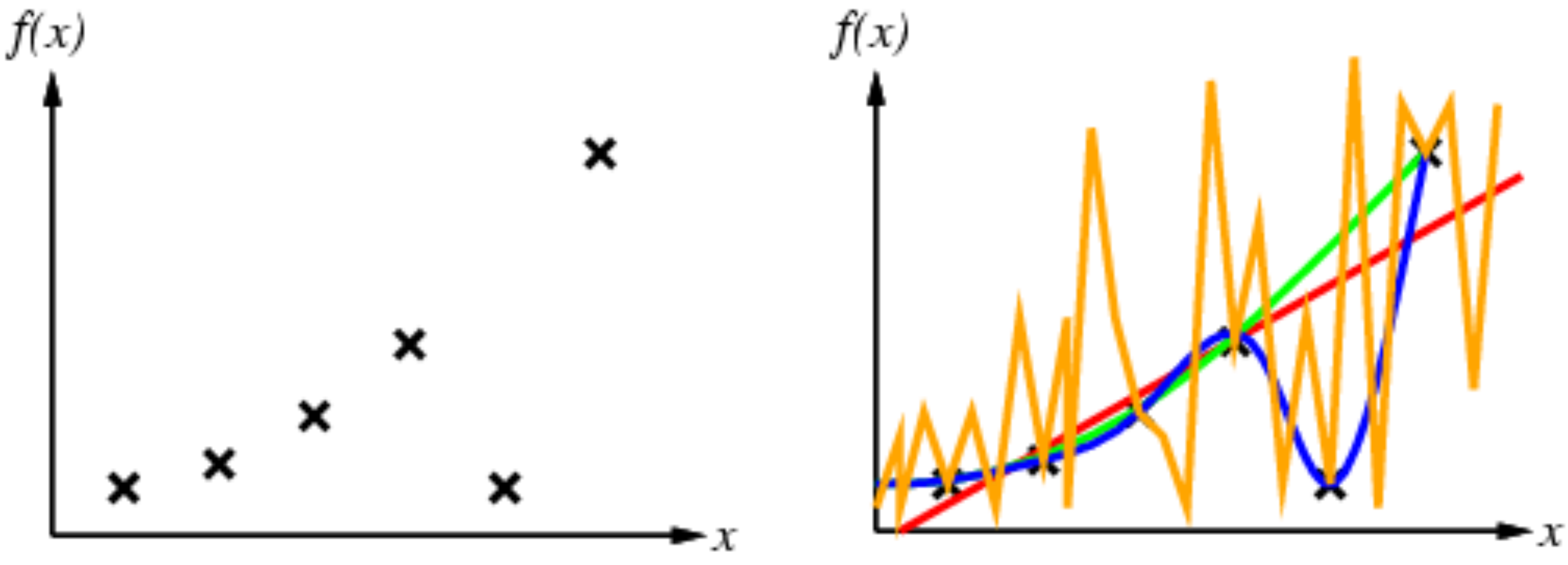
- Bayes error rate
  (improve with more info in X)
- Mis-estimating probability
  (improve with more data)

Sepal length	Iris setosa	Iris virsicolor	Iris virginica
…			
5.25	0.57	0.07	0.36
5.5	0.09	0.48	0.43
5.75	0.08	0.38	0.54
…			

Sepal length	Iris setosa	Iris virsicolor	Iris virginica
…			
5.48	1	0	0
5.5	0	0	1
5.52	0	0	0
…			

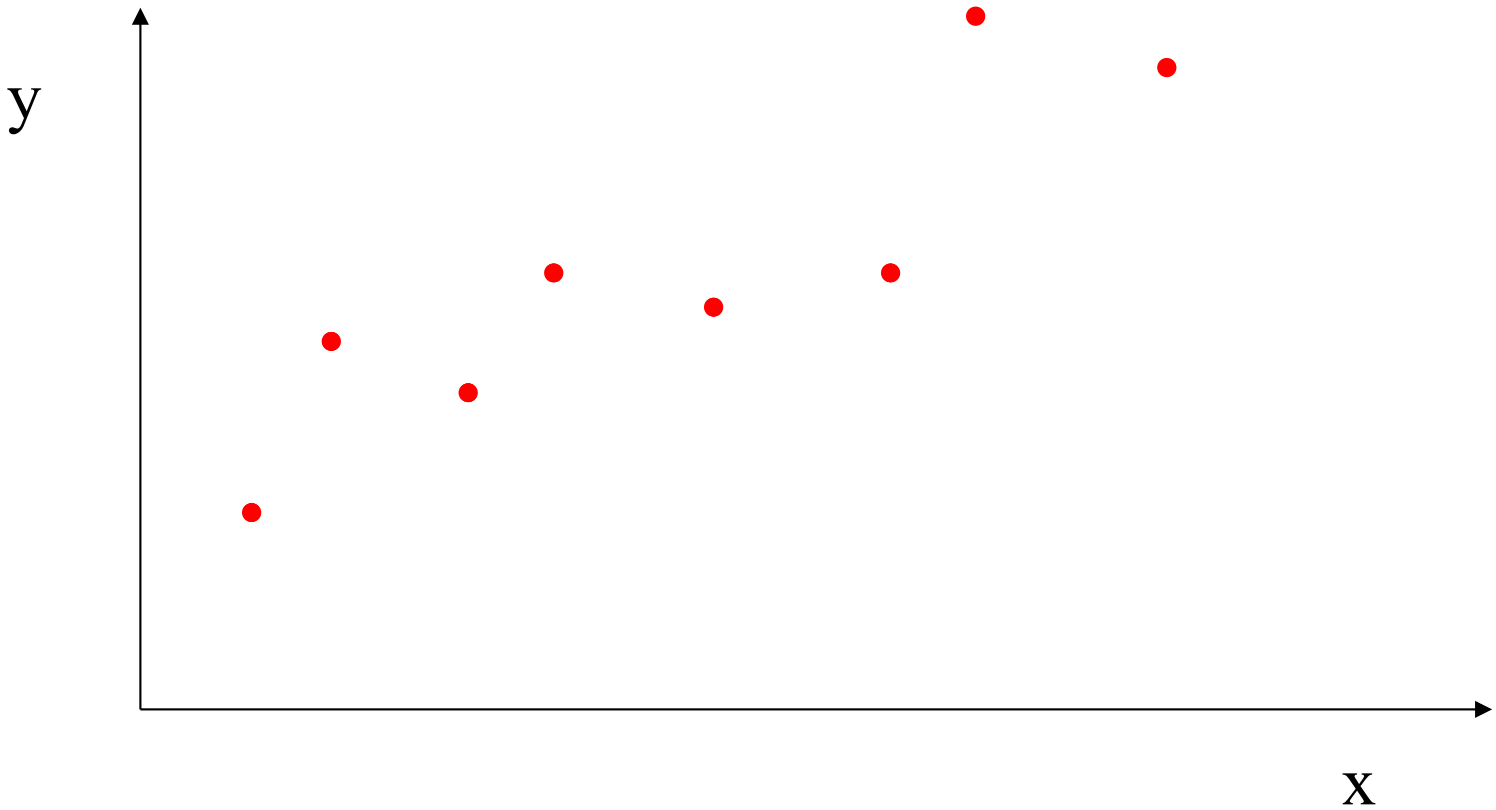
Inductive bias

- Allow us to extend observed data to unobserved ones
  - Interpolation / extrapolation
- What relationships do we expect in the data?
  - A (perhaps *the*) key question in ML models
  - Usually, data pull us away from assumptions only with evidence!



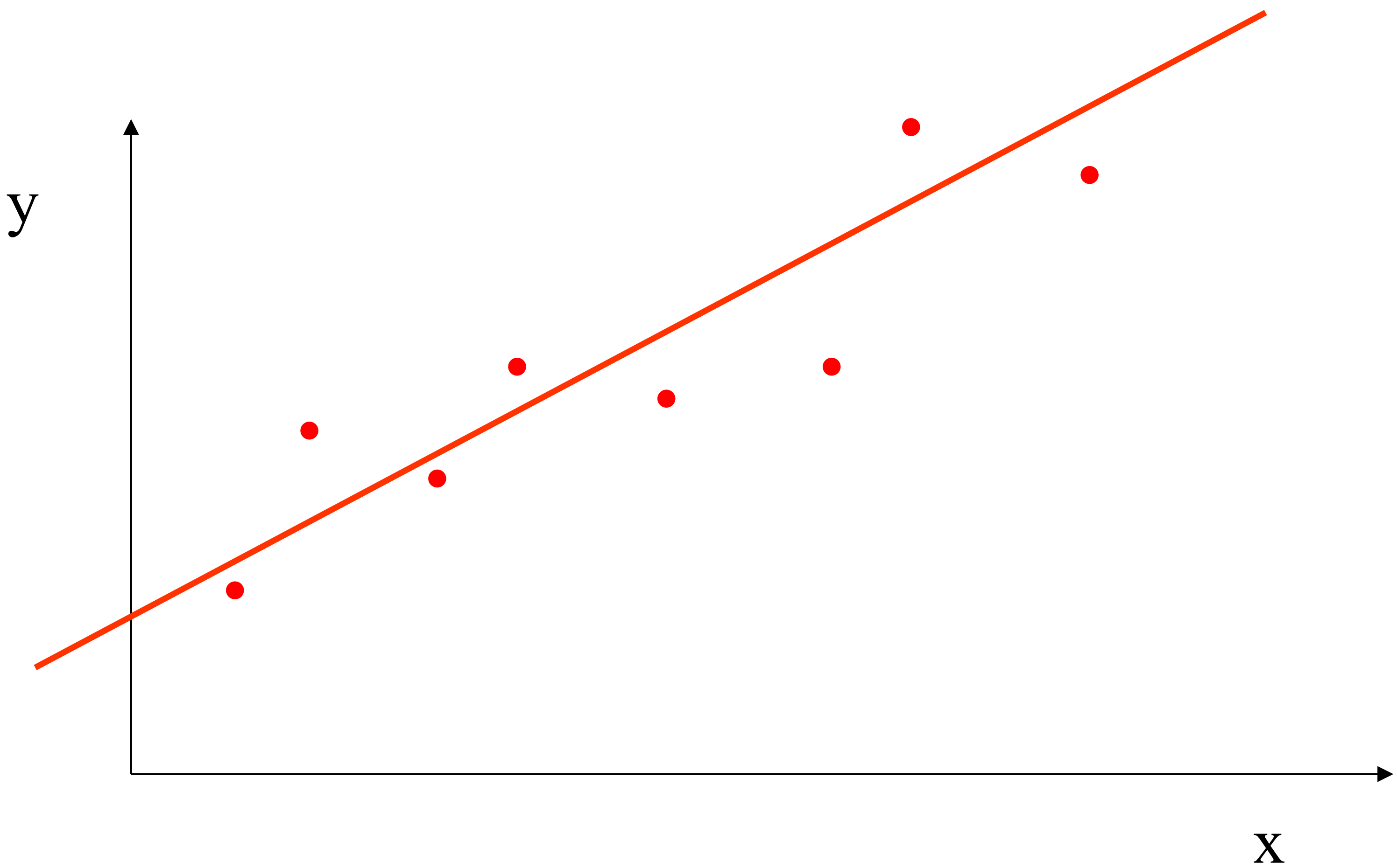
All of these explain the data in some way!

Overfitting & Complexity

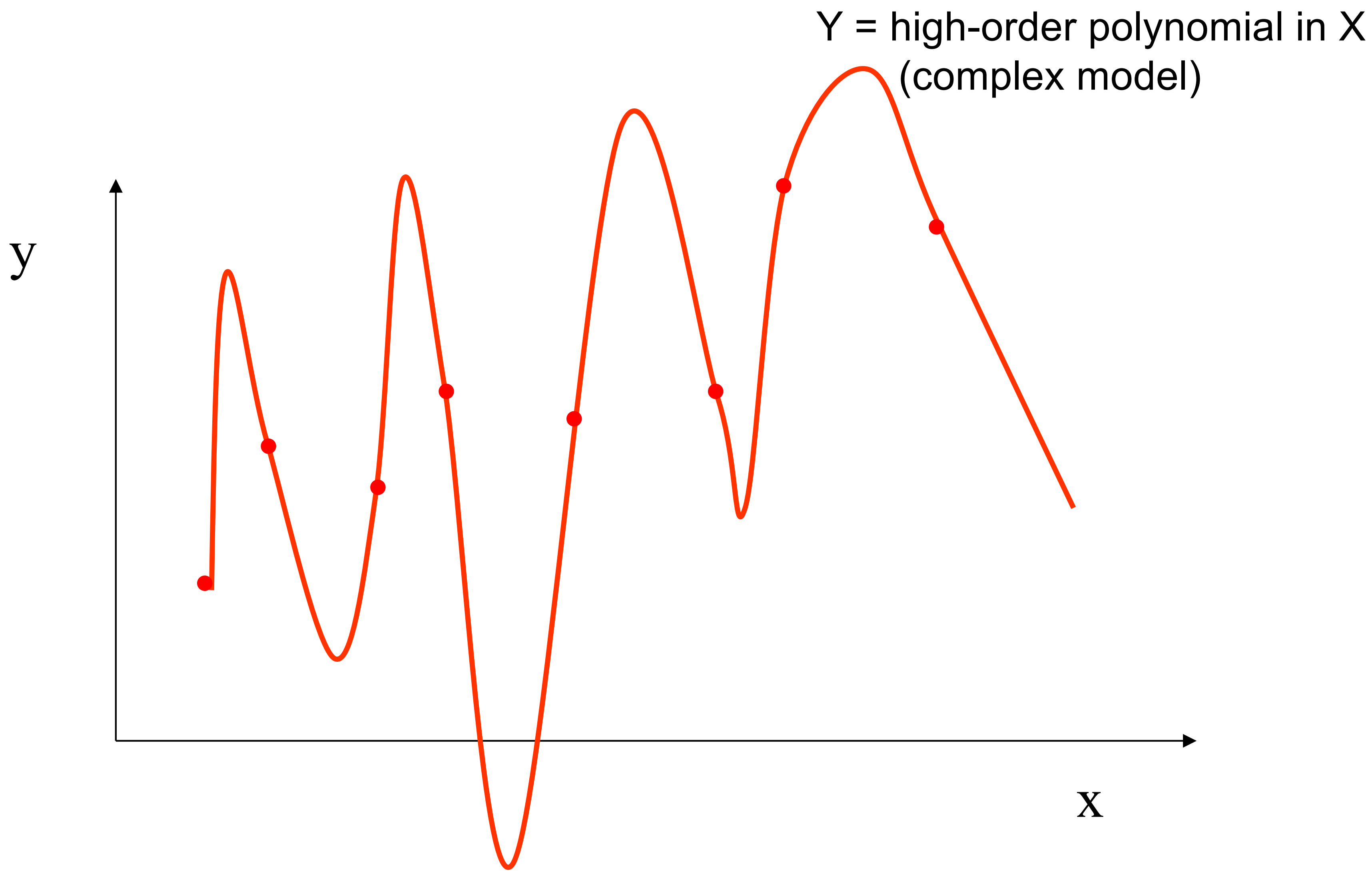


Overfitting & Complexity

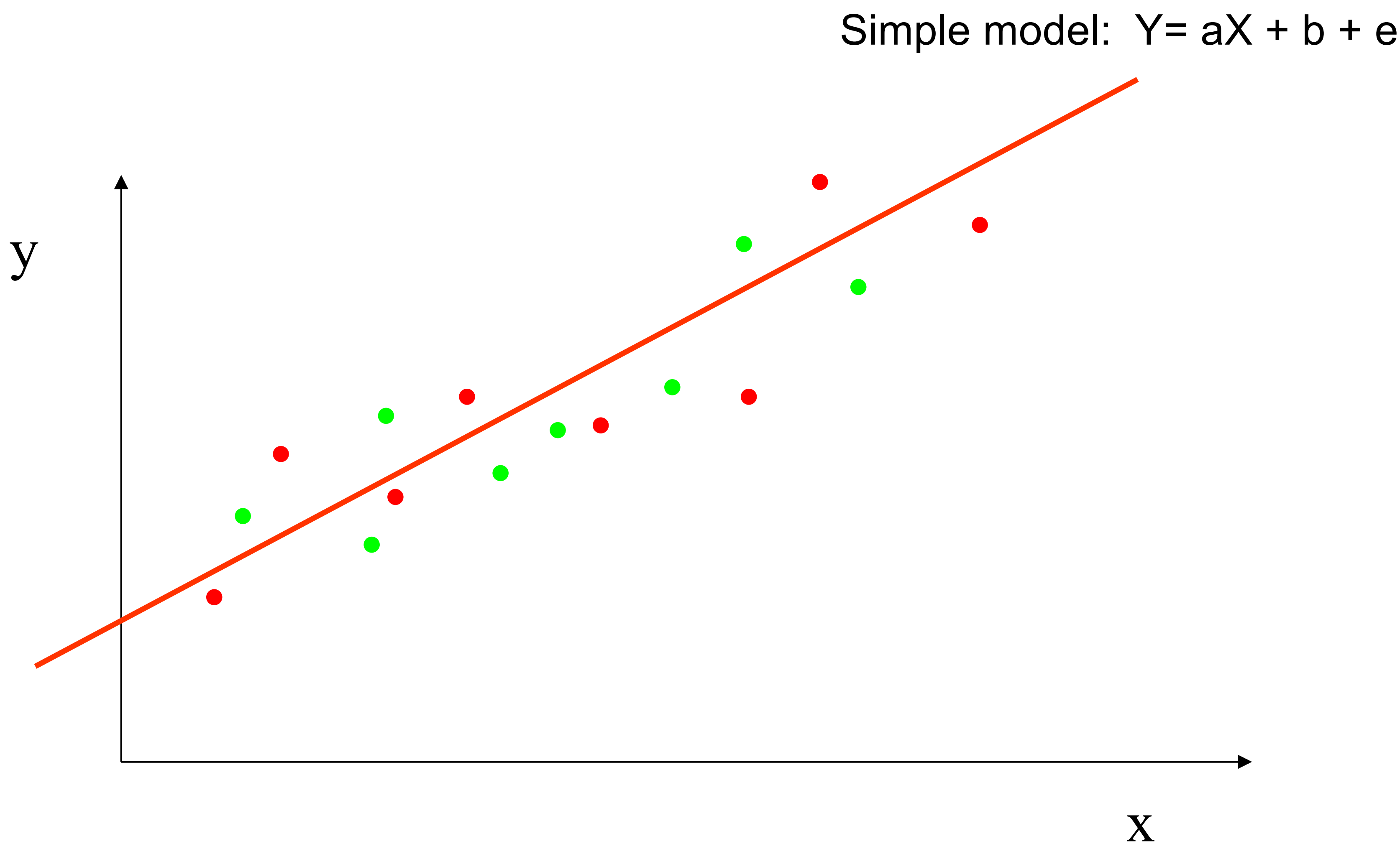
Simple model: $Y = aX + b + e$



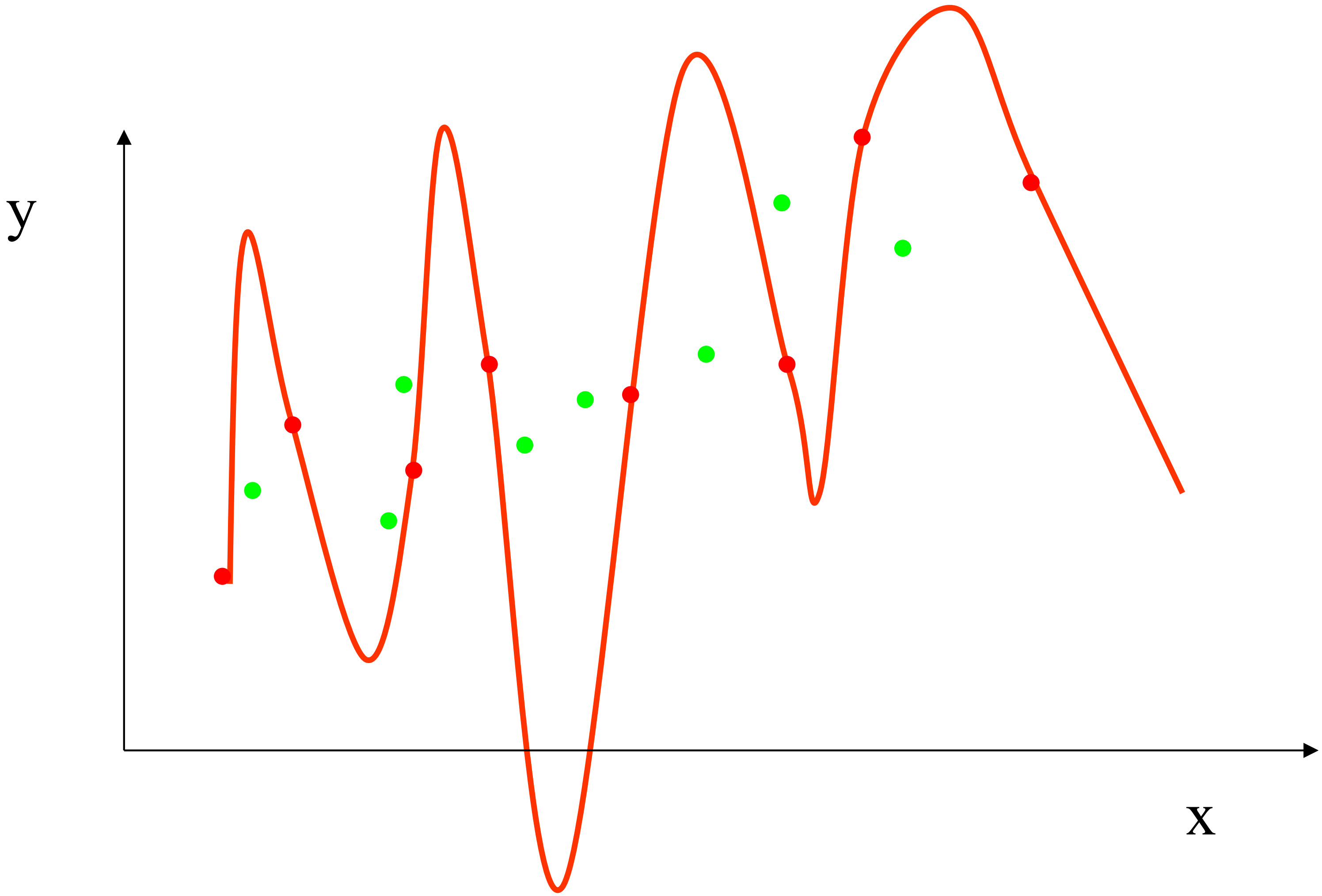
Overfitting & Complexity



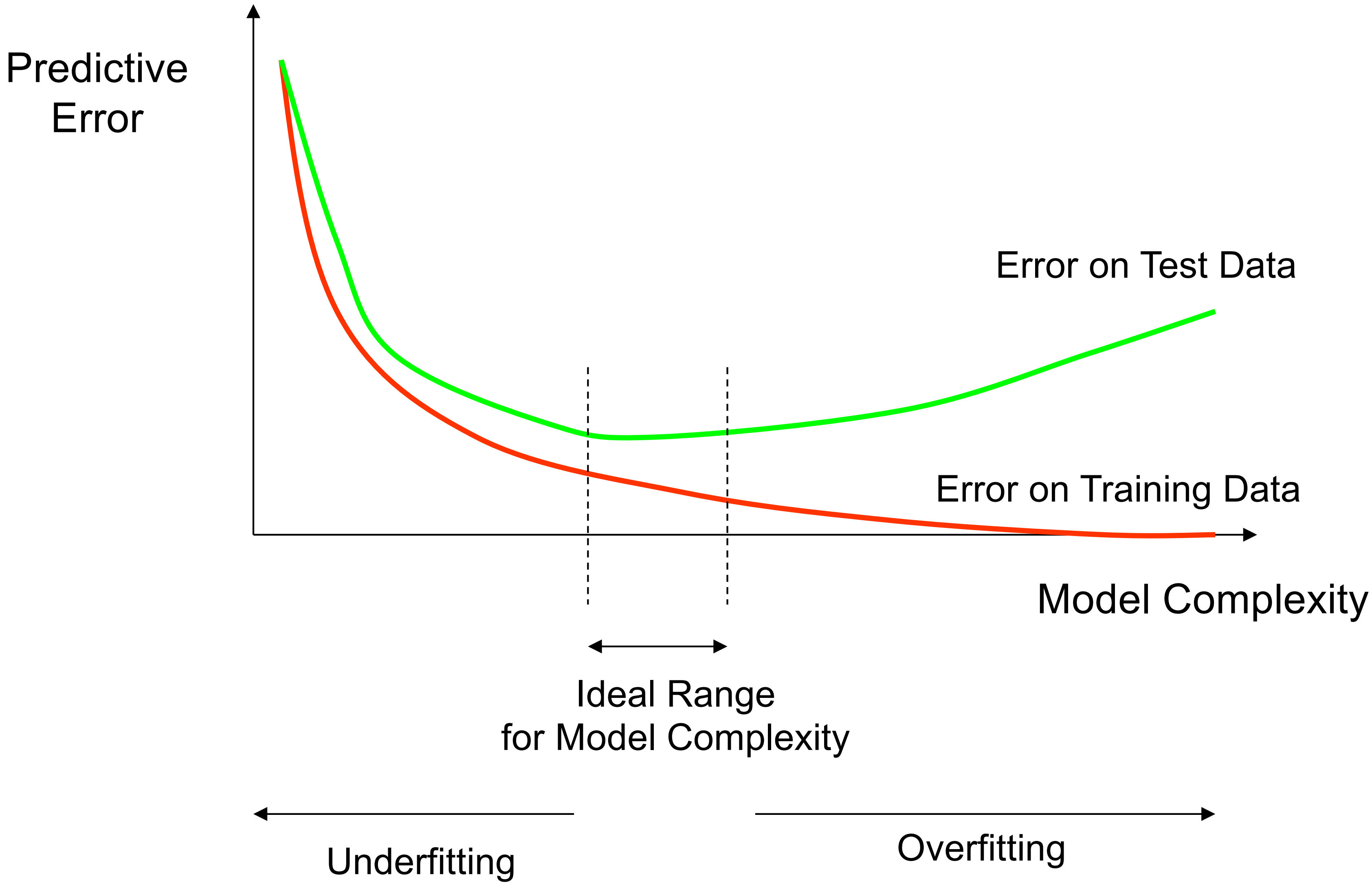
Overfitting & Complexity



Overfitting & Complexity



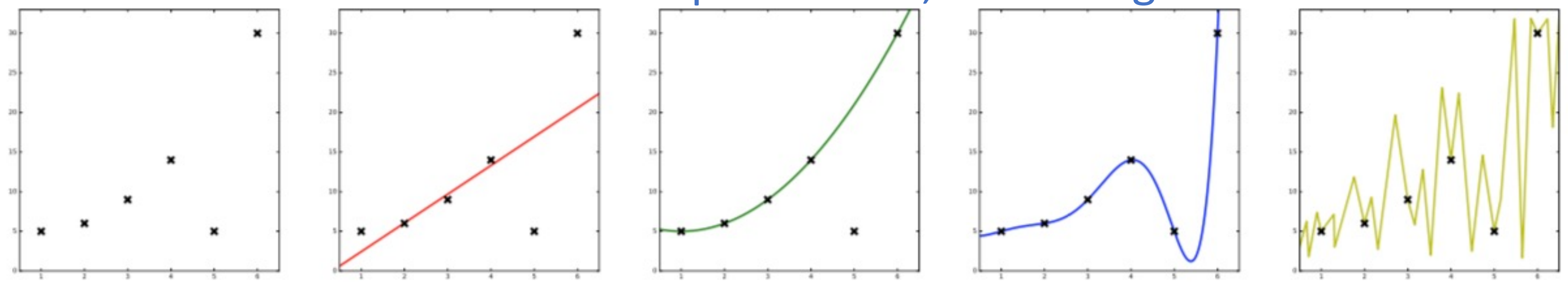
How Overfitting Affects Prediction



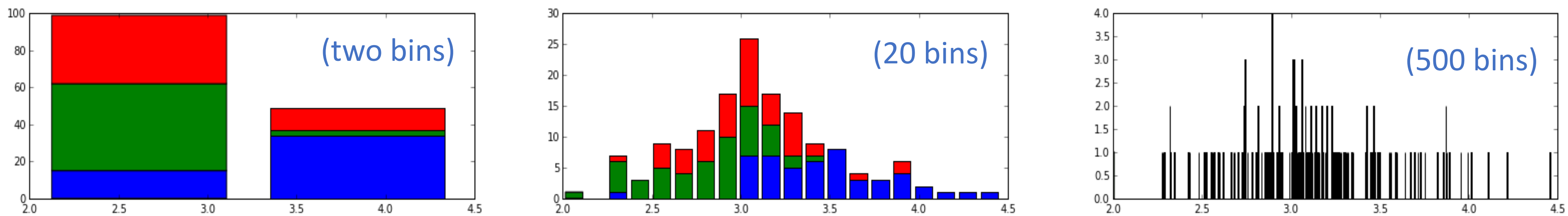
Recall: Inductive bias

- How can we transfer observations to other, unobserved values?

Data　　　　Models that “explain” data, extending to other values



- For p(x,y)? One option: discretize (histograms)



  - Binning “transfers” data density to nearby feature values
  - Too few bins = lose information; too many = noisy, no estimates at many locations

Fundamental issue of ML: How can we transfer information from “similar” examples?

Outline

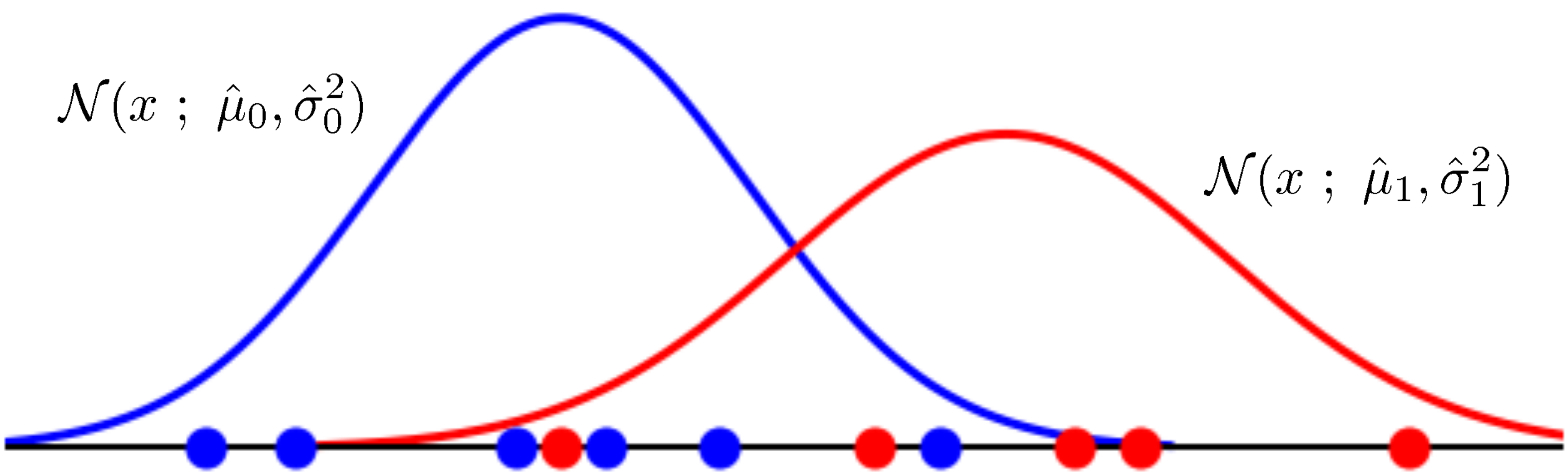
- Optimal Decisions (in theory)
- **Bayes Classifiers**
- Types of Errors
- Training & Validation Data
- K-Nearest Neighbor Models

Gaussian probability models

- Estimate parameters of a Gaussian distribution from data
  - Gaussian dist: $\mathcal{N}(x\,;\,\mu_c,\sigma_c^2) = \left(2\pi\sigma_c^2\right)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(x-\mu_c)^2/\sigma_c^2\right]$
  - Empirical (and maximum likelihood) parameter estimates:

$$\hat{p}(Y=1) = \frac{m_1}{m} \qquad \hat{\mu}_1 = \frac{1}{m_1}\sum_{i:y^{(i)}=1} x^{(i)} \qquad \hat{\sigma}_1^2 = \frac{1}{m_1}\sum_{i:y^{(i)}=1} (x^{(i)}-\hat{\mu}_1)^2$$

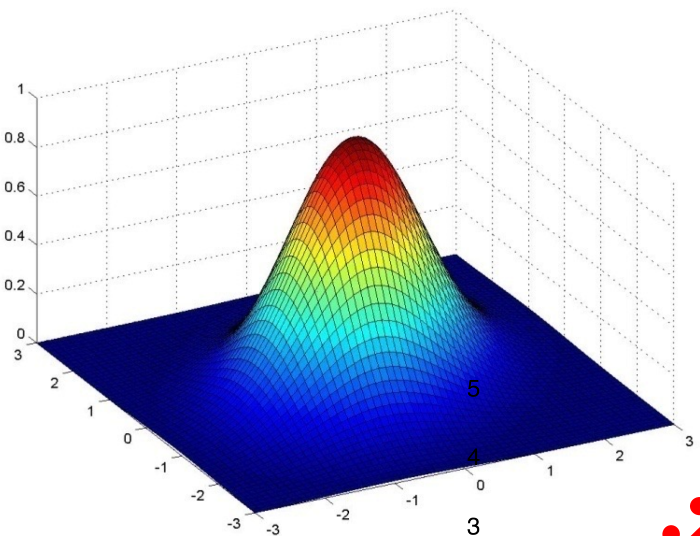
(and similarly for class 0)



Multivariate Gaussian models

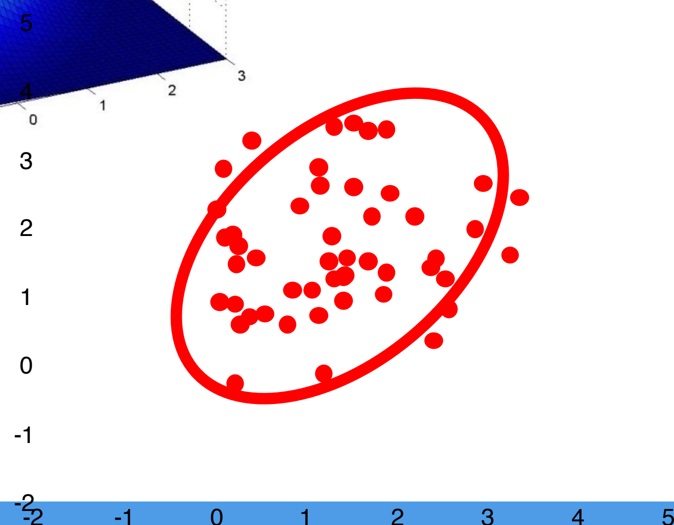
- Similar to univariate case

$$\mathcal{N}(x\,;\,\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}}|\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$



$$\mu = n \times 1 \text{ mean vector}$$

$$\Sigma = n \times n \text{ covariance matrix}$$



Maximum likelihood estimate:

$$\hat{\mu} = \frac{1}{m}\sum_j x^{(j)}$$

$$\hat{\Sigma} = \frac{1}{m}\sum_j (x^{(j)} - \hat{\mu})^T(x^{(j)} - \hat{\mu})$$

Bayes rule

- How to compute the probability of a hidden "cause" Y, after observing some evidence "effect" X:

How probable is the hidden cause?

$$p(Y|X)\ p(X) = p(X,Y) = p(X|Y)\ p(Y)$$

How often does Y cause X?

$$\Rightarrow \quad p(Y|X) = \frac{p(X|Y)\ p(Y)}{p(X)}$$

"Bayes rule"

- Example: flu
  - P(F), P(H|F)
  - P(F=1 | H=1) = ?

$$= \frac{0.50 * 0.05}{0.50 * 0.05 + 0.20 * 0.95} = 0.116$$

F	P(F)
0	0.95
1	0.05

F	H	P(H|F)
0	0	0.80
0	1	0.20
1	0	0.50
1	1	0.50

Bayes Classifiers from Data

- Estimate prior probability of each class, p(y)
  - E.g., how common is each type of Iris?

- Distribution of features given the class, p(x | y=c)
  - How likely are we to see "x" in each type of iris?

- Joint distribution $p(y|x)p(x) = p(x, y) = p(x|y)p(y)$

- Bayes Rule: $\Rightarrow \quad p(y|x) = p(x|y)p(y)/p(x)$

$$= \frac{p(x|y)p(y)}{\sum_c p(x|y=c)p(y=c)}$$

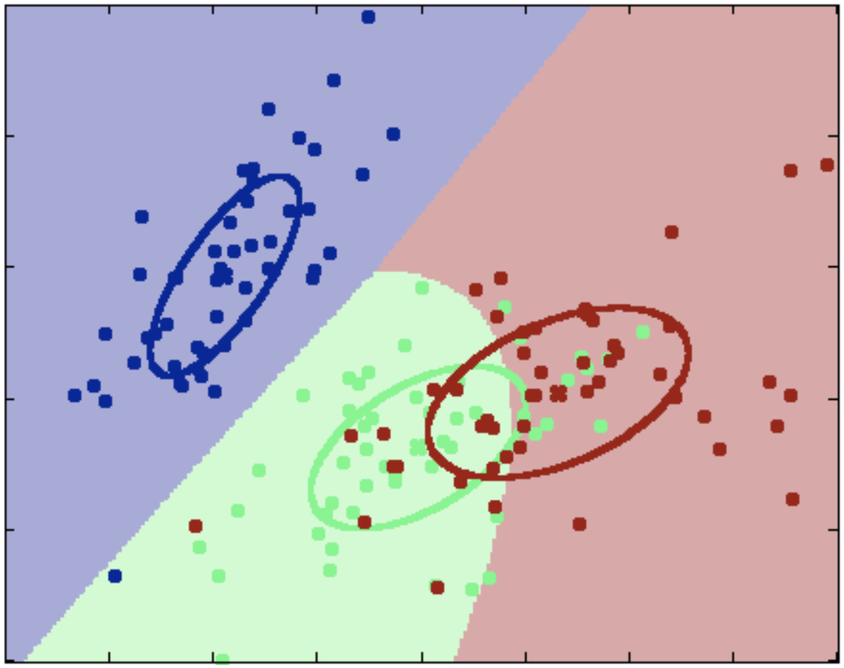
(Use the rule of total probability to calculate the denominator!)

Example: Gaussian Bayes, Iris Data

- Fit Gaussian distribution to each class {0,1,2}

$$p(y) = \mathrm{Discrete}(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3})$$

$$p(x_1, x_2 | y = 0) = \mathcal{N}(x\,;\,\mu_0, \Sigma_0)$$
$$p(x_1, x_2 | y = 1) = \mathcal{N}(x\,;\,\mu_1, \Sigma_1)$$
$$p(x_1, x_2 | y = 2) = \mathcal{N}(x\,;\,\mu_2, \Sigma_2)$$



Then, Bayes rule:

(How well does Y=blue explain x?)

(How well do Y=green or Y=red explain x?)

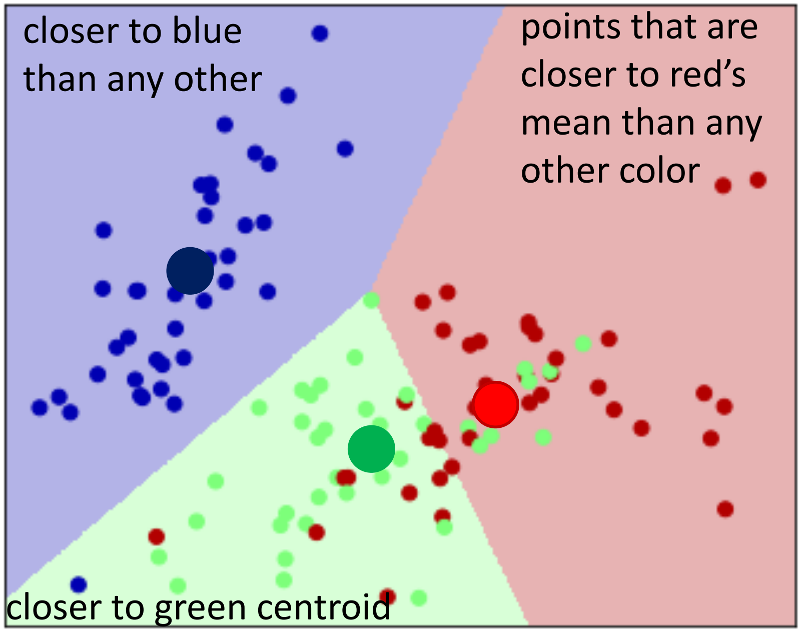
$$p(Y = b|x) = \frac{p(Y = b)p(x|Y = b)}{p(Y = b)p(x|Y = b) + p(Y = g)p(x|Y = g) + p(Y = r)p(x|Y = r)}$$

Homework: Centroid Classifier

- Simple, special case of Gaussian Bayes classifier
- Estimate just the mean (centroid) of each data class
  - Then, rule is simply:

  predict class y by:

$$\hat{y}(x) = \arg\min_c \|x - \mu_c\|^2$$



Typically, use Euclidean distance:

$$\|x - \mu\|^2 = \sum_j (x_j - \mu_j)^2$$

though other distances also possible (more later…)

What about discrete features?

- Estimate joint probability for each class
  - E.g., how many times (what fraction) did each outcome occur?

A	B	C	p(A,B,C | Y=1)
0	0	0	4/10
0	0	1	1/10
0	1	0	0/10
0	1	1	0/10
1	0	0	1/10
1	0	1	2/10
1	1	0	1/10
1	1	1	1/10

- m data << 2^n parameters?

- What about the zeros?
  - We learn that certain combinations are impossible?
  - What if we see these later in test data?

- Overfitting!

What about discrete features?

- Estimate joint probability for each class
  - E.g., how many times (what fraction) did each outcome occur?

A	B	C	p(A,B,C | Y=1)
0	0	0	4/10
0	0	1	1/10
0	1	0	0/10
0	1	1	0/10
1	0	0	1/10
1	0	1	2/10
1	1	0	1/10
1	1	1	1/10

- *m* data << 2^n parameters?
- What about the zeros?
  - We learn that certain combinations are impossible?
  - What if we see these later in test data?
- One option: regularize $\hat{p}(a,b,c) \propto (M_{abc} + \alpha)$
- Normalize to make sure values sum to one…

Naïve Bayes Classifiers

- Another option: reduce the model complexity by assuming the features are (conditionally) independent of one another
- Independence: $p(a,b) = p(a)\, p(b)$
- $p(x_1, x_2, \ldots x_N \mid y=1) = p(x_1 \mid y=1)\, p(x_2 \mid y=1) \ldots p(x_N \mid y=1)$
- Only need to estimate each individually

A	p(A|Y=1)
0	.4
1	.6

B	p(B |Y=1)
0	.7
1	.3

C	p(C |Y=1)
0	.1
1	.9

$\Rightarrow$

A	B	C	p(A,B,C | Y=1)
0	0	0	.4 * .7 * .1
0	0	1	.4 * .7 * .9
0	1	0	.4 * .3 * .1
0	1	1	…
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Example: Naïve Bayes

Observed Data:

x_1	x_2	y
1	1	0
1	0	0
1	0	1
0	0	0
0	1	1
1	1	0
0	0	1
1	0	1

$$\hat{p}(y=1) = \frac{4}{8} \qquad = (1 - \hat{p}(y=0))$$

$$\hat{p}(x_1, x_2 | y=0) = \hat{p}(x_1 | y=0)\, \hat{p}(x_2 | y=0)$$

$$\hat{p}(x_1 = 1 | y=0) = \frac{3}{4} \qquad \hat{p}(x_1 = 1 | y=1) = \frac{2}{4}$$

$$\hat{p}(x_2 = 1 | y=0) = \frac{2}{4} \qquad \hat{p}(x_2 = 1 | y=1) = \frac{1}{4}$$

Prediction given some observation x?

$$\hat{p}(y=1)\hat{p}(x=11|y=1) \quad \begin{matrix} < \\ > \end{matrix} \quad \hat{p}(y=0)\hat{p}(x=11|y=0)$$

$$\frac{4}{8} \times \frac{2}{4} \times \frac{1}{4} \qquad\qquad \frac{4}{8} \times \frac{3}{4} \times \frac{2}{4}$$

Decide class 0

Example: Naïve Bayes

Observed Data:

x_1	x_2	y
1	1	0
1	0	0
1	0	1
0	0	0
0	1	1
1	1	0
0	0	1
1	0	1

$$\hat{p}(y=1) = \frac{4}{8} \qquad = (1 - \hat{p}(y=0))$$

$$\hat{p}(x_1, x_2 | y=0) = \hat{p}(x_1 | y=0)\, \hat{p}(x_2 | y=0)$$

$$\hat{p}(x_1 = 1 | y=0) = \frac{3}{4} \qquad \hat{p}(x_1 = 1 | y=1) = \frac{2}{4}$$

$$\hat{p}(x_2 = 1 | y=0) = \frac{2}{4} \qquad \hat{p}(x_2 = 1 | y=1) = \frac{1}{4}$$

$$\hat{p}(y=1 | x_1 = 1, x_2 = 1) = \frac{\frac{4}{8} \times \frac{2}{4} \times \frac{1}{4}}{\frac{3}{4} \times \frac{2}{4} \times \frac{4}{8} + \frac{2}{4} \times \frac{1}{4} \times \frac{4}{8}}$$

$$= \frac{1}{4}$$

Example: Joint Bayes

Observed Data:

x_1	x_2	y
1	1	0
1	0	0
1	0	1
0	0	0
0	1	1
1	1	0
0	0	1
1	0	1

$$\hat{p}(y=1) = \frac{4}{8} \quad = (1 - \hat{p}(y=0))$$

$$\hat{p}(x_1, x_2 | y=0) =$$

x_1	x_2	p(x | y=0)
0	0	1/4
0	1	0/4
1	0	1/4
1	1	2/4

$$\hat{p}(x_1, x_2 | y=1) =$$

x_1	x_2	p(x | y=1)
0	0	1/4
0	1	1/4
1	0	2/4
1	1	0/4

$$\hat{p}(y=1 | x_1=1, x_2=1) = \frac{\frac{4}{8} \times 0}{\frac{2}{4} \times \frac{4}{8} + 0 \times \frac{4}{8}}$$

$$= 0$$

Naïve Bayes Models

- Variable y to predict, e.g. "auto accident in next year?"
- *Many* co-observed variables $x=[x_1 \ldots x_n]$
  - Age, income, education, zip code, …
- Learn $p(y \mid x_1 \ldots x_n)$, to predict y?
  - Arbitrary distribution: $O(d^n)$ values!
- Naïve Bayes:

Now only 2*n*d parameters!

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

Bayes Rule

$$p(x|y) = \prod_j p(x_j|y)$$

"Naïve" : conditional independence

- Note: may not be a good model of the data
  - Doesn't capture correlations in features
  - Can't capture some dependencies
- But in practice it often does quite well!

Outline

- Optimal Decisions (in theory)
- Bayes Classifiers
- **Types of Errors**
- Training & Validation Data
- K-Nearest Neighbor Models

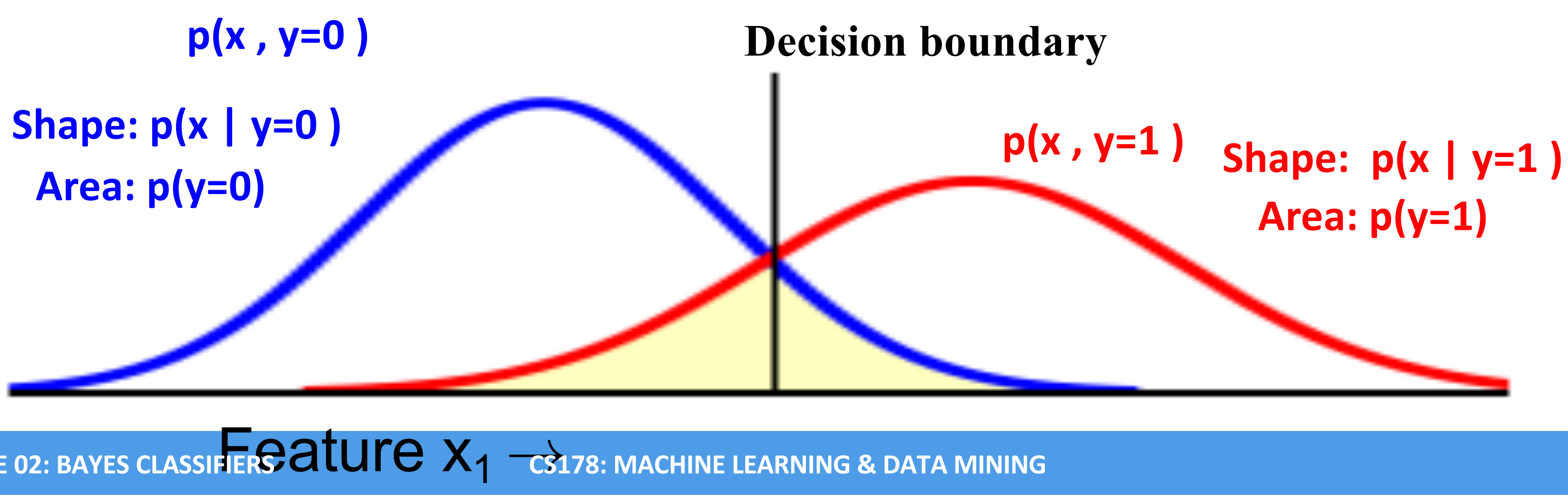
Bayes Classifiers

- Bayes classification decision rule compares probabilities:

$$p(y=0|x) \underset{>}{<} p(y=1|x)$$

$$= \quad p(y=0,x) \underset{>}{<} p(y=1,x)$$

- Can visualize this nicely if x is a scalar:

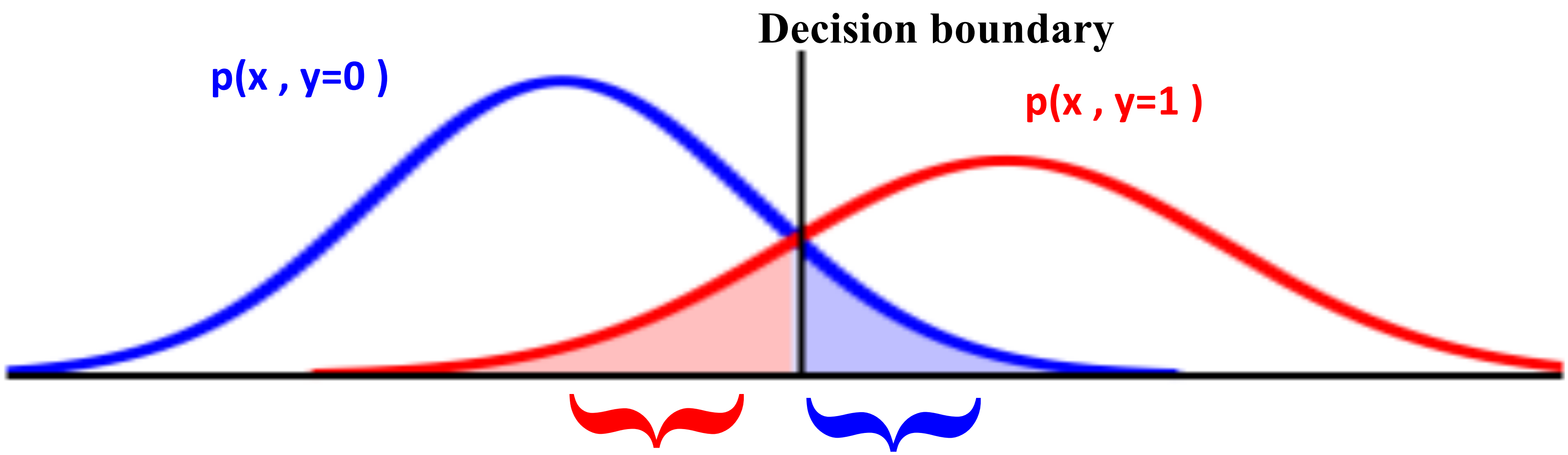


Bayes Classifiers

- Not all errors are created equally…
- Risk associated with each outcome?

Add multiplier alpha:

$$\alpha \; p(y=0,x) \; \begin{matrix} < \\ > \end{matrix} \; p(y=1,x)$$



Type 1 errors: false positives

Type 2 errors: false negatives

False positive rate: (# y=0, ŷ=1) / (#y=0)

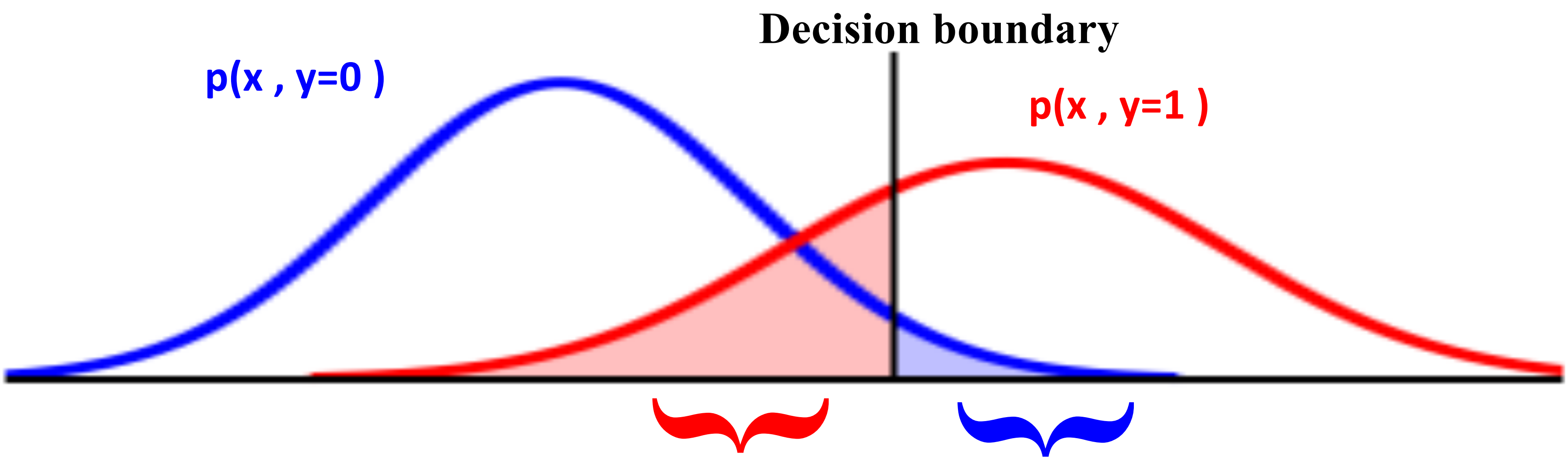
False negative rate: (# y=1, ŷ=0) / (#y=1)

Bayes Classifiers

- Increase alpha: prefer class 0
- Spam detection

Add multiplier alpha:

$$\alpha \ p(y=0,x) \ \begin{matrix} < \\ > \end{matrix} \ p(y=1,x)$$



Type 1 errors: false positives

Type 2 errors: false negatives

False positive rate: (# y=0, ŷ=1) / (#y=0)

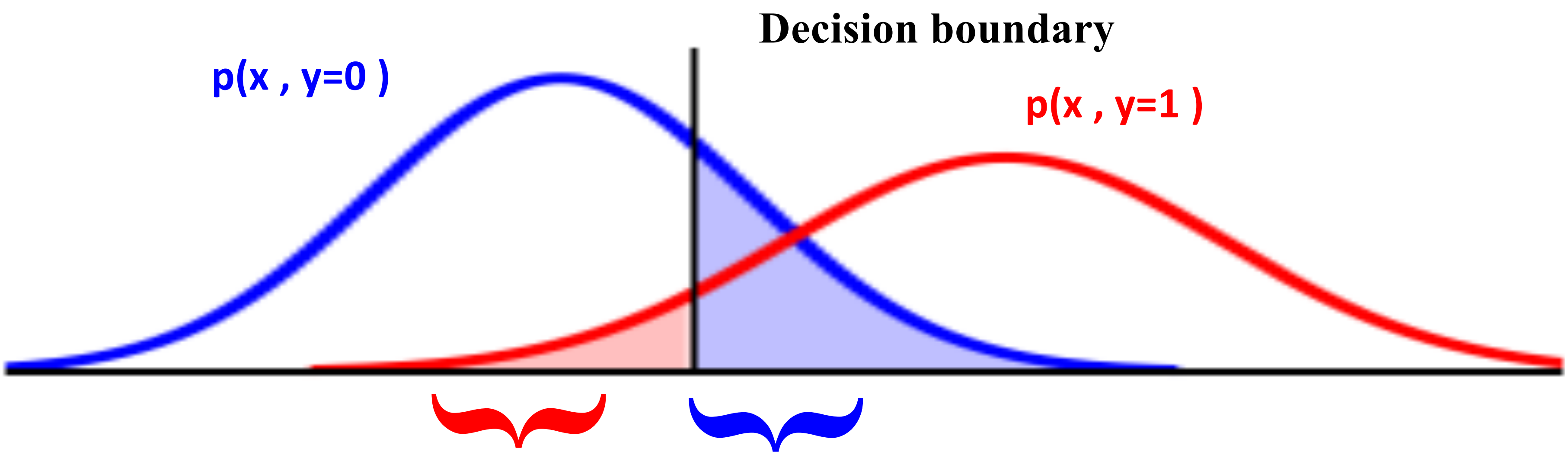
False negative rate: (# y=1, ŷ=0) / (#y=1)

Bayes Classifiers

- Decrease alpha: prefer class 1
- Cancer detection

Add multiplier alpha:

$$\alpha \ p(y=0,x) \ \underset{>}{<} \ p(y=1,x)$$



Type 1 errors: false positives

Type 2 errors: false negatives

False positive rate: (# y=0, ŷ=1) / (#y=0)

False negative rate: (# y=1, ŷ=0) / (#y=1)

Measuring Errors

- Confusion matrix
- Can extend to more classes

	Predict 0	Predict 1
Y=0	380	5
Y=1	338	3

- True positive rate: #(y=1 , ŷ=1) / #(y=1) -- "sensitivity"
- False negative rate: #(y=1 , ŷ=0) / #(y=1)
- False positive rate: #(y=0 , ŷ=1) / #(y=0)
- True negative rate: #(y=0 , ŷ=0) / #(y=0) -- "specificity"

Likelihood Ratio Tests

- Connection to classical, statistical decision theory:

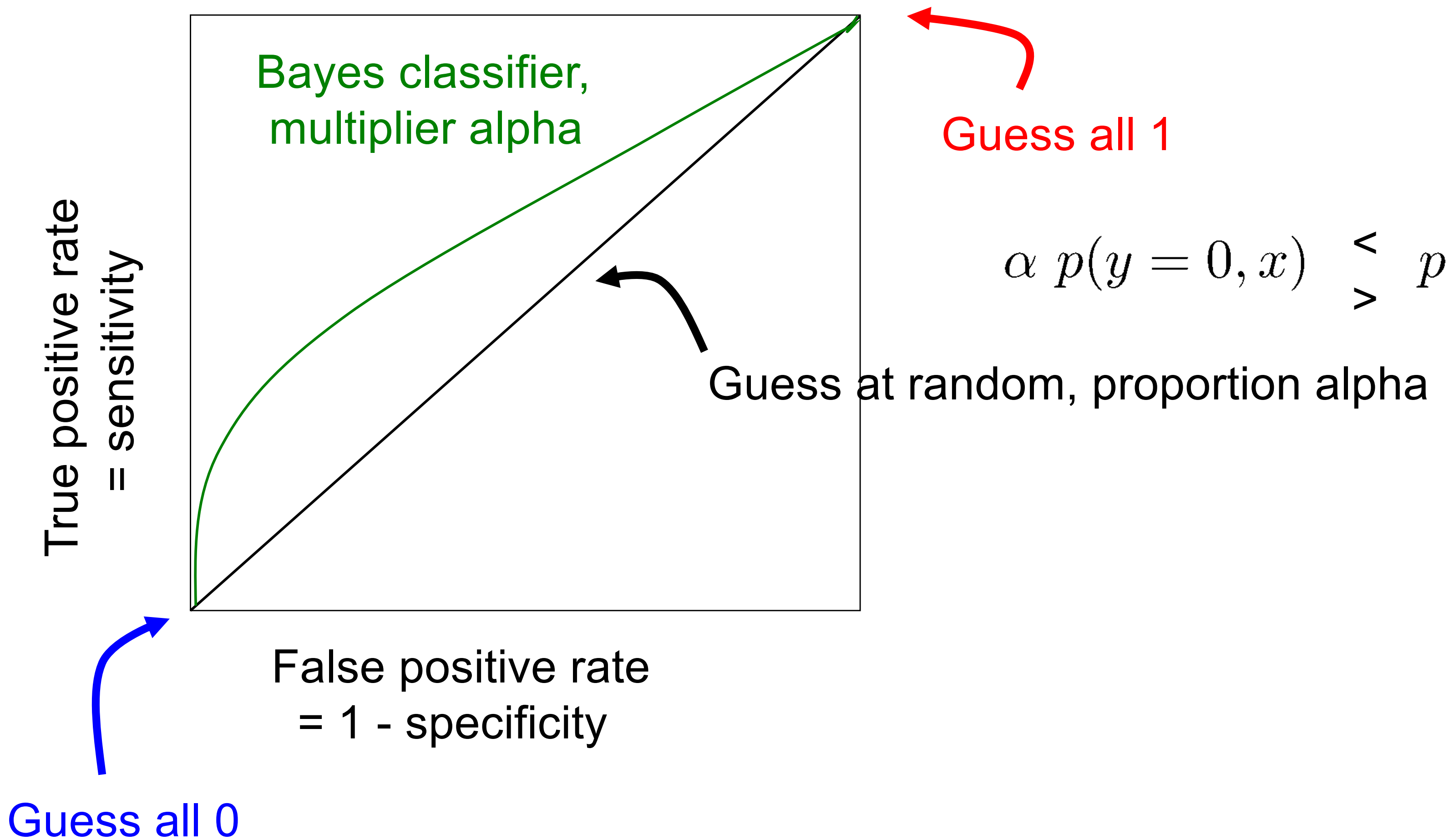
$$p(y=0,x) \ \underset{>}{<} \ p(y=1,x) \quad = \quad \log \frac{p(y=0)}{p(y=1)} \ \underset{>}{<} \ \log \frac{p(x|y=1)}{p(x|y=0)}$$

"log likelihood ratio"

- Likelihood ratio: relative support for observation "x" under "alternative hypothesis" y=1, compared to "null hypothesis" y=0
- Can vary the decision threshold: $\gamma \ \underset{>}{<} \ \log \frac{p(x|y=1)}{p(x|y=0)}$
- Classical testing:
  - Choose gamma so that FPR is fixed ("p-value")
  - Given that y=0 is true, what's the probability we decide y=1?

ROC Curves

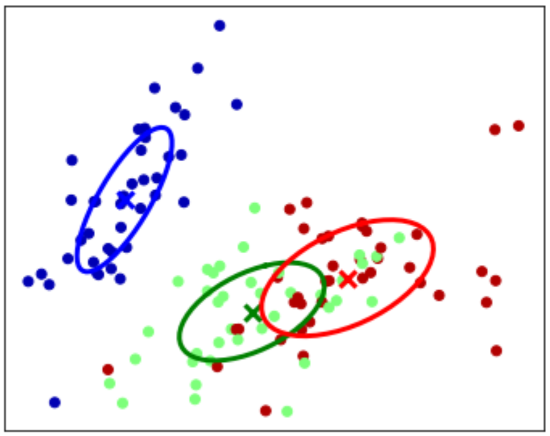
- Characterize performance as we vary the decision threshold?



$$\alpha\ p(y=0,x) \underset{>}{<} p(y=1,x)$$

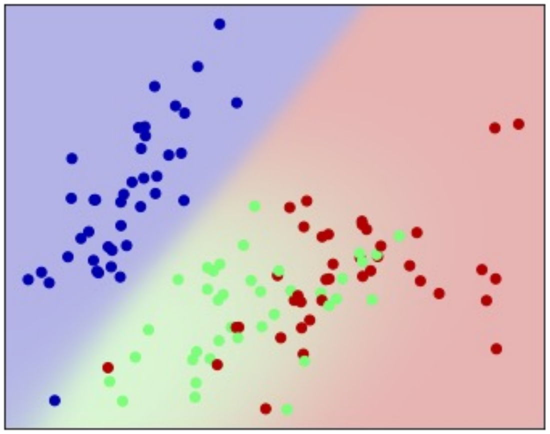
Types of Supervised Learning

Probabilistic
Generative Learning



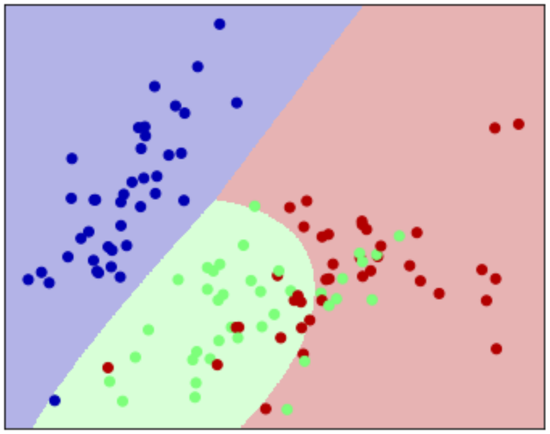
Full "generative" model
Also explain features,
e.g., p(y,x)

Probabilistic
Discriminative Learning



"Soft" predictions
Probability / confidence,
e.g., p(y|x)

Discriminative Learning



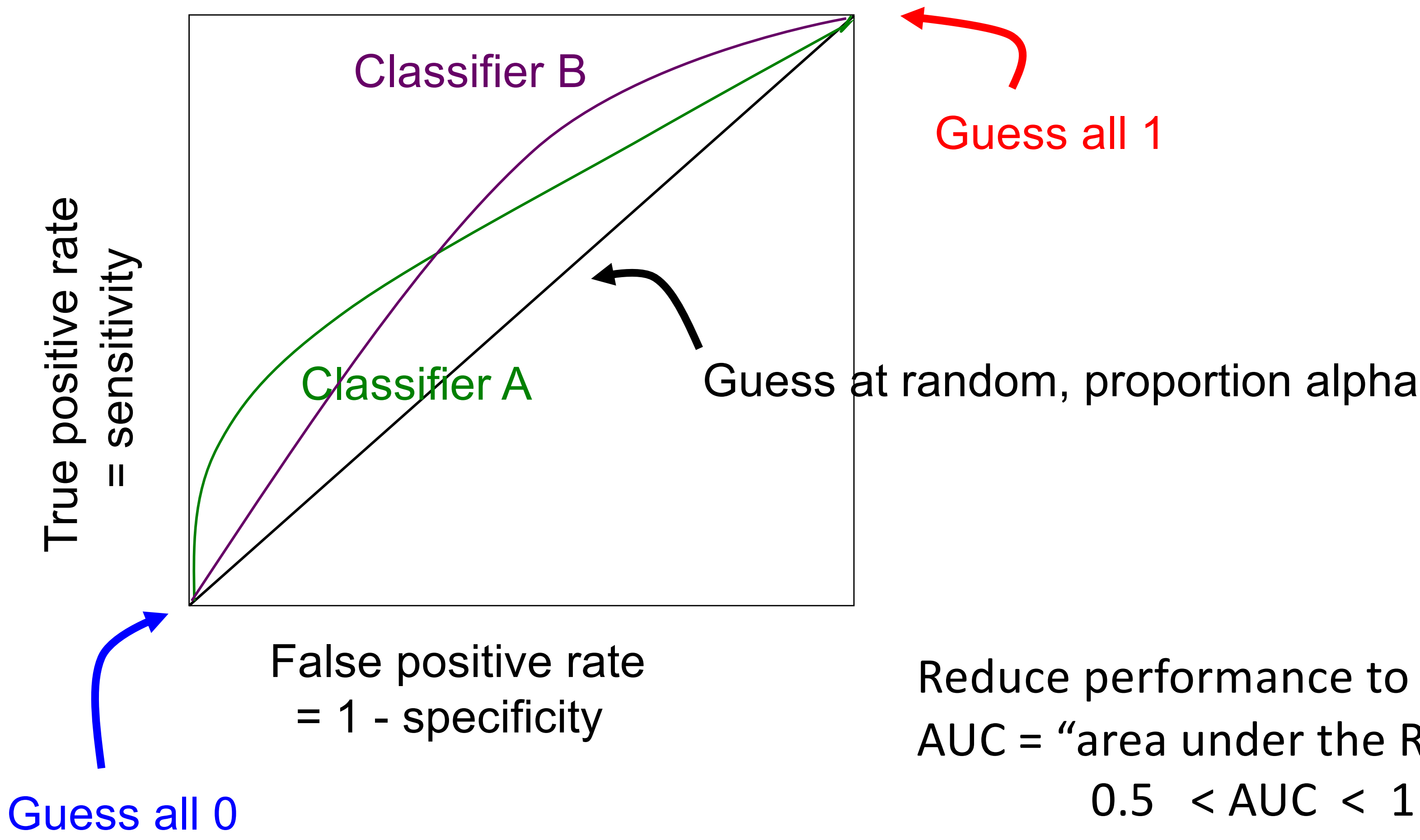
"Hard" (discrete) predictions
Minimize loss, e.g., error rate

Confidence predictions allow us to change our desired loss "after" training:

- Care more about one type of error than another?
- Expect more of one class than the other?
- (Easier to) combine different predictions? (see: ensembles)

ROC Curves

- Characterize performance as we vary our confidence threshold?



Reduce performance to one number?

AUC = “area under the ROC curve”

0.5 < AUC < 1

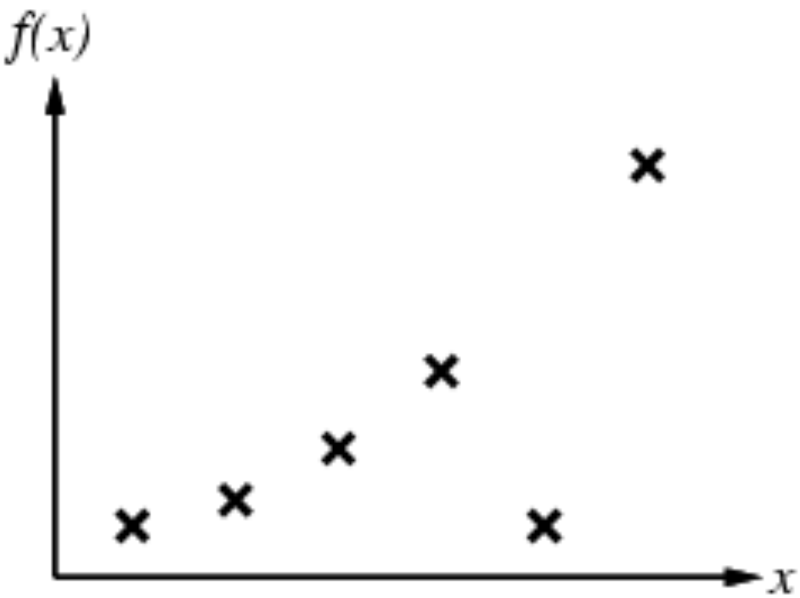
Questions?

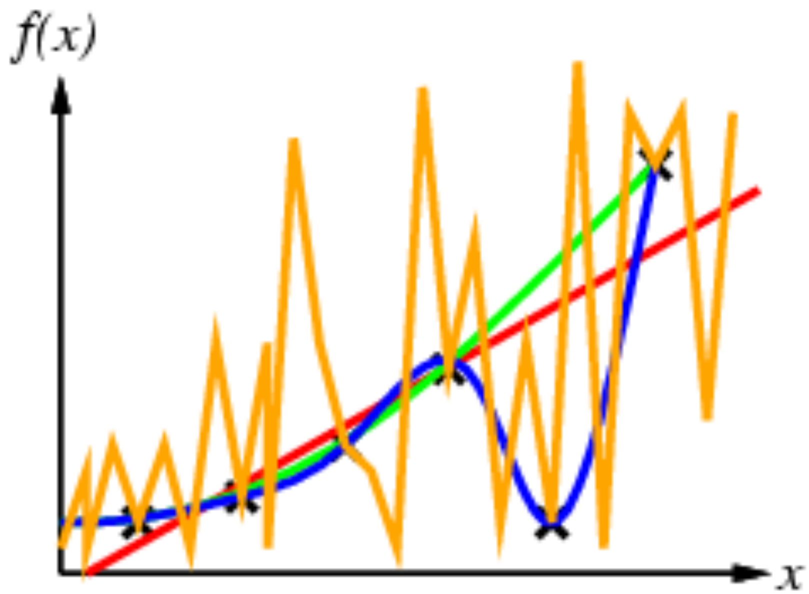
Outline

- Optimal Decisions (in theory)
- Bayes Classifiers
- Types of Errors
- **Training & Validation Data**
- K-Nearest Neighbor Models

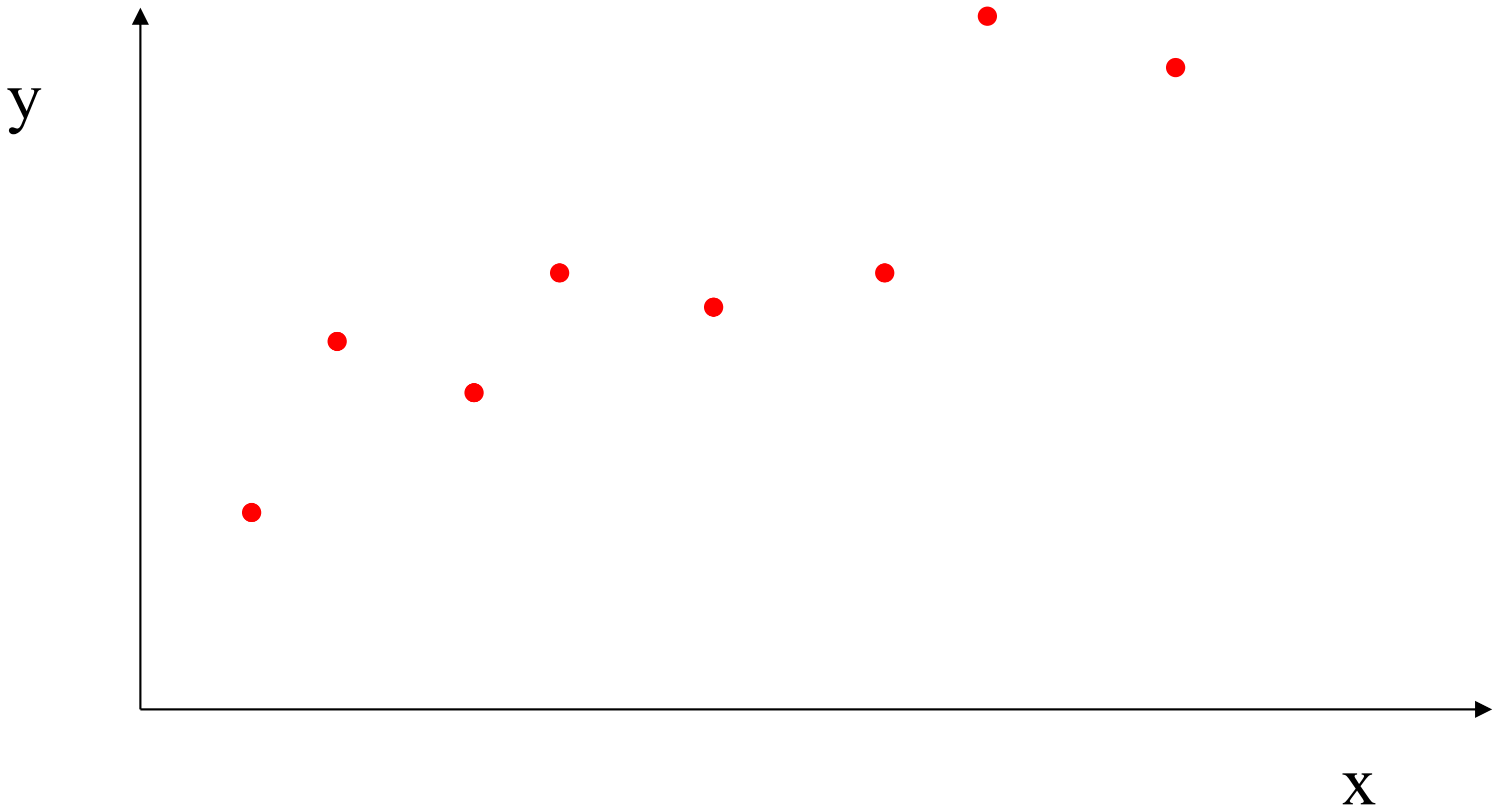
Inductive bias

- “Extend” observed data to unobserved examples
  - “Interpolate” / “extrapolate”
- What kinds of functions to expect? Prefer these (“bias”)
  - Usually, let data pull us away from assumptions only with evidence!

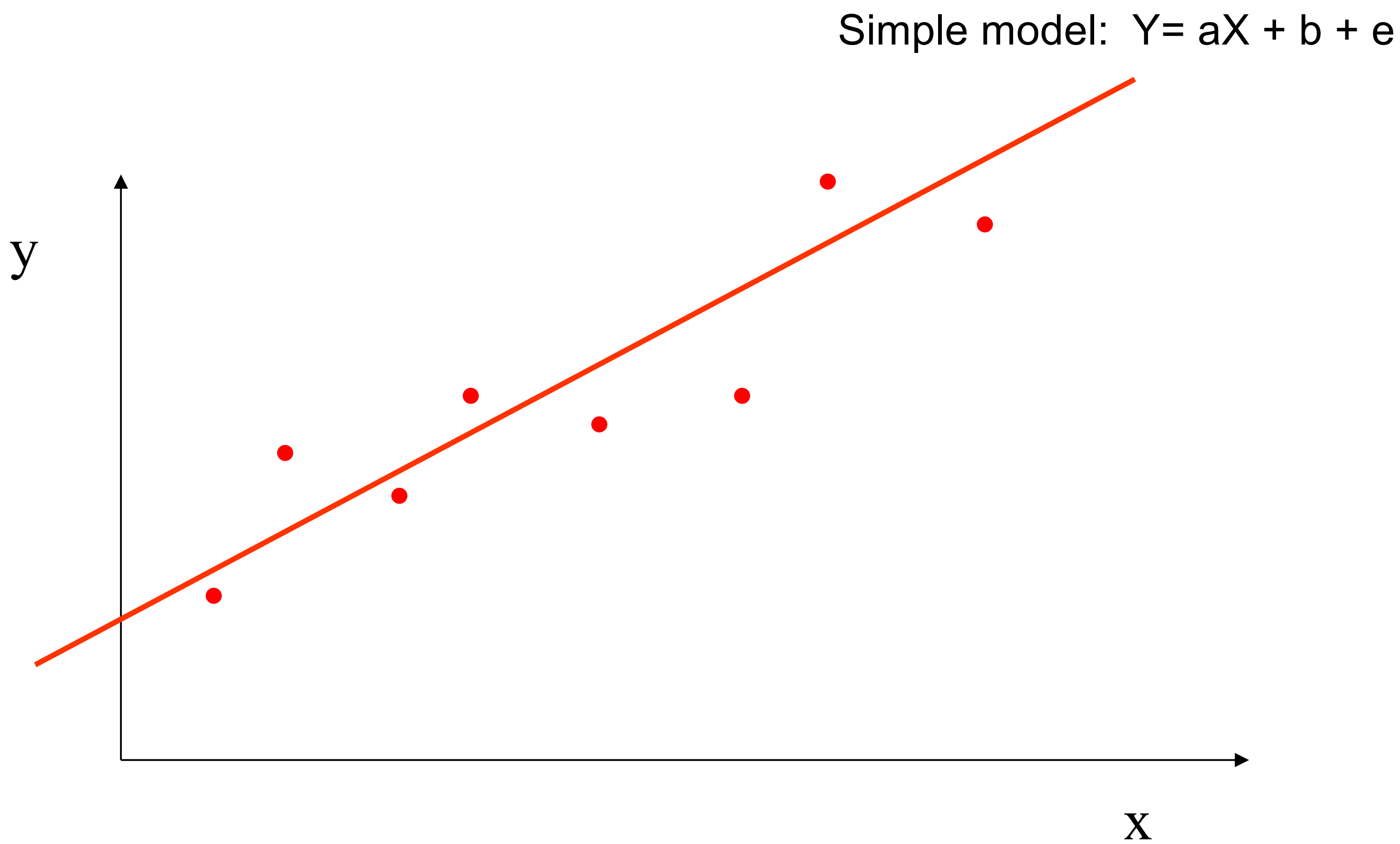




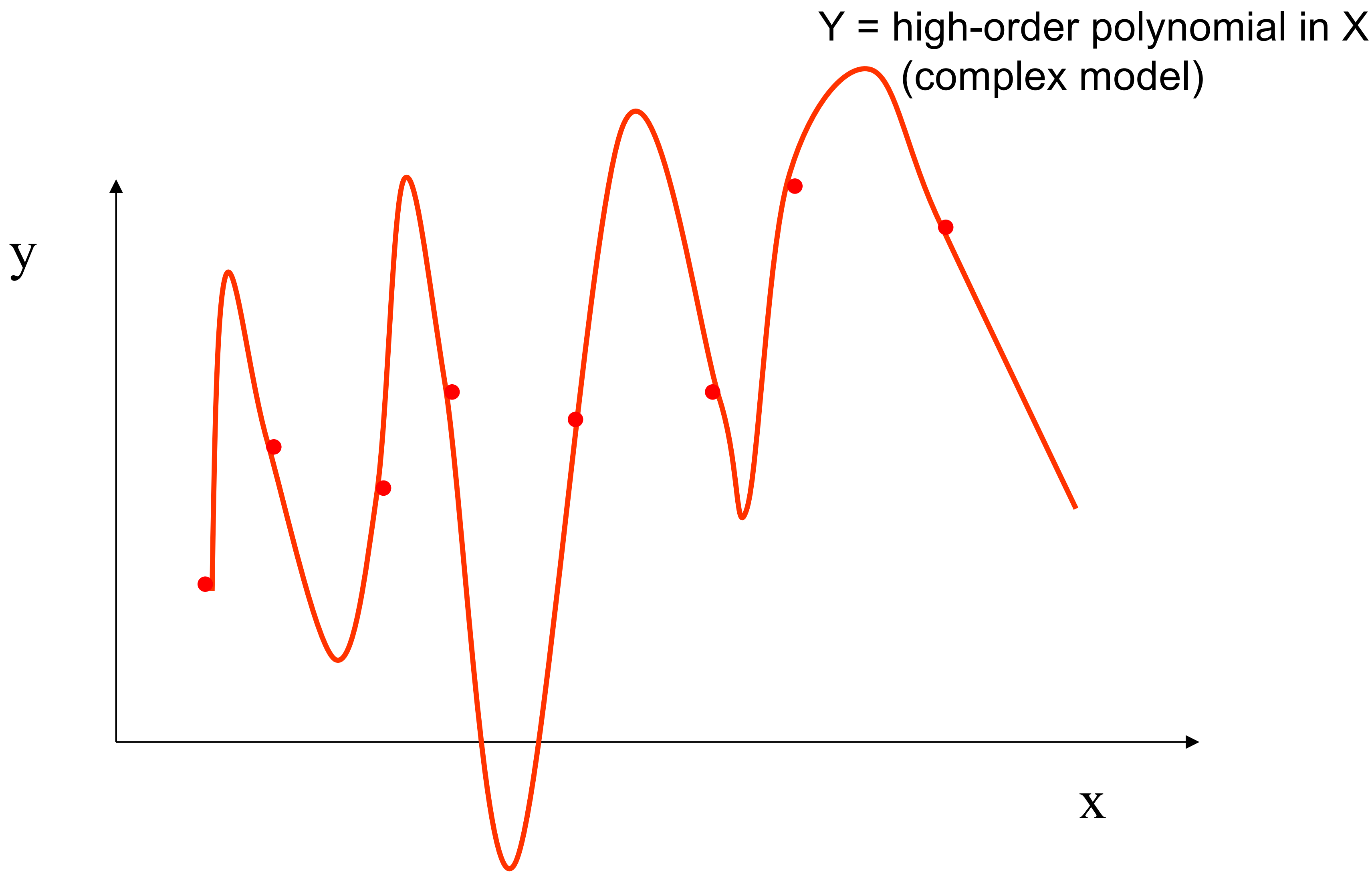
Overfitting and complexity



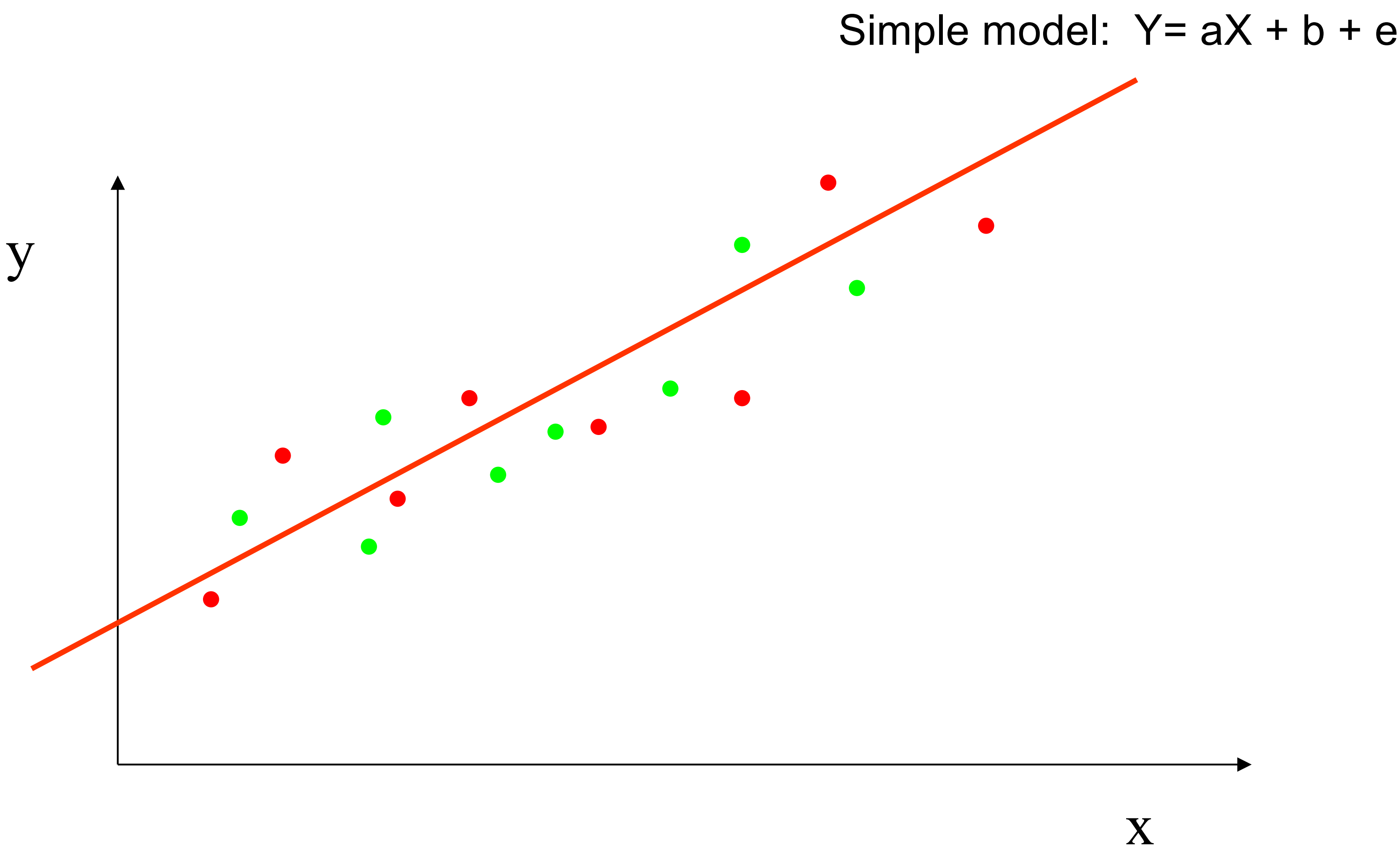
Overfitting and complexity



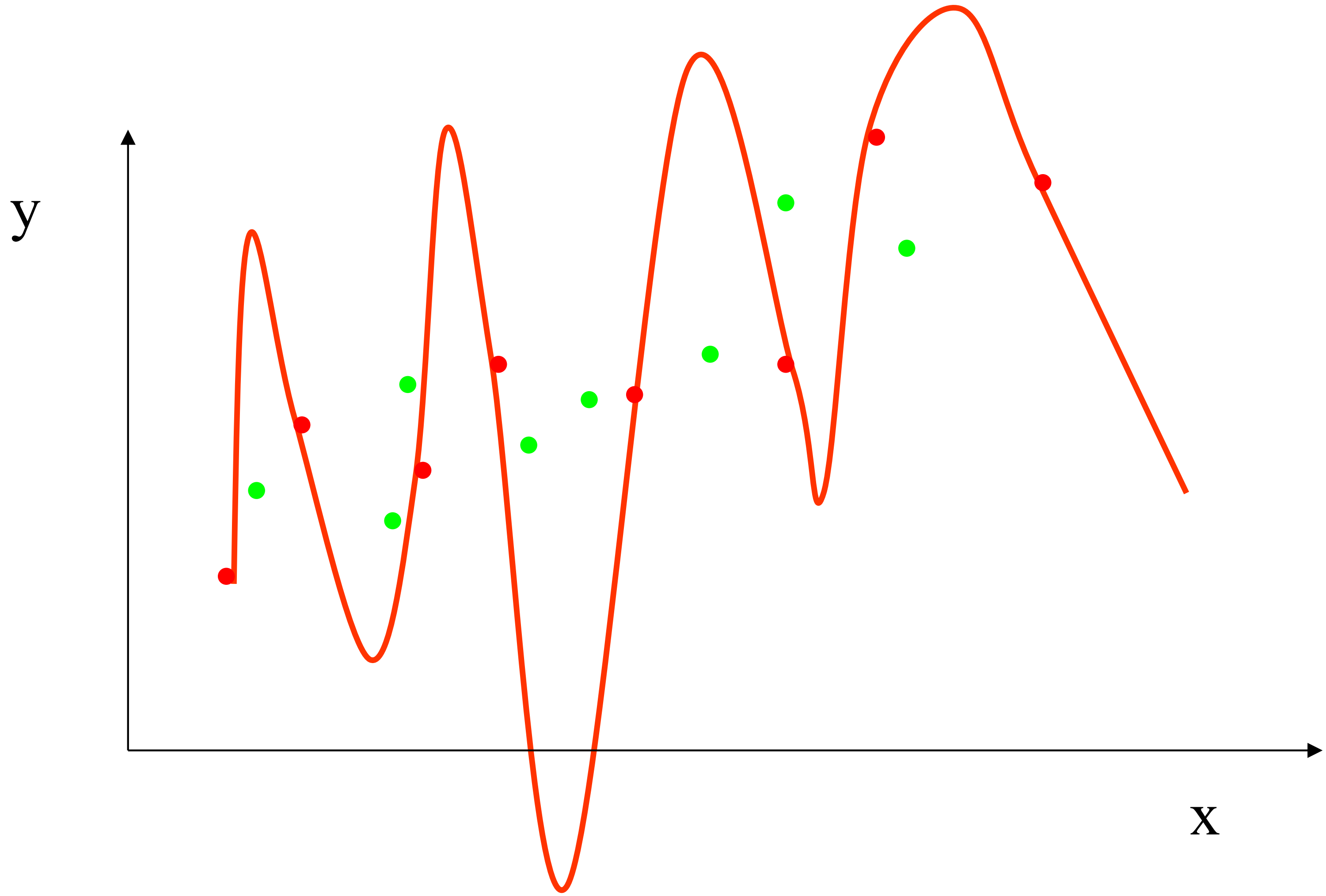
Overfitting and complexity



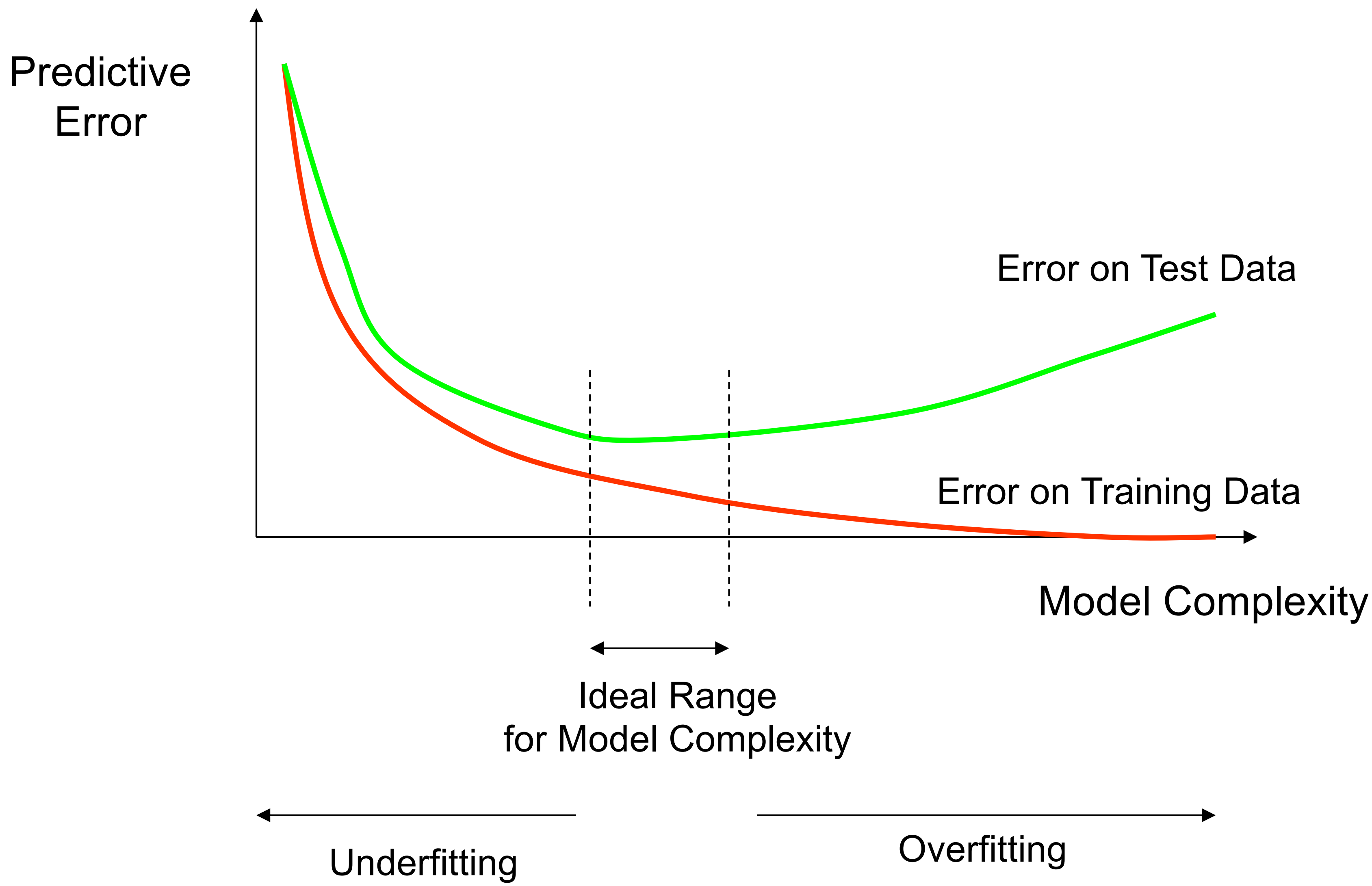
Overfitting and complexity



Overfitting and complexity



How Overfitting affects Prediction



Training and Test Data

Data

- Several candidate learning algorithms or models, each of which can be fit to data and used for prediction
- How can we decide which is best?

Approach 1: Split into train and test data

Training Data	Test Data

- Learn parameters of each model from training data
- Evaluate all models on test data, and pick best performer

Problem:

- Over-estimates test performance (“lucky” model)
- Learning algorithms should *never* have access to test data

Training, Validation, and Test Data

Data

- Several candidate learning algorithms or models, each of which can be fit to data and used for prediction
- How can we decide which is best?

Approach 2: Reserve some data for validation

Training Data | Validation | Test Data

- Learn parameters of each model from training data
- Evaluate models on validation data, pick best performer
- Reserve test data to benchmark chosen model

Problem:

- Wasteful of training data (learning can’t use validation)
- May bias selection towards overly simple models

Competitions

- Training data
  - Used to build your model(s)
- Validation data
  - Used to assess, select among, or combine models
  - Personal validation; leaderboard; …
- Test data
  - Used to estimate “real world” performance

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UT
1	-	BrickMover *	1.21251	40	Sat, 31 Aug 2013 23:
2	new	vsu *	1.21552	13	Sat, 31 Aug 2013 20:
3	↑2	Merlion	1.22724	29	Sat, 31 Aug 2013 23:
4	↓2	Sergey	1.22856	15	Sat, 31 Aug 2013 23:
5	new	liuyongqi	1.22980	13	Sat, 31 Aug 2013 13:

Outline

Optimal Decisions (in theory)

Bayes Classifiers

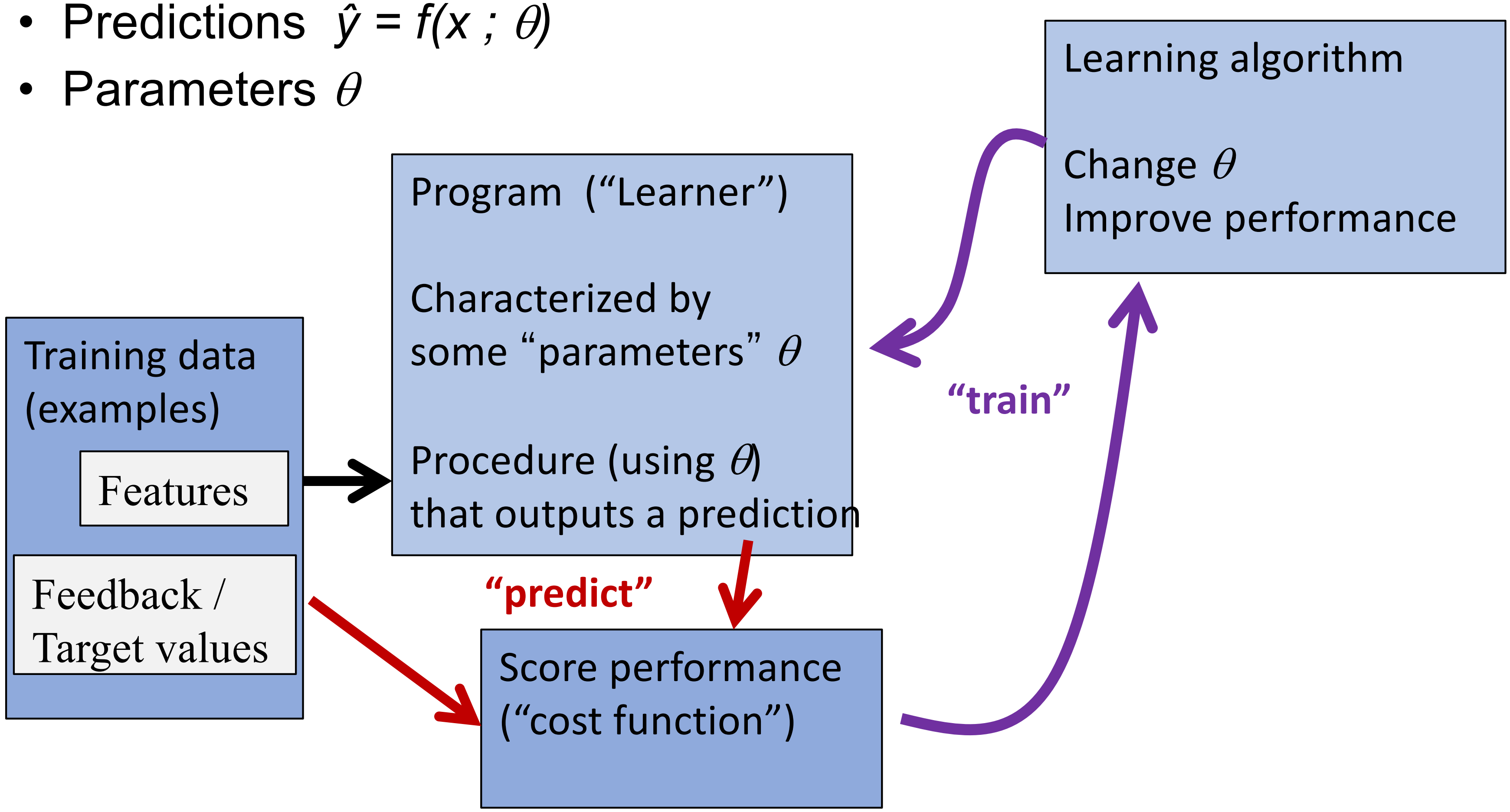
Types of Errors

Training & Validation Data

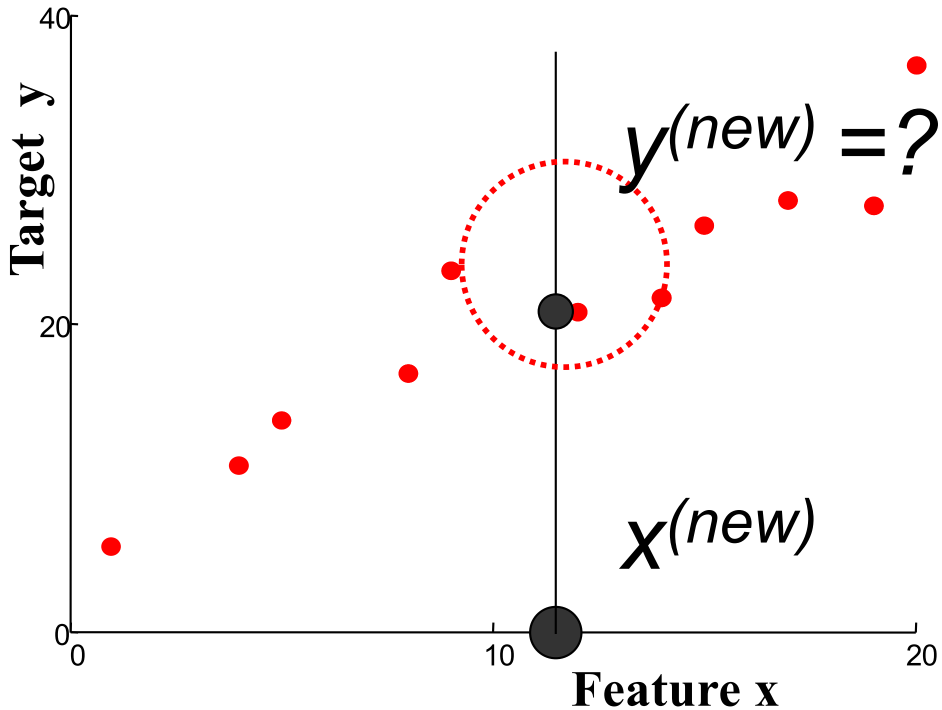
K-Nearest Neighbor Models

Supervised learning

- Notation
  - Features x
  - Targets y
  - Predictions $\hat{y} = f(x\,;\,\theta)$
  - Parameters θ



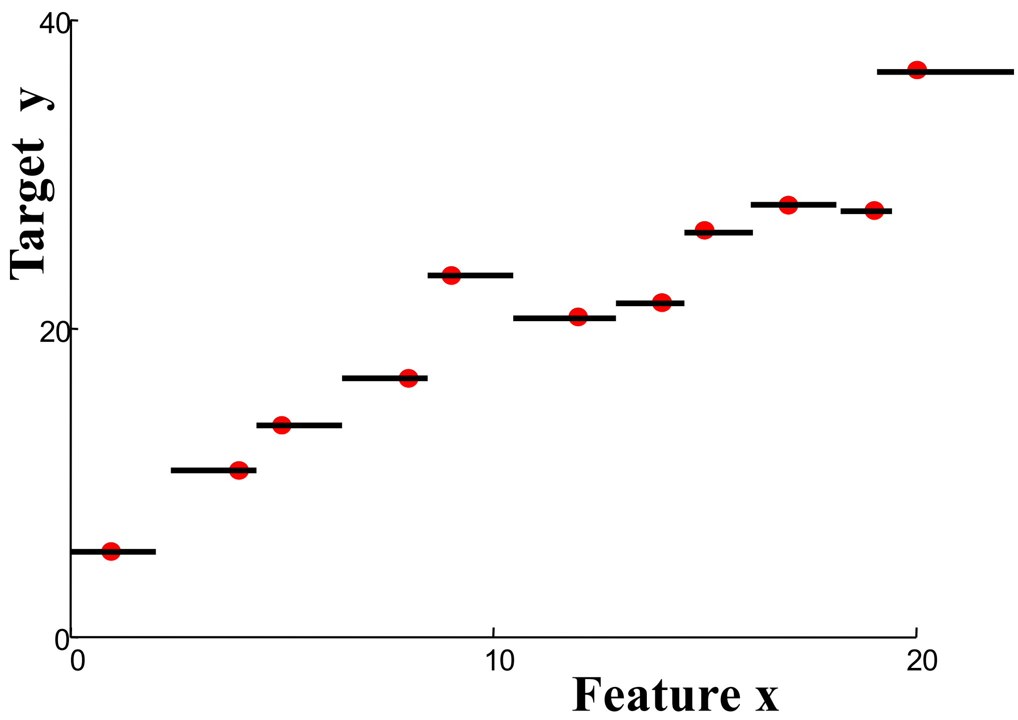
Nearest neighbor regression



"Predictor":
Given new features:
Find nearest example
Return its value

- Find training datum $x^{(i)}$ closest to $x^{(new)}$; predict $y^{(i)}$

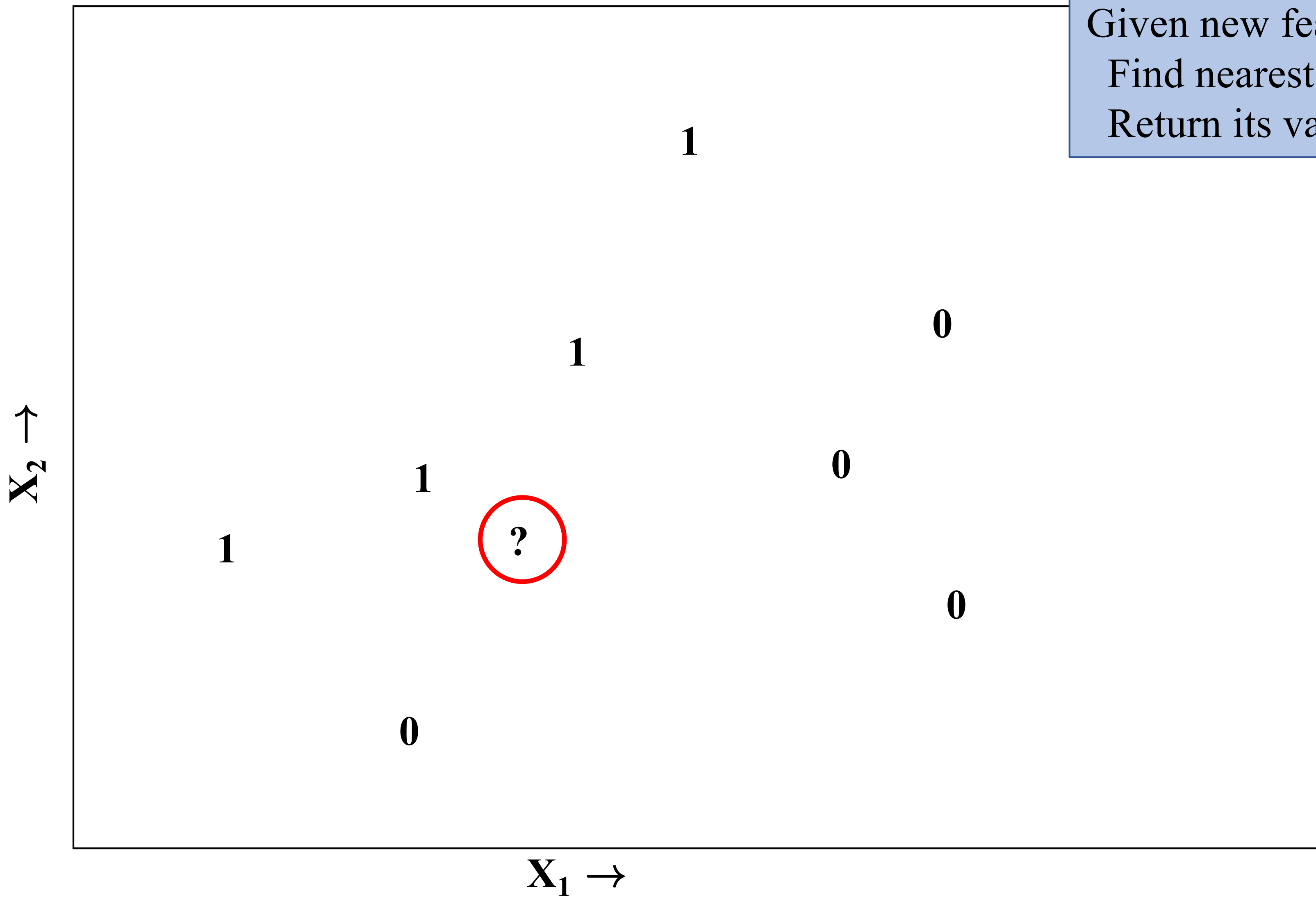
Nearest neighbor regression



"Predictor":
Given new features:
Find nearest example
Return its value

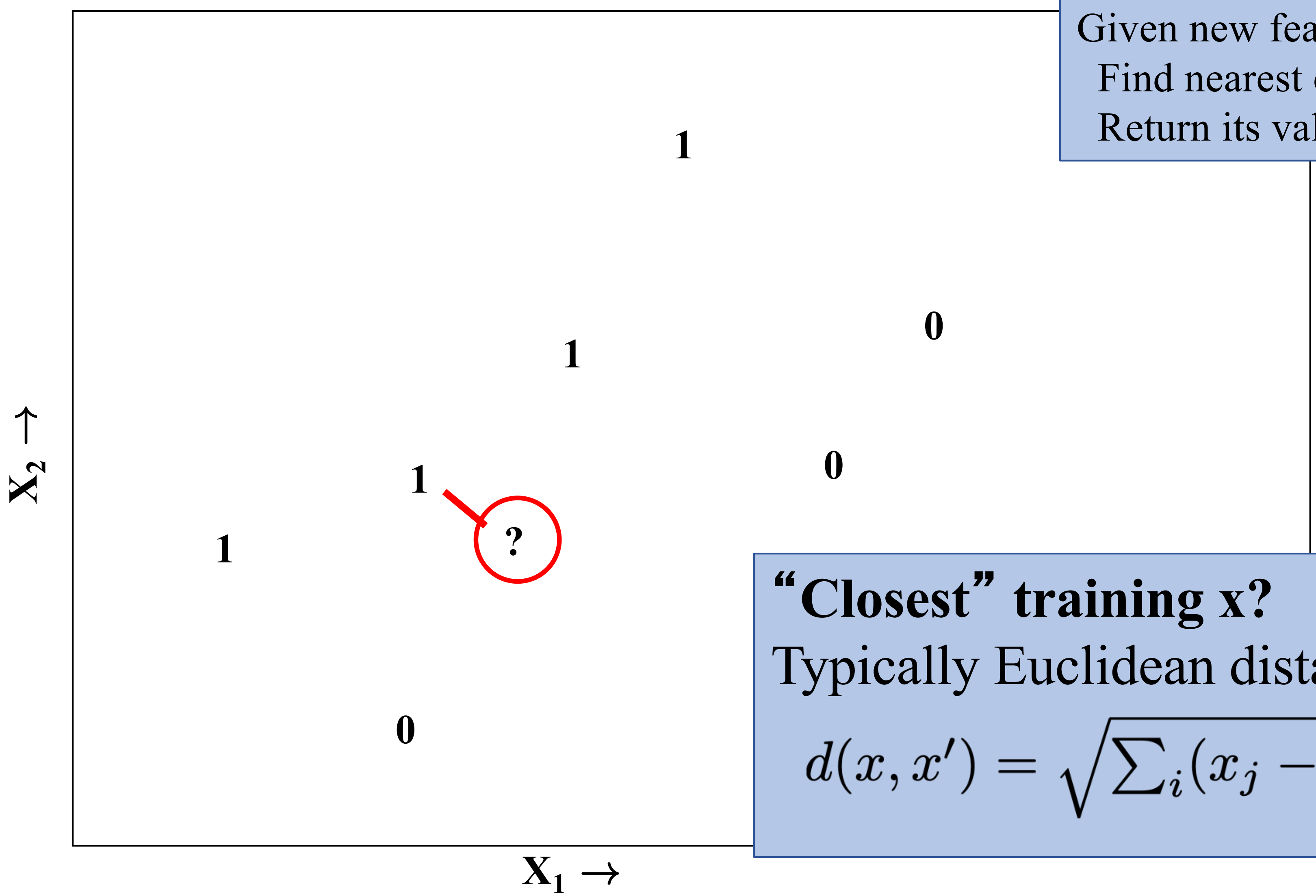
- Find training datum $x^{(i)}$ closest to $x^{(new)}$; predict $y^{(i)}$
- Defines an (implict) function f(x)
- "Form" is piecewise constant

Nearest neighbor classifier



"Predictor":
Given new features:
Find nearest example
Return its value

Nearest neighbor classifier

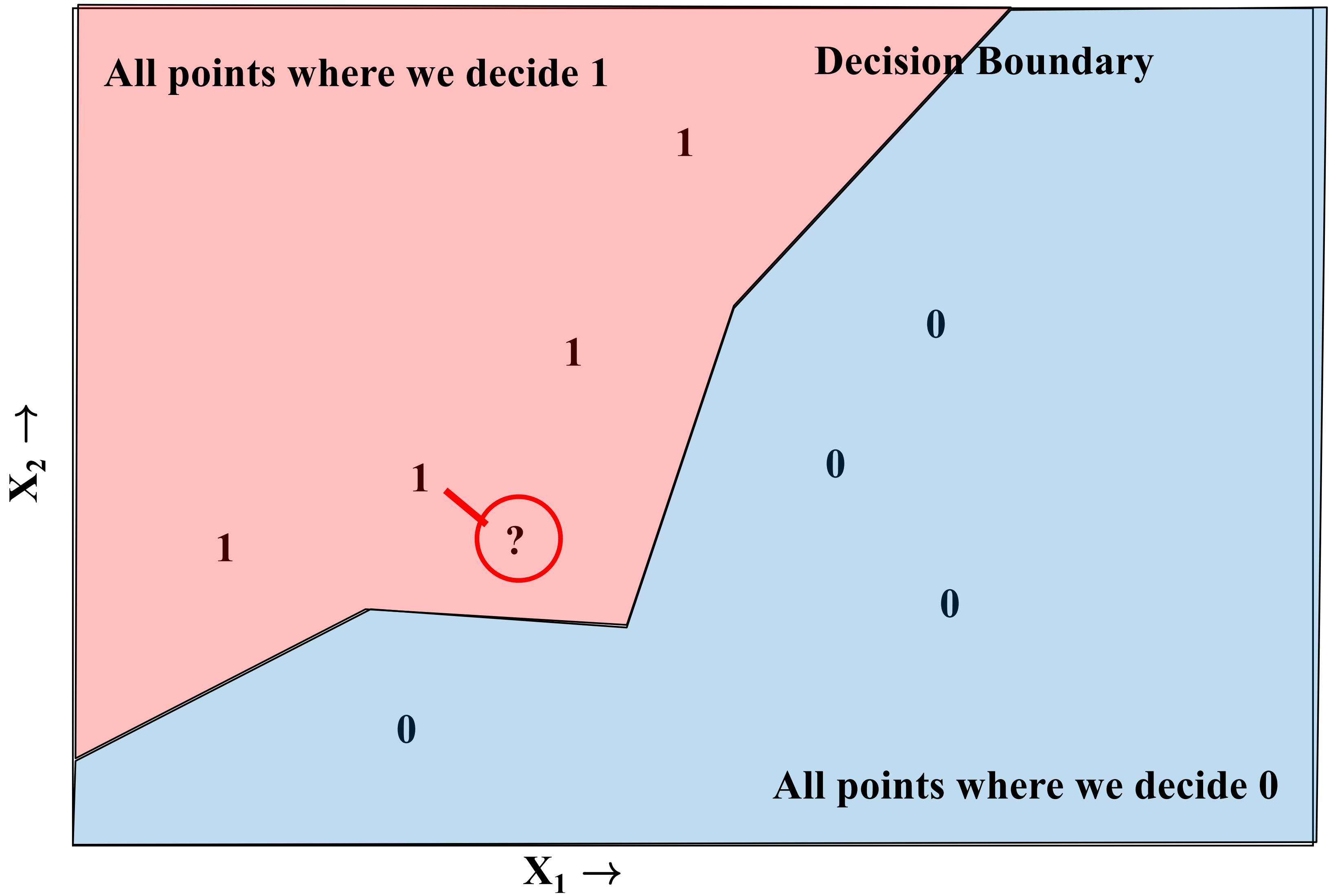


"Predictor":
Given new features:
Find nearest example
Return its value

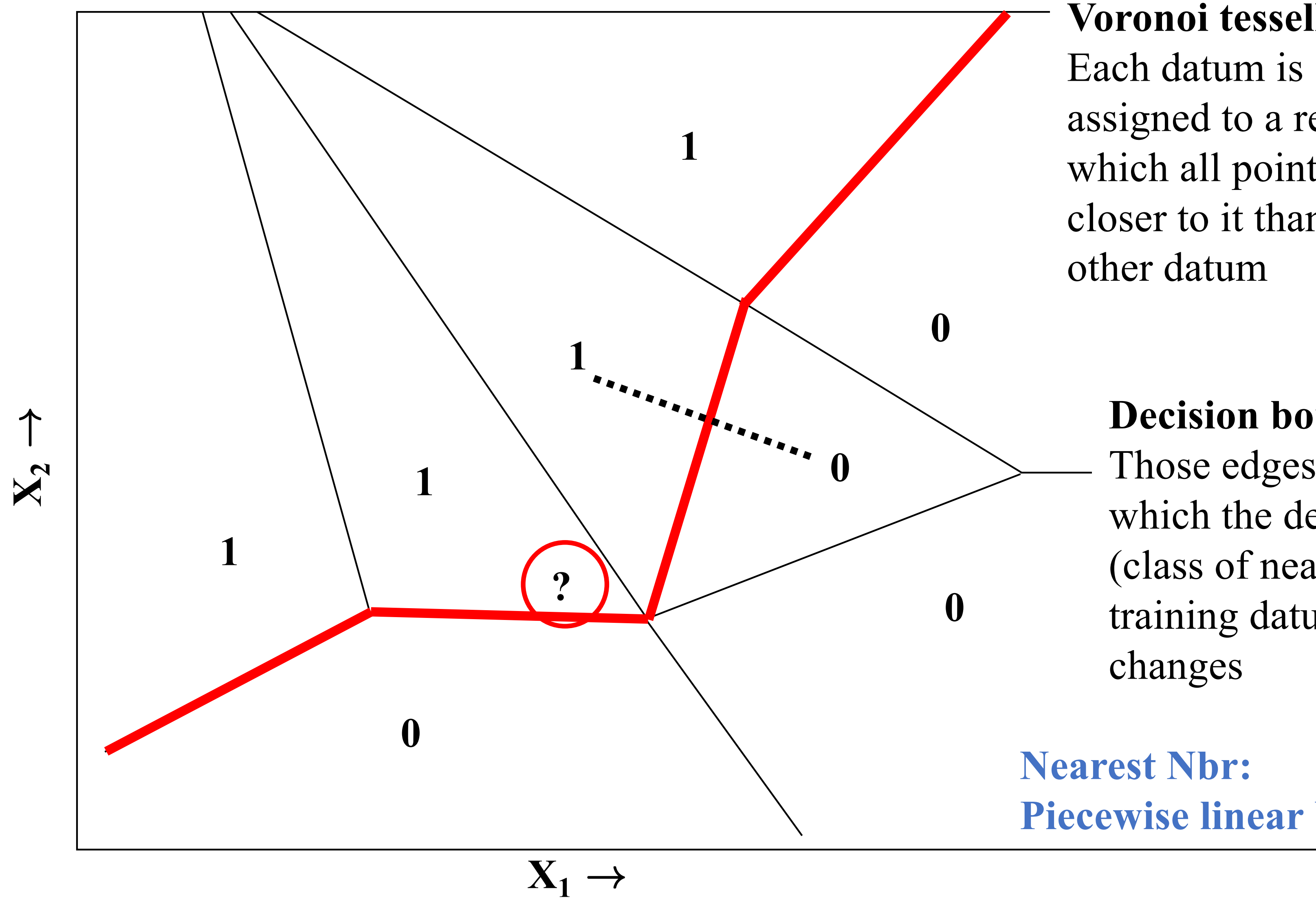
"Closest" training x?
Typically Euclidean distance:

$$d(x, x') = \sqrt{\sum_i (x_j - x'_j)^2}$$

Nearest neighbor classifier



Nearest neighbor classifier

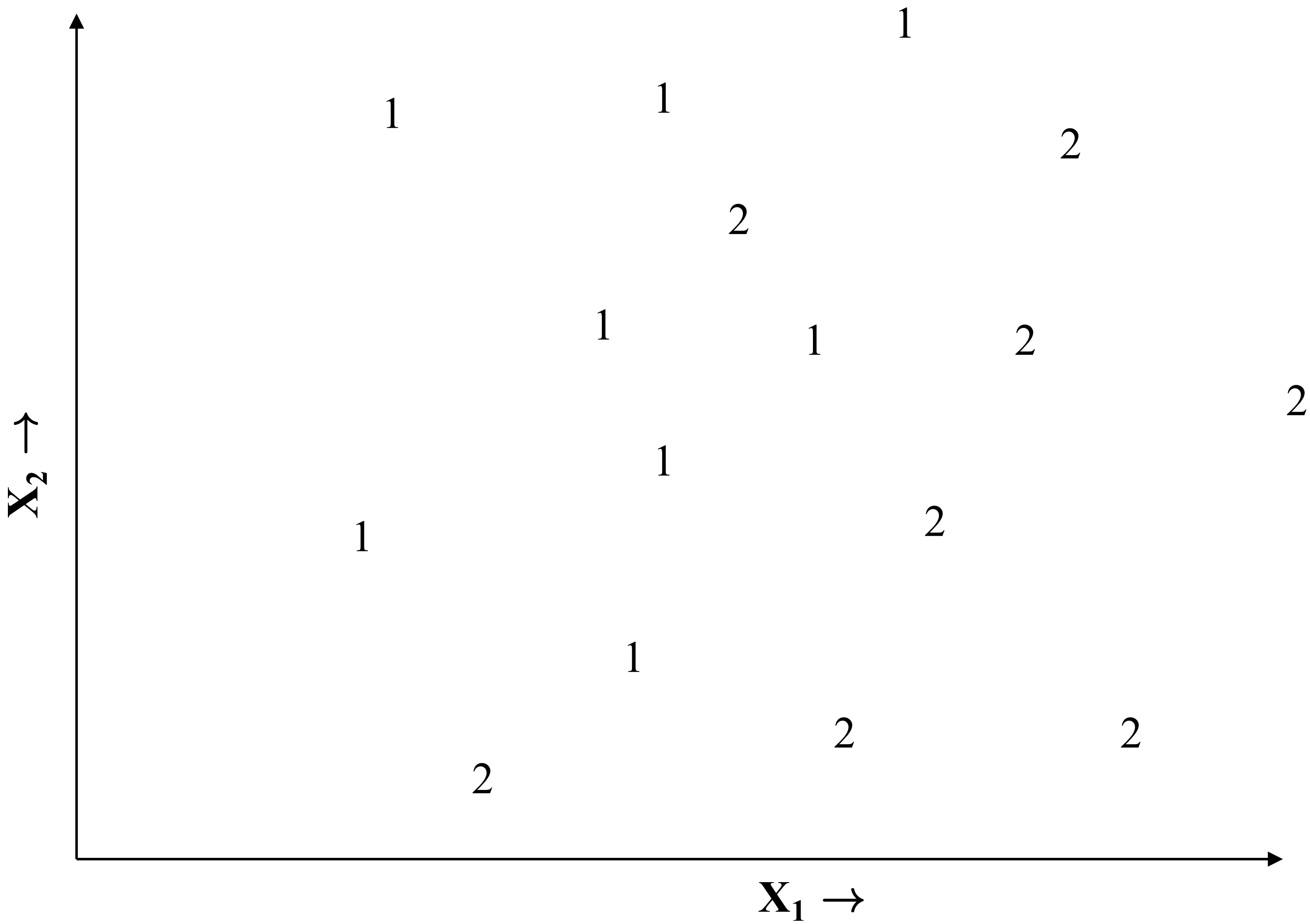


Voronoi tessellation: Each datum is assigned to a region, in which all points are closer to it than any other datum

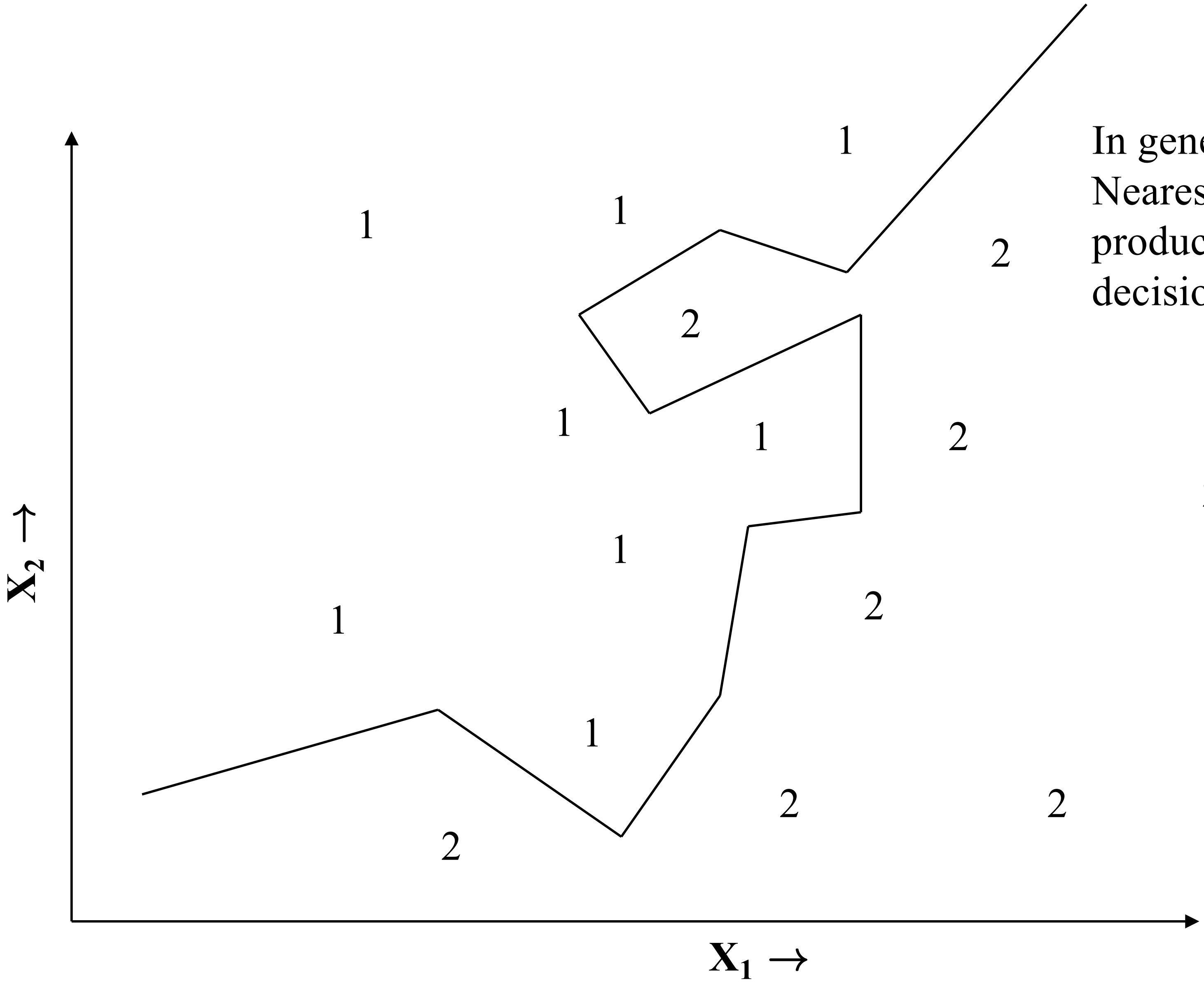
Decision boundary: Those edges across which the decision (class of nearest training datum) changes

Nearest Nbr: Piecewise linear boundary

More Data Points

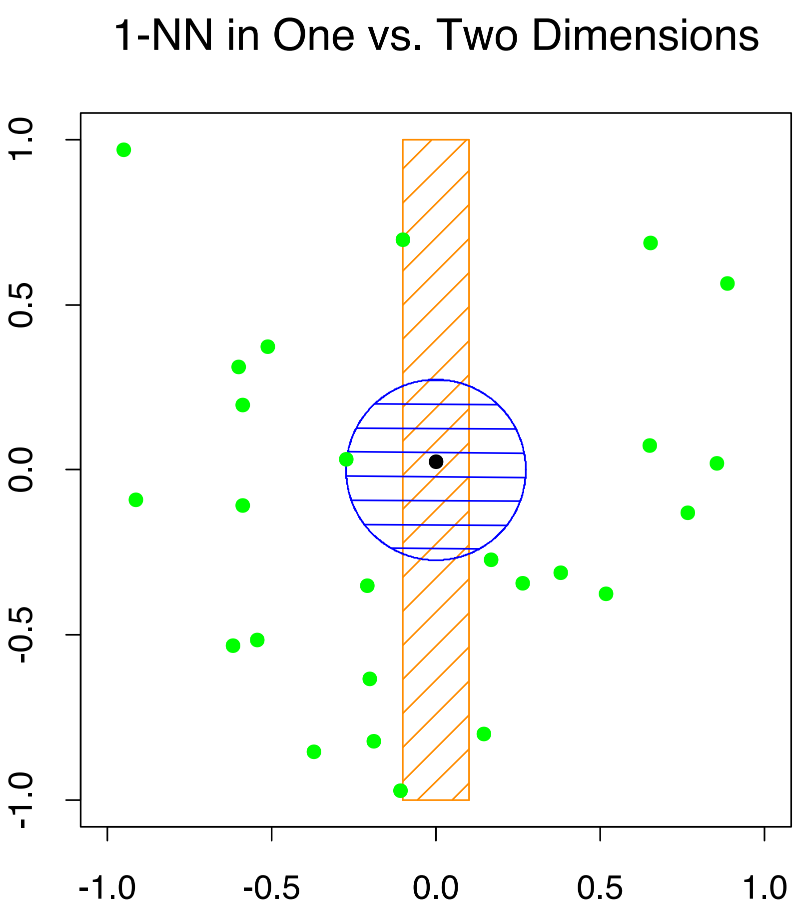


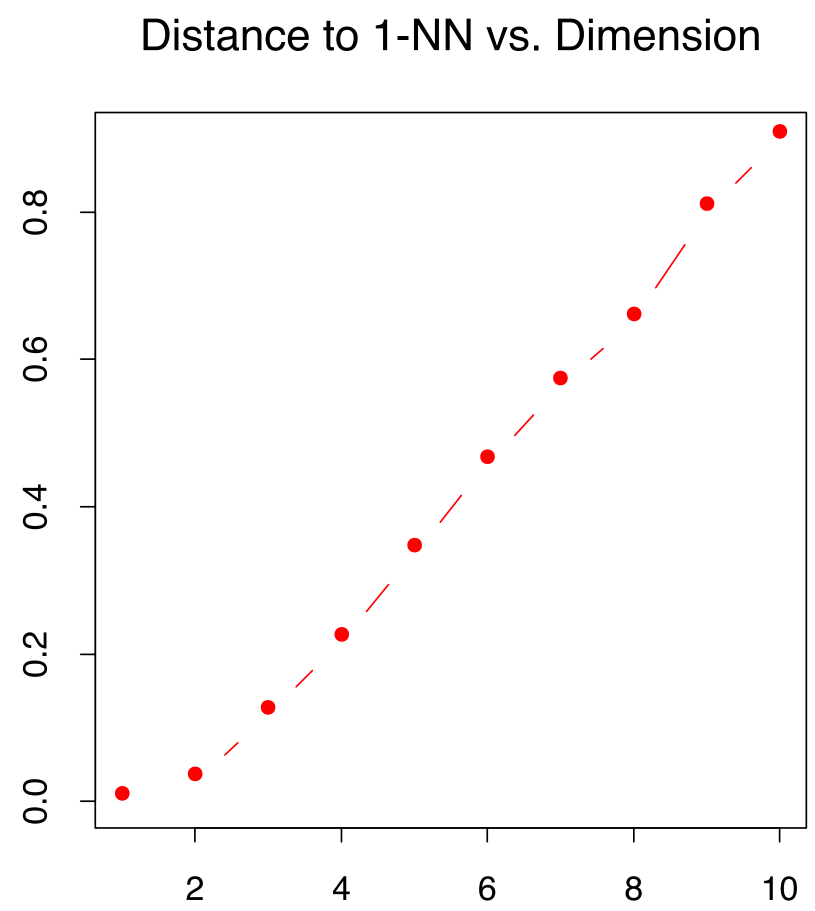
More Complex Decision Boundary



In general:
Nearest-neighbor classifier produces piecewise linear decision boundaries

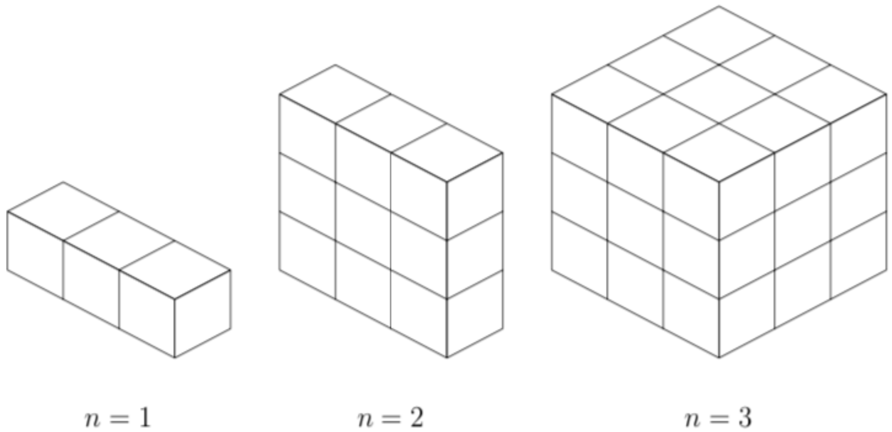
Issue: Neighbor Distance & Dimension





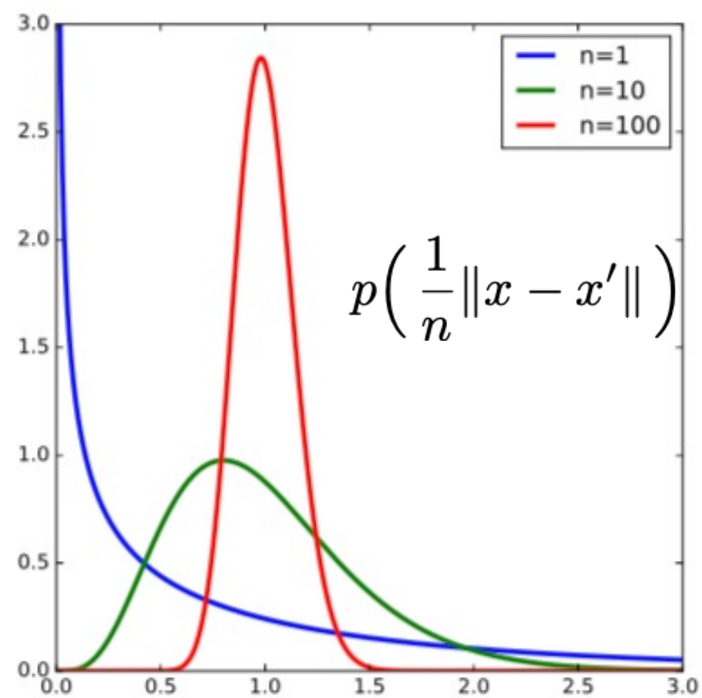
For m=1000 data in unit hypercube

The "Curse of Dimensionality"



- Function is "smooth"?
  - Want a training example that is close by (epsilon)
  - How many data do we need?

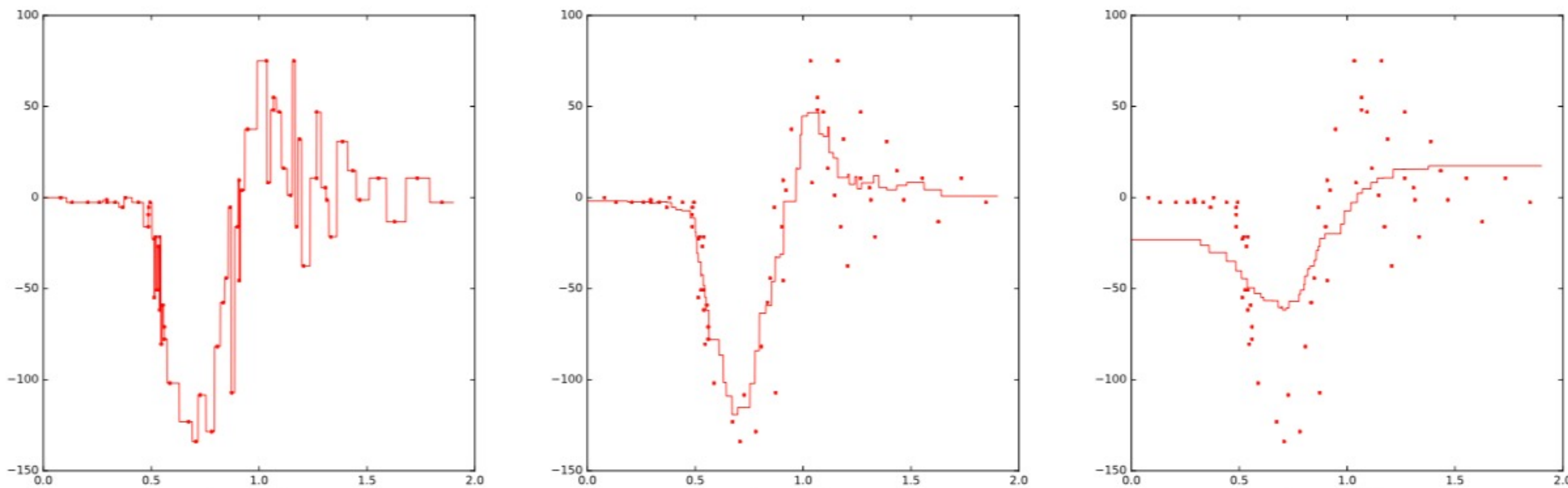
- How far away are the other data points?
  - In higher dimension, "almost all" data are equally far away!



K-Nearest Neighbor (kNN) Predictor

- Find the k-nearest neighbors to x in the data
  - i.e., rank the feature vectors according to Euclidean distance
  - select the k vectors which are have smallest distance to x

- Regression
  - Ranking gives k closest examples and their target values “y”
  - Usually just average the y-values of the k closest training examples
  - Larger k = average over a larger area for prediction

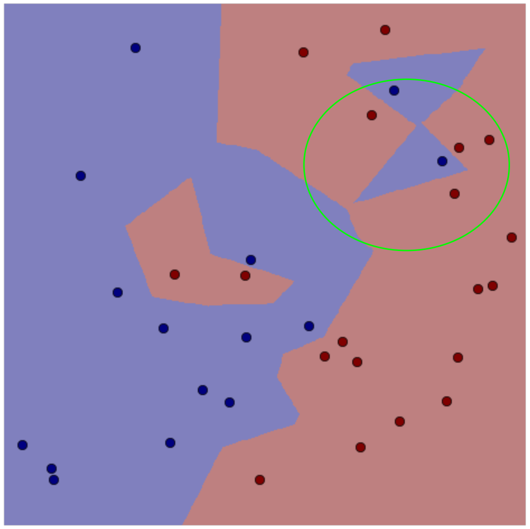


$k = 1$ $\quad$ $k = 5$ $\quad$ $k = 25$

K-Nearest Neighbor (kNN) Predictor

- Find the k-nearest neighbors to x in the data
  - i.e., rank the feature vectors according to Euclidean distance
  - select the k vectors which are have smallest distance to x

- Classification
  - ranking yields k feature vectors and a set of k class labels
  - pick the class label which is most common in this set ("vote")
  - Note: for two-class problems, if k is odd (k=1, 3, 5, …) there will never be any "ties"; otherwise, just use (any) tie-breaking rule

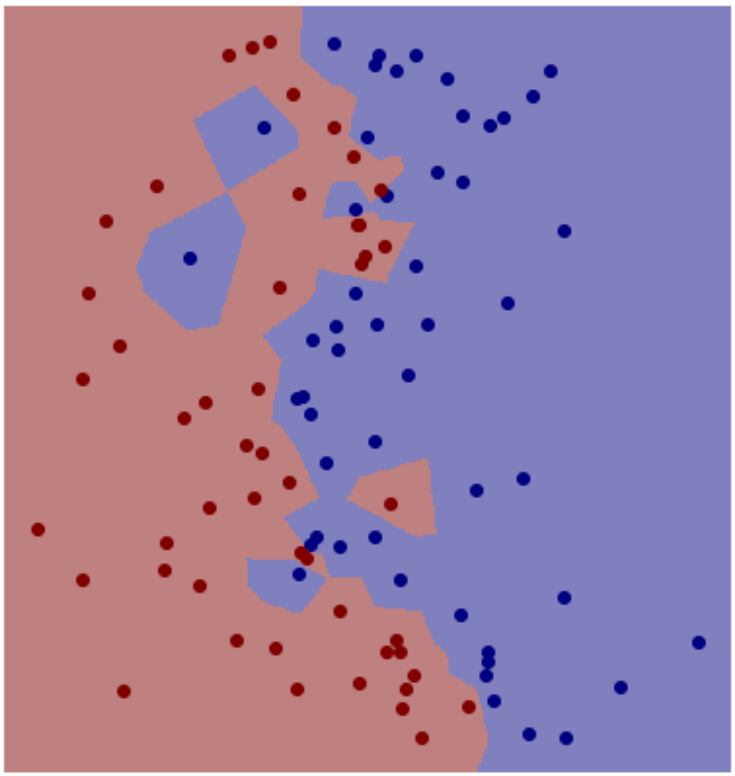


Predict blue?
Or just noise?

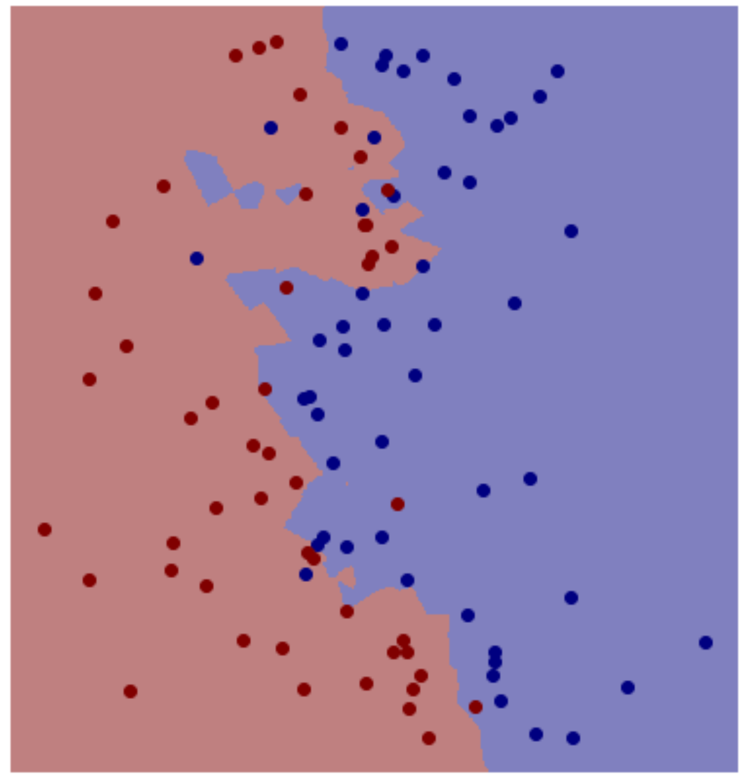
kNN Decision Boundary

- Piecewise linear decision boundary
- Increasing k “simplifies” decision boundary
    - Majority voting means less emphasis on individual points

K = 1



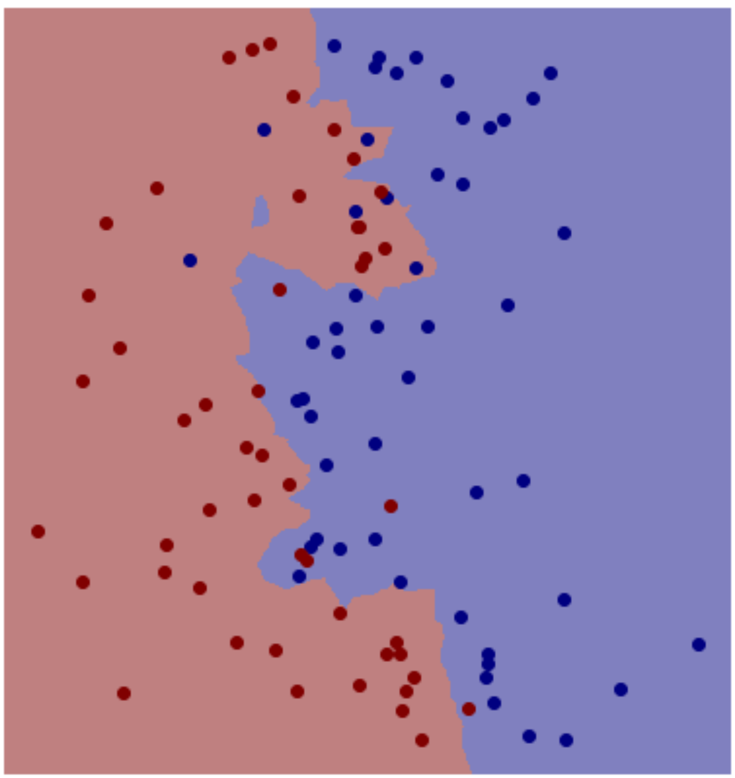
K = 3



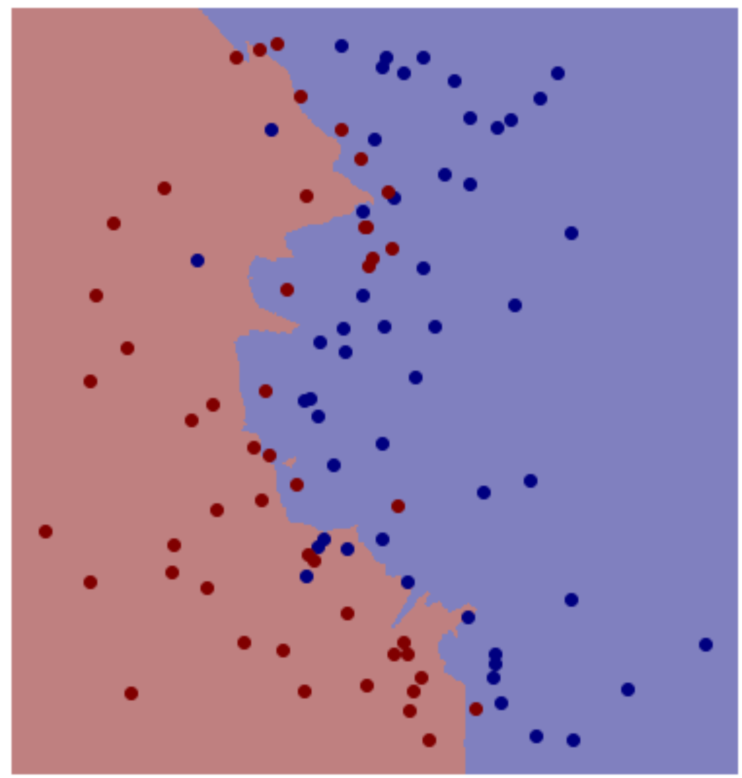
kNN Decision Boundary

- Piecewise linear decision boundary
- Increasing k “simplifies” decision boundary
  - Majority voting means less emphasis on individual points

K = 5



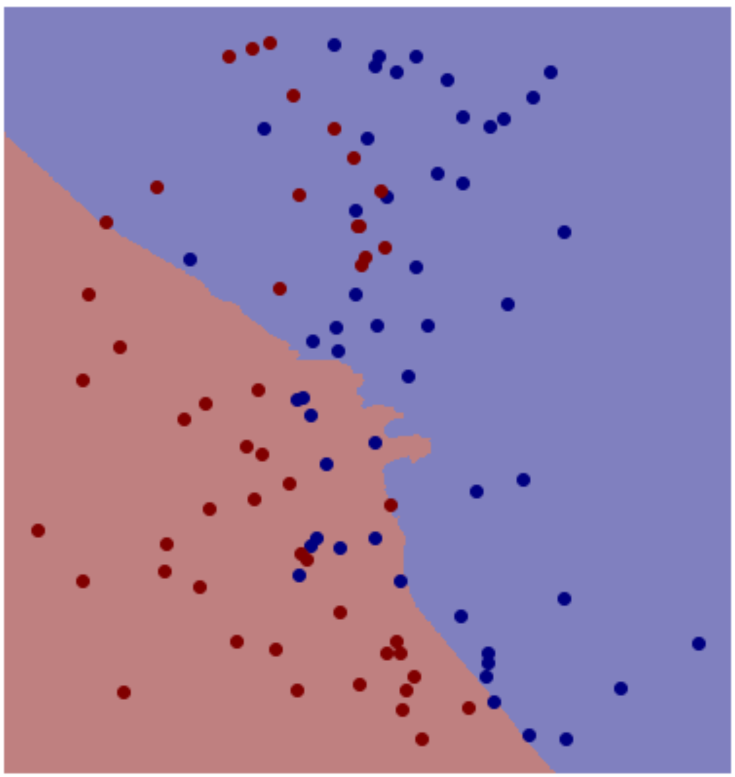
K = 20



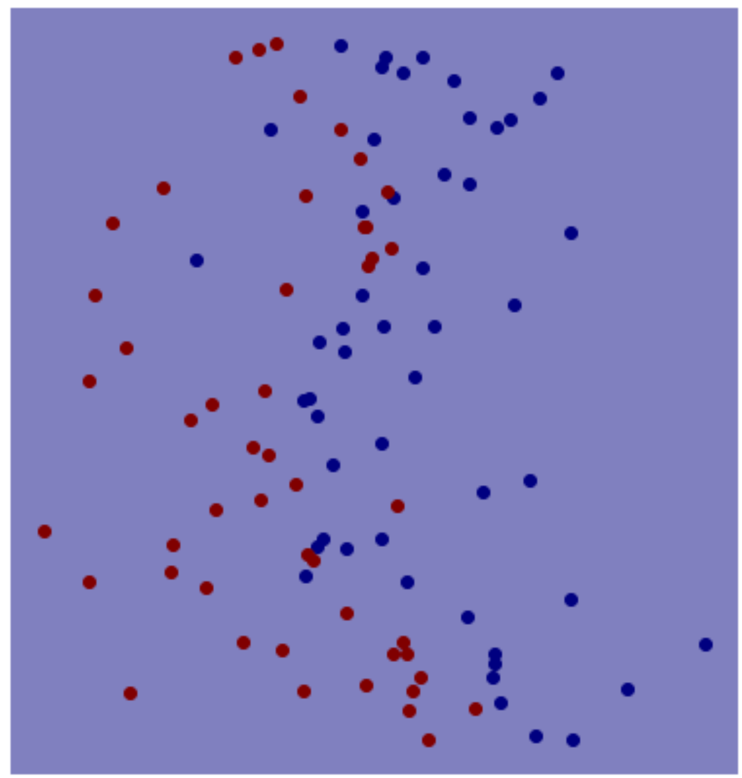
kNN Decision Boundary

- Piecewise linear decision boundary
- Increasing k "simplifies" decision boundary
  - Majority voting means less emphasis on individual points

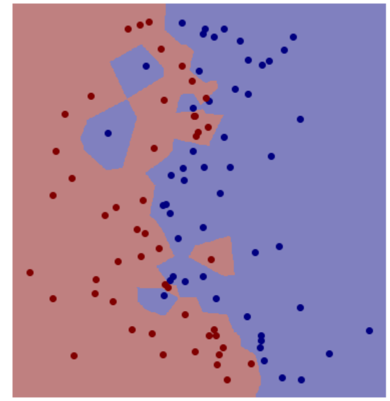
K = 70

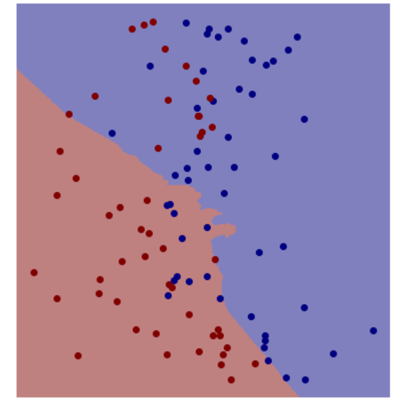


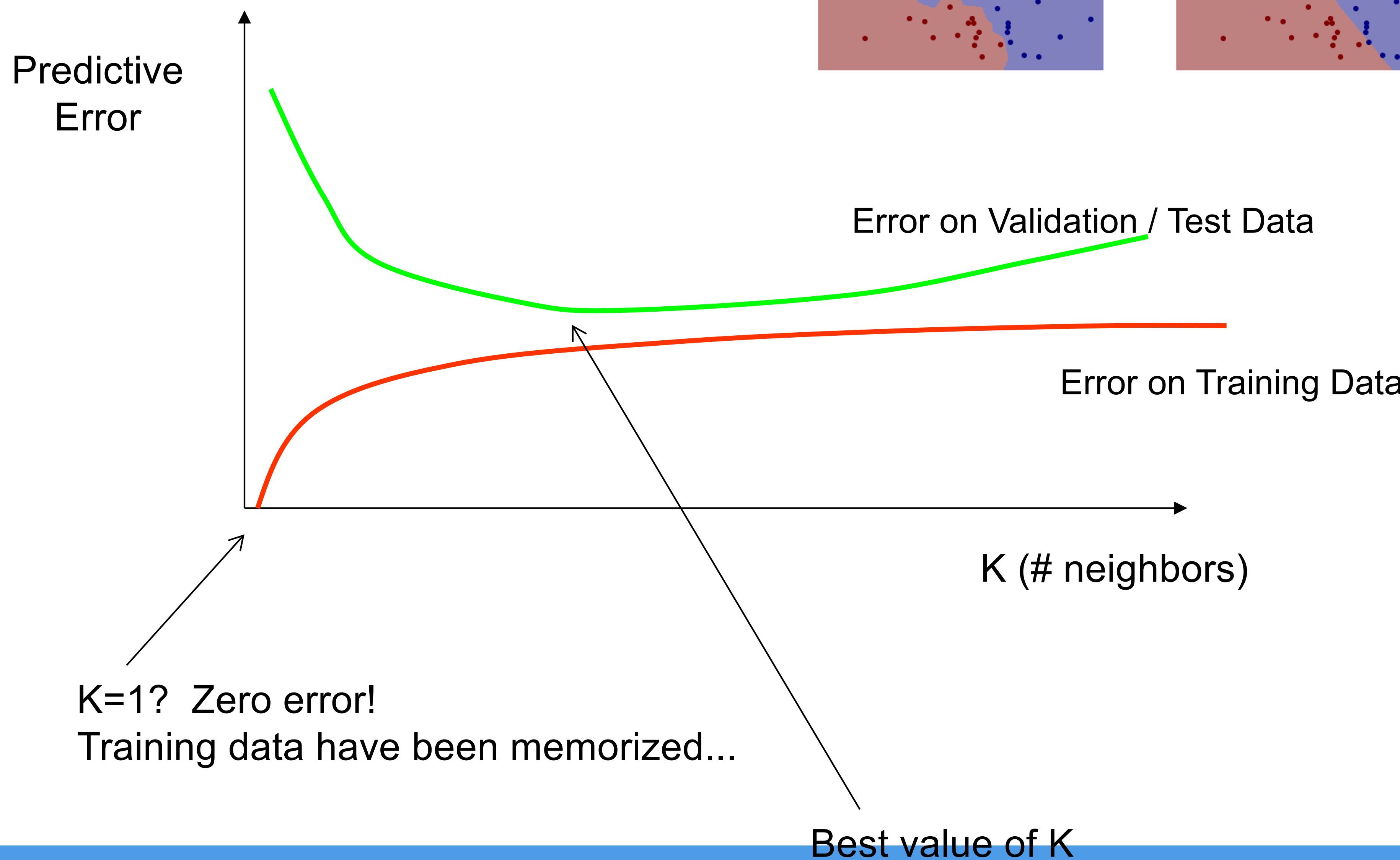
K = 100 (=m)



Error rates and K

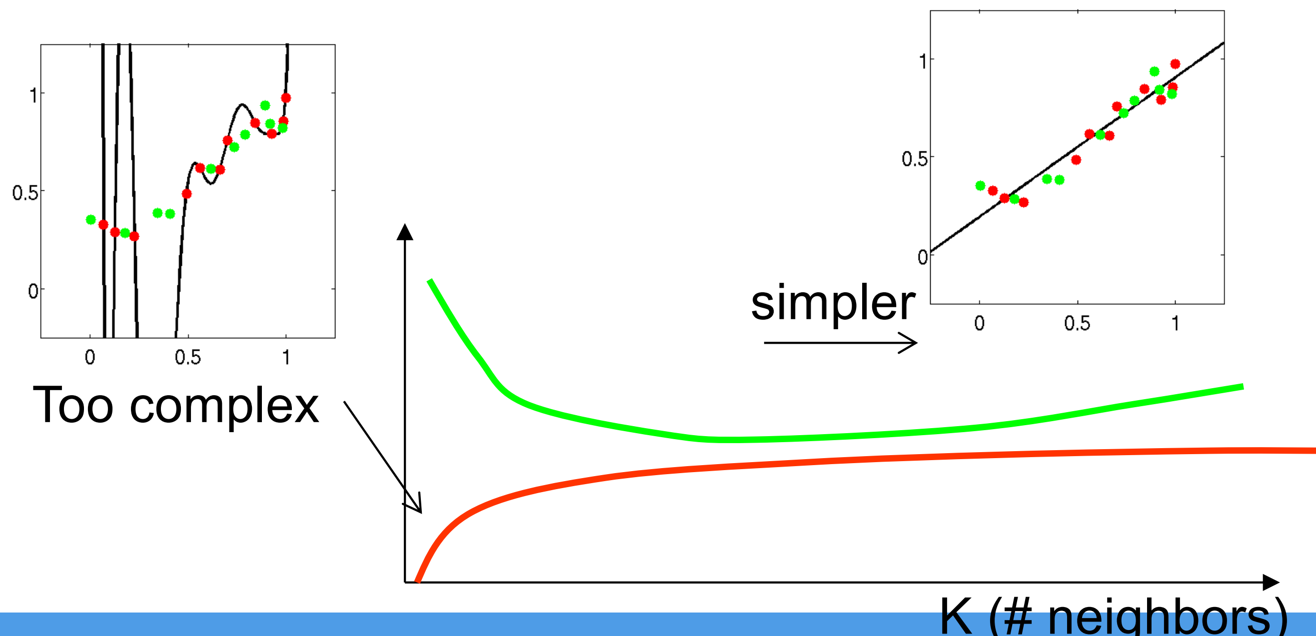






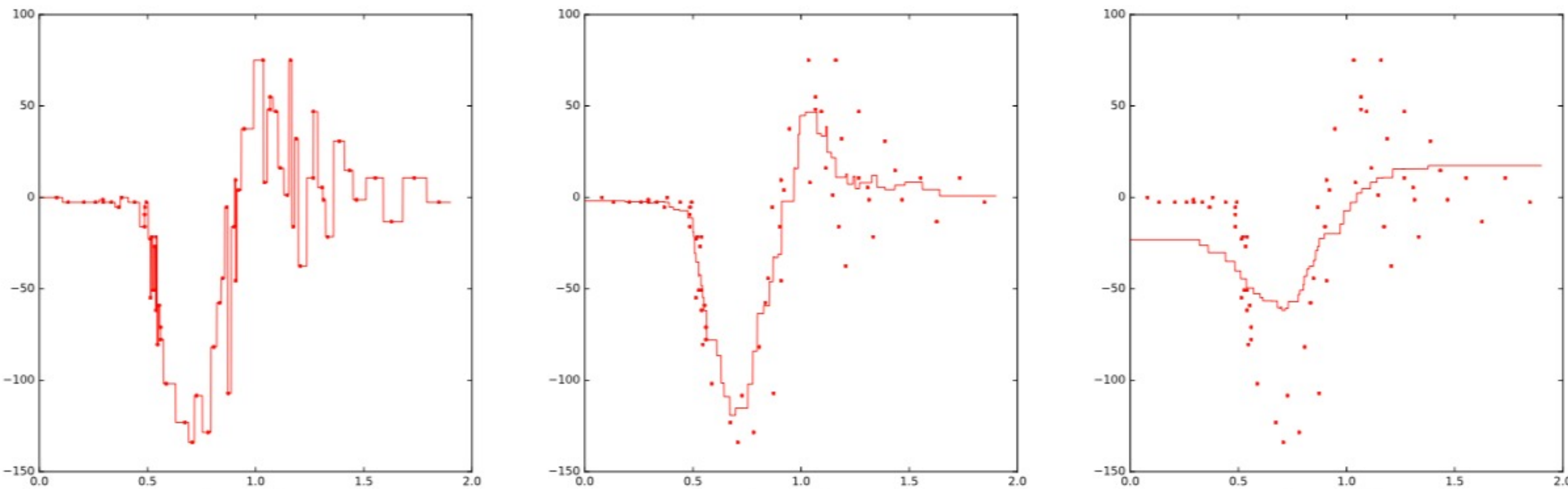
Complexity & Overfitting

- Complex model predicts all training points well
- Doesn't generalize to new data points
- k = 1 : perfect memorization of examples (complex)
- k = m : always predict majority class in dataset (simple)
- Can select k using validation data, etc.



Regression example

- Similar behavior for regression predictions:



$k = 1$

$k = 5$

$k = 25$

- K (& data density) defines a local area to average over

- K too small: follow "noise" in the data
- K too big: smooth over too large an area

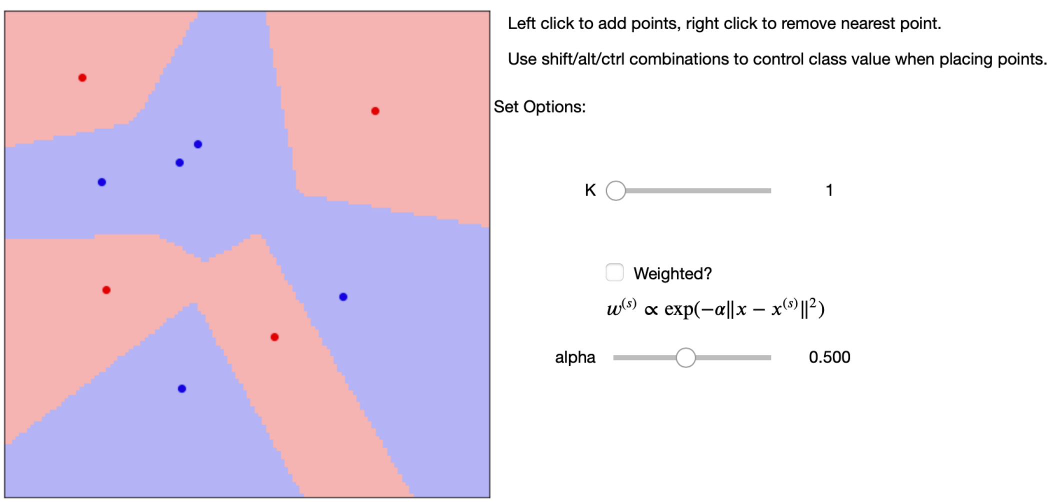
Live Demo

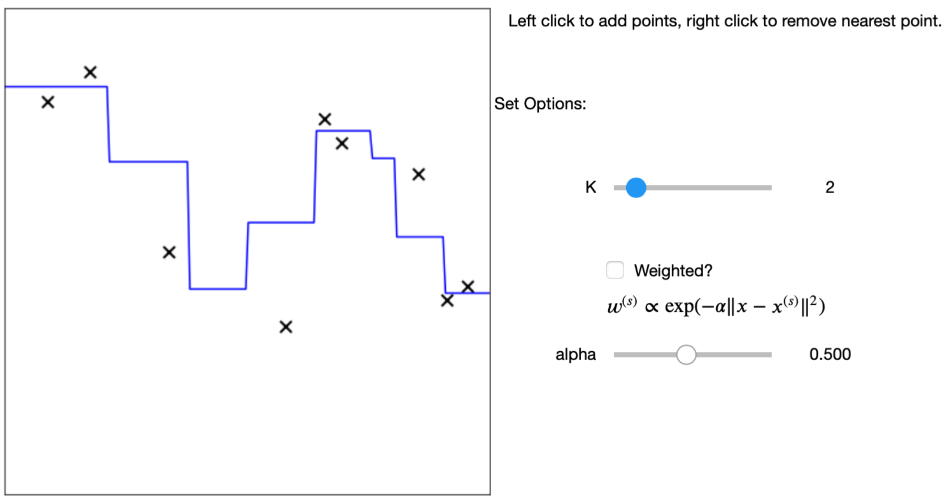
- Via Binder:

    https://mybinder.org/v2/gh/ihler/ml-demos/HEAD?filepath=notebooks%2FDemo-Live-KNN.ipynb

    (can be slow to start…)

- Ex:





K-Nearest Neighbor (kNN) Classifier

- Theoretical Considerations
  - as k increases
    - we are averaging over more neighbors
    - the effective decision boundary is more "smooth"
  - as m increases, the optimal k value tends to increase
  - k=1, m increasing to infinity : error < 2x optimal

- Extensions of the Nearest Neighbor classifier
  - Weighted distances $d(x, x') = \sqrt{\sum_i w_i (x_i - x'_i)^2}$
    - e.g., some features may be more important; others may be irrelevant
  - Fast search techniques (indexing) to find k-nearest points in d-space
  - Weighted average / voting based on distance

Digit & Hand Gesture Recognition

Athitsos et al., CVPR 2004 & PAMI 2008

Tricks to build efficient & accurate nearest-neighbor classifiers:

- Gather large training sets (possibly by generating synthetic data)
- Engineer clever distance functions that are invariant to aspects of the data unrelated to the class label
- Use algorithms to find (approximate) nearest neighbors in sub-linear time (locality sensitive hashing, class-sensitive embeddings, KD-trees, etc.)

Summary

- K-nearest neighbor models
  - Classification (vote)
  - Regression (average or weighted average)

- Piecewise linear decision boundary
  - How to calculate

- Test data and overfitting
  - Model "complexity" for knn
  - Use validation data to estimate test error rates & select k