Lecture 3

Oct 3 2008

Review of last lecture

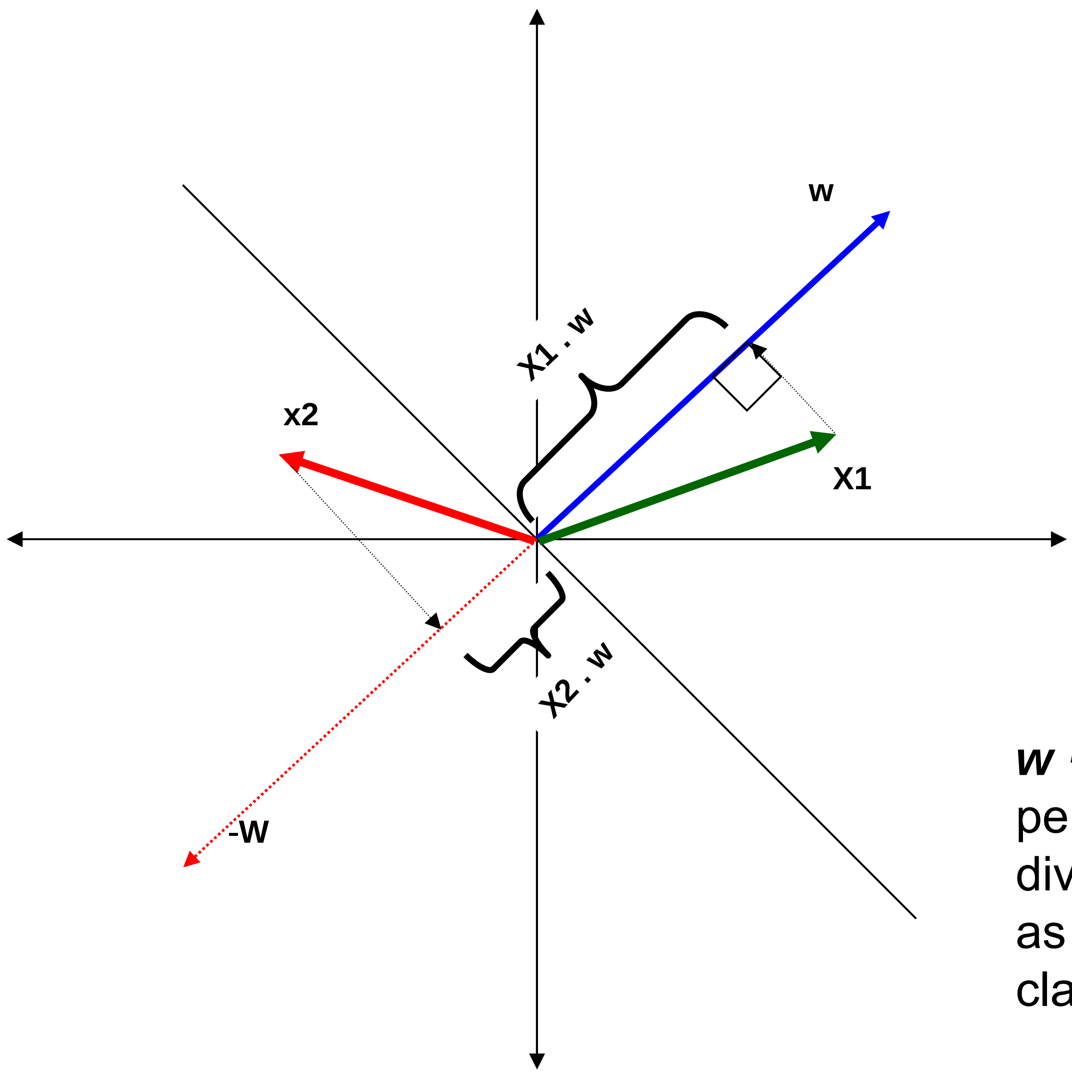
- A supervised learning example – spam filter, and the design choices one need to make for this problem
  - use *bag-of-words* to represent emails
  - *linear functions* as our functional forms to learn: *produces linear decision boundaries*
  - The perceptron algorithm for learning the function: *online vs. batch*

Reviews

- Geometric properties of a linear decision boundary as represented by

  $g(\mathbf{x},\mathbf{w}) = \boldsymbol{w} \cdot \boldsymbol{x} = 0$

  The reading posted online (by William Cohen from CMU) contains a good explanation of this.



Visually, **x** · **w** is the distance you get if you "project **x** onto **w**"

In 3d: line→plane
In 4d: plane→hyperplane
…

w* · *x = 0 gives the <u>line</u> perpendicular to **w,** which divides the points classified as positive from the points classified as negative.

Courtesy of William Cohen, CMU

Review cont

- Perceptron algorithm:
  - Start with a random ***w***
  - Update if make an mistake (what does this update do?)
- When is the perceptron algorithm guaranteed to converge?
- What happens if this is not satisfied?

Voted-Perceptron

Idea two: keep around intermediate hypotheses, and have them "vote" [Freund and Schapire, 1998]

```
Let $w_0 = (0,0,0,...,0)$
$c_0 = 0$
repeat
    Take example $i : (x^i, y^i)$
    $u^i \leftarrow \mathbf{w}_n \cdot \mathbf{x}^i$
    if $y^i \cdot u^i <= 0$
        $\mathbf{w}_{n+1} \leftarrow \mathbf{w}_n + y^i \mathbf{x}^i$
        $c_{n+1} = 0$
        $n = n + 1$
    else
        $c_n = c_n + 1$
```

Store a collection of linear separators w_0, w_1,…, along with their **survival time c_0, c_1,** …

The c's can be good measures of **reliability of the w's**.

For classification, take a weighted vote among all separators:

$$\text{sgn}\left\{\sum_{n=0}^{N} c_n \, \text{sgn}(w_n \cdot x)\right\}$$

What is now we have more than two classes?

- We learn one LTU for each class

$$h_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x} \qquad k = 1, \dots, c$$

  - The training is done on a transformed data set where class k examples are considered positive, the others considered negative

- Classify ***x*** to according to

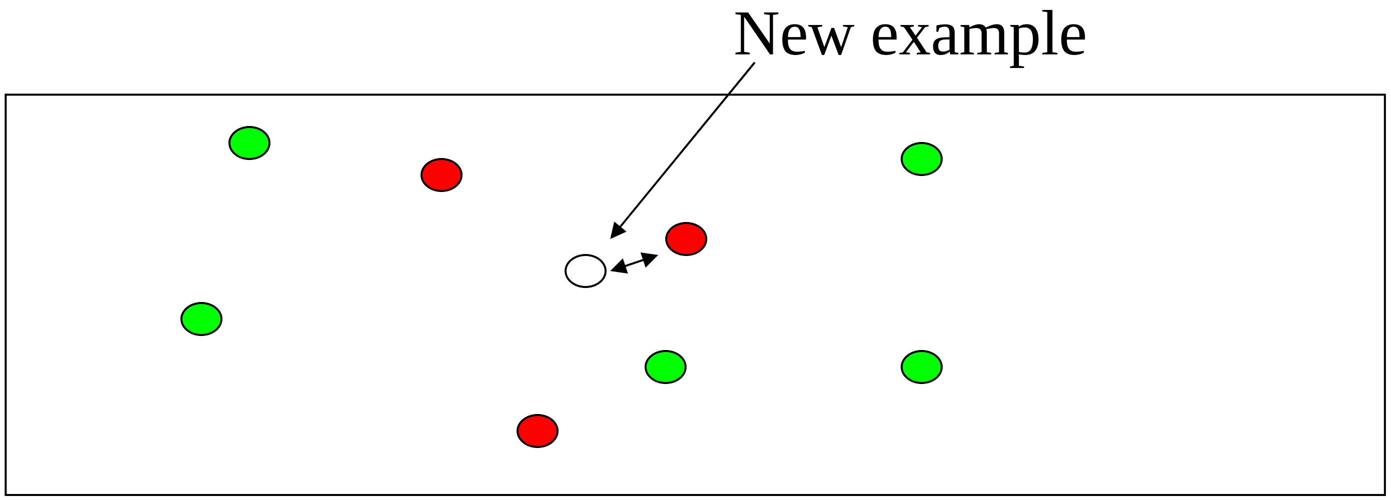
$$\widehat{y} = \arg\max_k h_k(\mathbf{x})$$

- This is called a *linear machine*

When the data is not linearly separable, a different approach is to classify an email by asking the question “ which of the training email does this one look most similar to” – this is the basic idea behind our next learning algorithm

Nearest Neighbor Algorithm

- Remember all training examples
- Given a new example **x,** find the its closest training example <**x**^{i}, *y*^{i}> and predict y^i



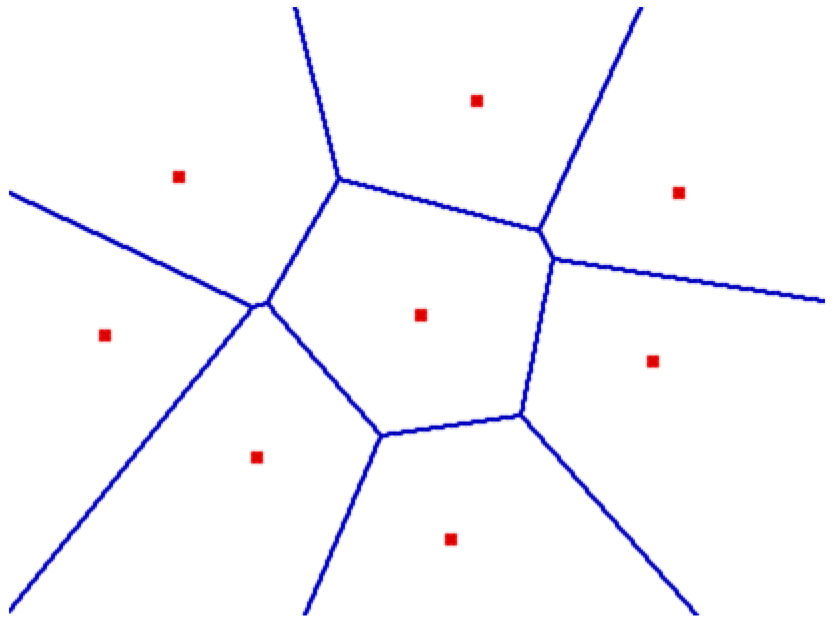
- Euclidean distance (straight line distance):

$$\left\|\mathbf{x}-\mathbf{x}^i\right\|^2=\sum_j (x_j - x_j^i)^2$$

Note that || * || represents the length (magnitude) of the vector. | * | is mainly used for norm of a scalar.

Decision Boundaries: The Voronoi Diagram

- Given a set of points, a Voronoi diagram describes the areas that are nearest to any given point.
- These areas can be viewed as zones of control.



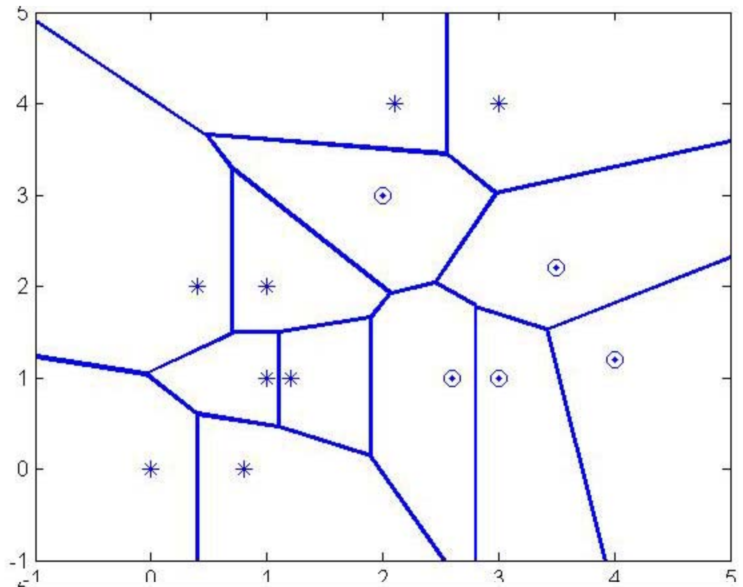
Voroni diagram

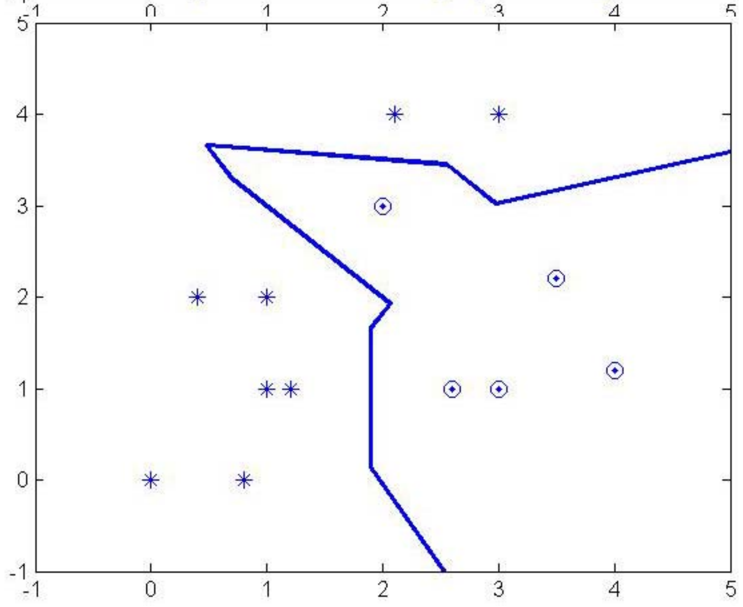
- Demo

  http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide/index.html.en

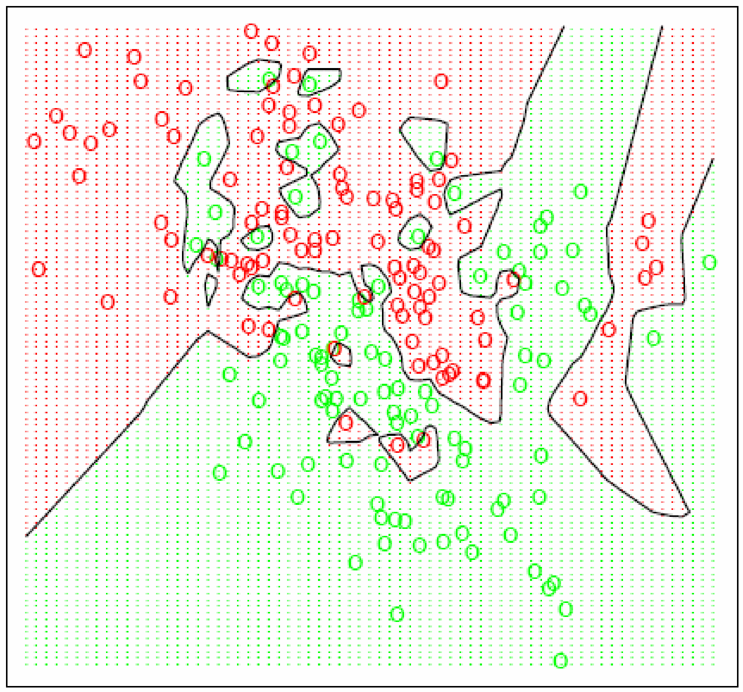
Decision Boundaries: Subset of the Voronoi Diagram

- Each example controls its own neighborhood
- Create the voroni diagram
- Decision boundary are formed by only retaining these line segments separating different classes.
- The more examples stored, the more complex the decision boundaries can become





Decision Boundaries



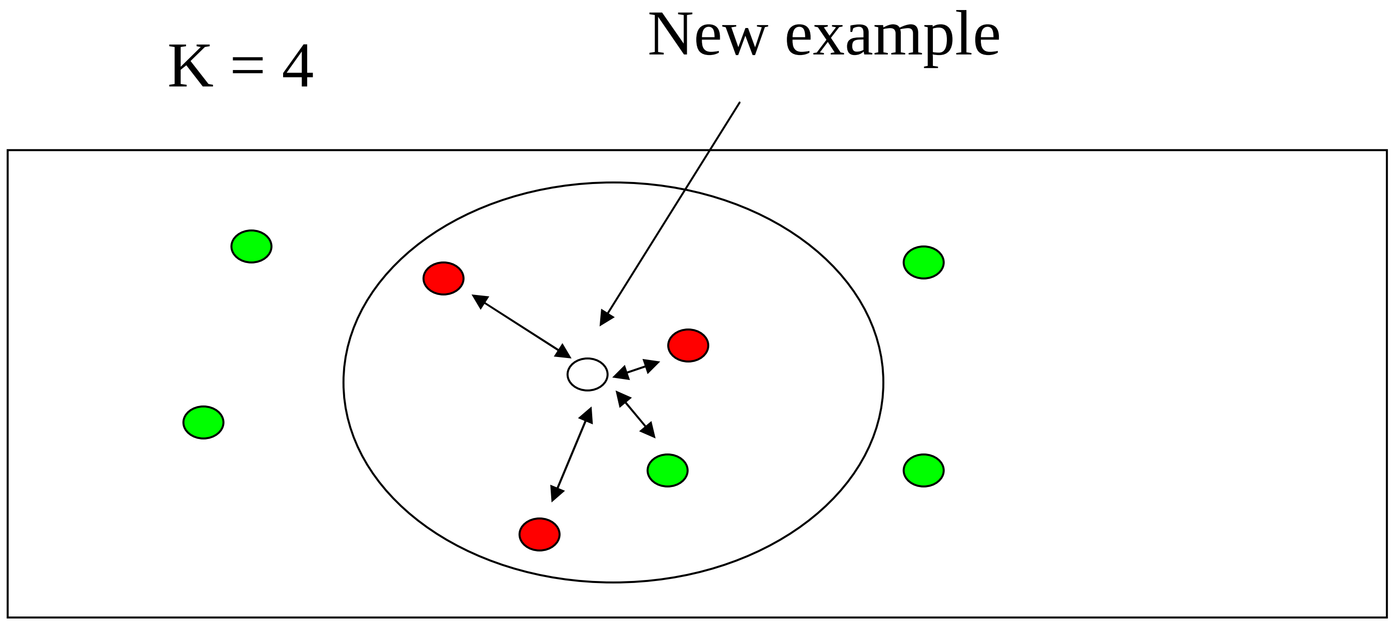
With large number of examples and noise in the labels, the decision boundary can become nasty!

How to deal with this issue?

K-Nearest Neighbor

Example:

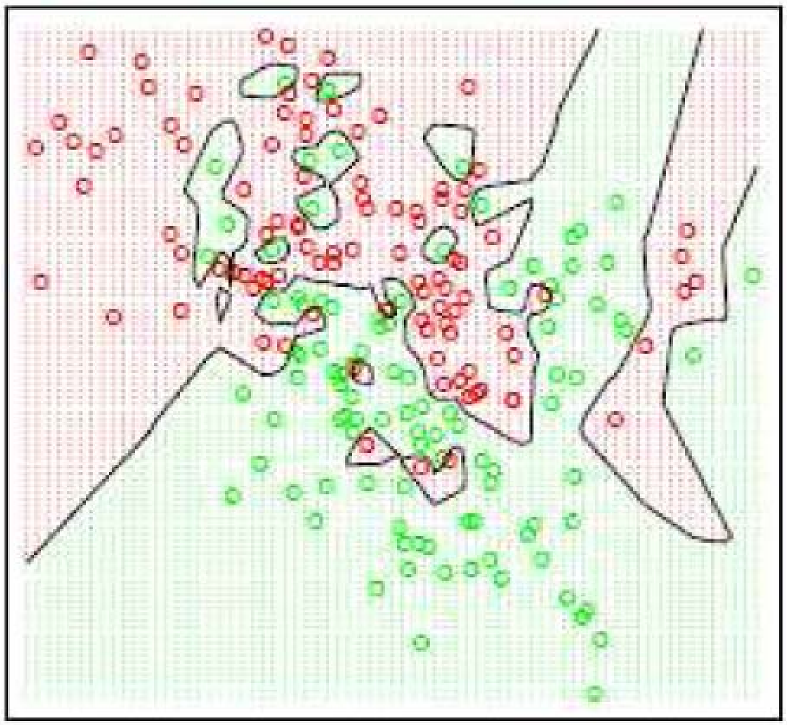
K = 4



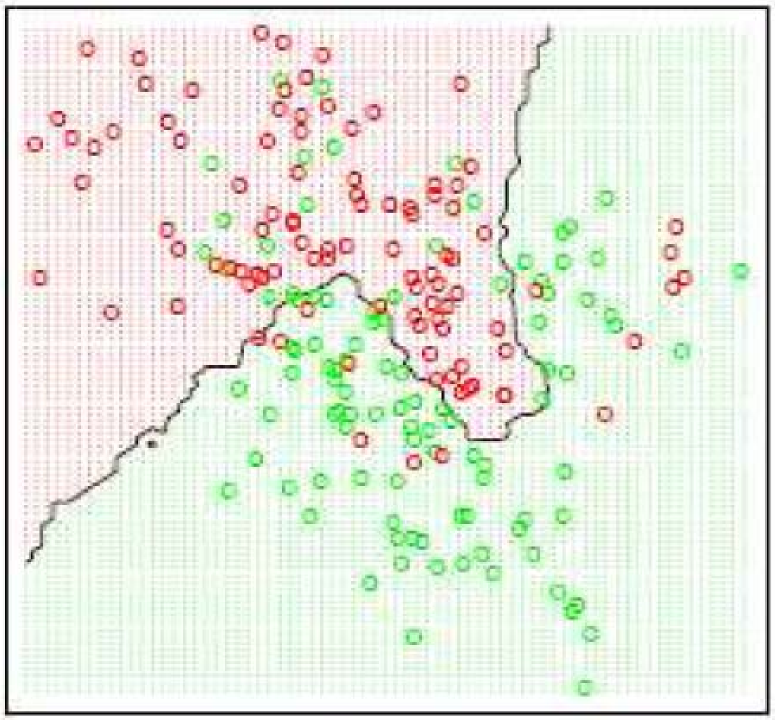
Find the ***k*** nearest neighbors and have them vote.

Effect of K

K=1



K=15



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Larger k produces smoother boundaries, why?

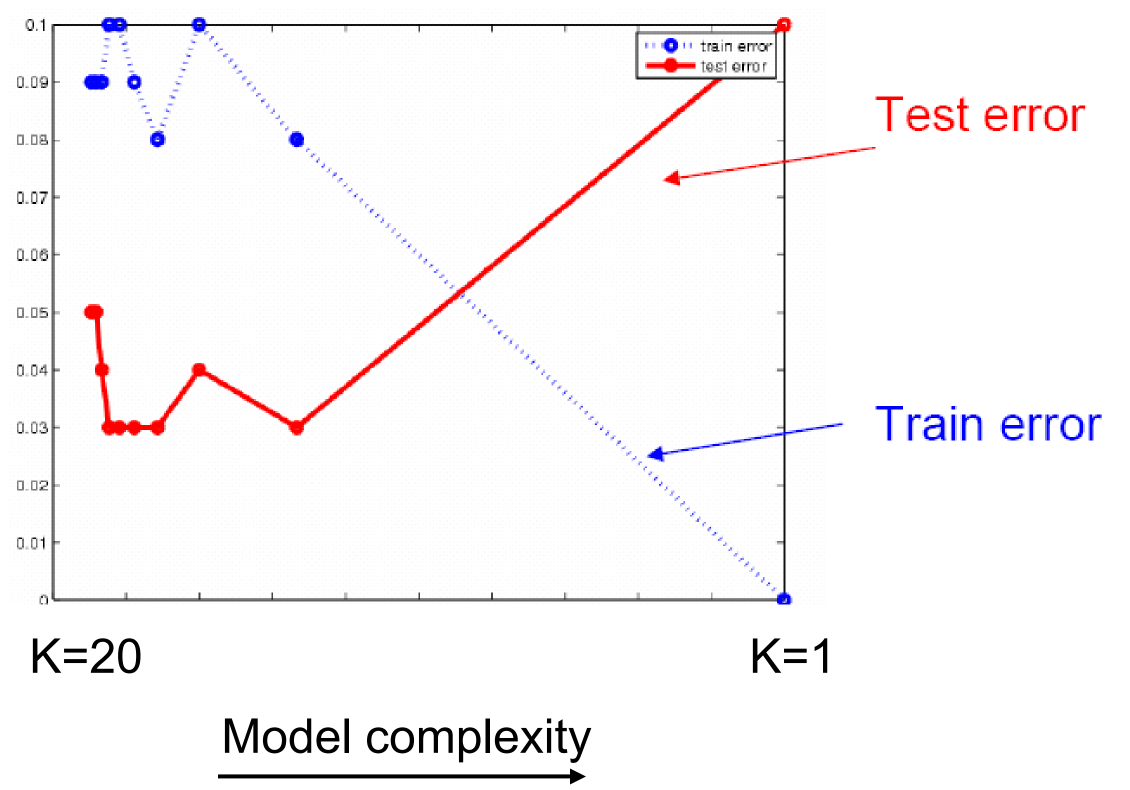
- The impact of class label noises canceled out by one another

But when k is too large, what will happen?

- Oversimplified boundaries, say k=N, we always predict the majority class

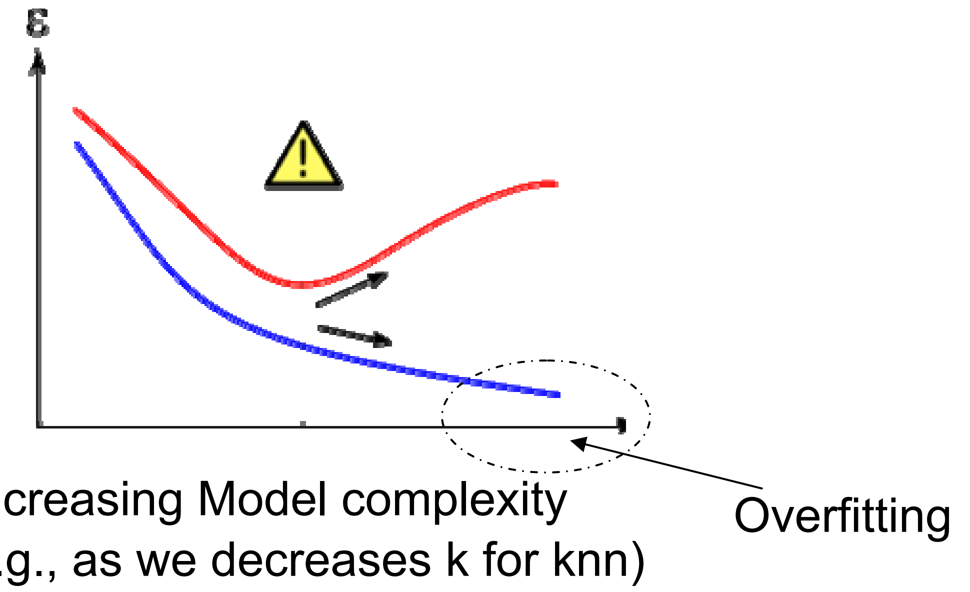
Question: how to choose k?

- Can we choose k to minimize the mistakes that we make on training examples (*training error*)?
- Question: 1-nn's training error is 0, why is that?



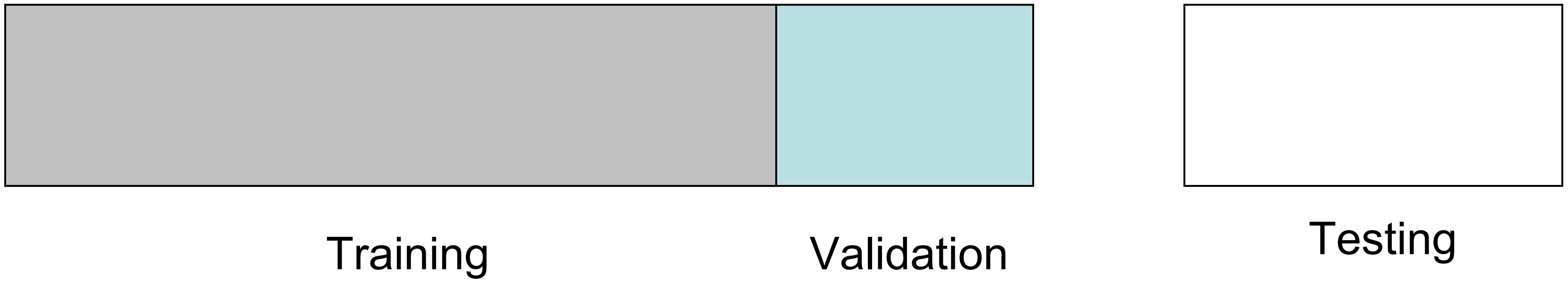
Model Selection

- Choosing k for k-nn is just one of the many model selection problems we face in machine learing
  - Choosing k-nn over LTU is also a model selection problem
  - This is a heavily studied topic in machine learning, and is of crucial importance in practice
- If we use training error to select models, we will always choose more complex ones

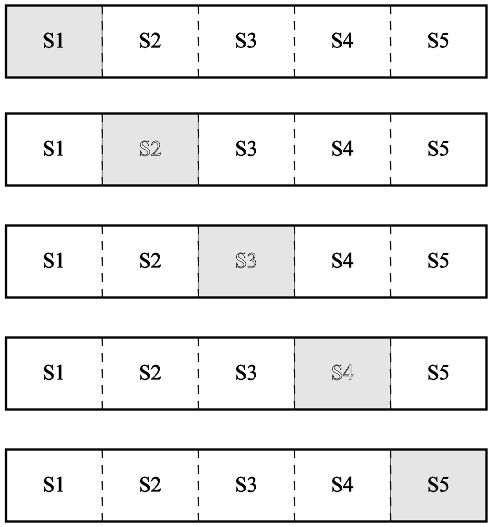


Use a Validation Set

- We can keep part of the labeled data apart as validation data
- Evaluate different k values based on the prediction accuracy on the validation data
- Choose k that minimize validation error



- When labeled set is small, we might not be able to get a big enough validation set (why do we need large validation set?)
- Solution: cross validation



Train on S2, S3, S4, S5, test on S1 $\longrightarrow \varepsilon_1$

Train on S1, S3, S4, S5, test on S2 $\longrightarrow \varepsilon_2$

⋮ ⋮

Train on S1, S2, S3, S4, test on S5 $\longrightarrow \varepsilon_5$

$$\boxed{\varepsilon = \frac{1}{5}\sum_{i=1}^{5}\varepsilon_i}$$

A 5-fold cross validation

Practical issues with KNN

- Suppose we want to build a model to predict a person’s shoe size
- Use the person’s height and weight to make the prediction
- P1: (6’, 175), P2: (5.7,168), PQ:(6.1’, 170)

$$\mathbf{D(PQ,P1)} = \sqrt{0.1^2 + 5^2} \approx 5 \qquad \mathbf{D(PQ,P2)} = \sqrt{0.4^2 + 2^2} \approx 2.04$$

- There is a problem with this what is it?

Because weight has a much larger range of values, the differences look bigger numerically.

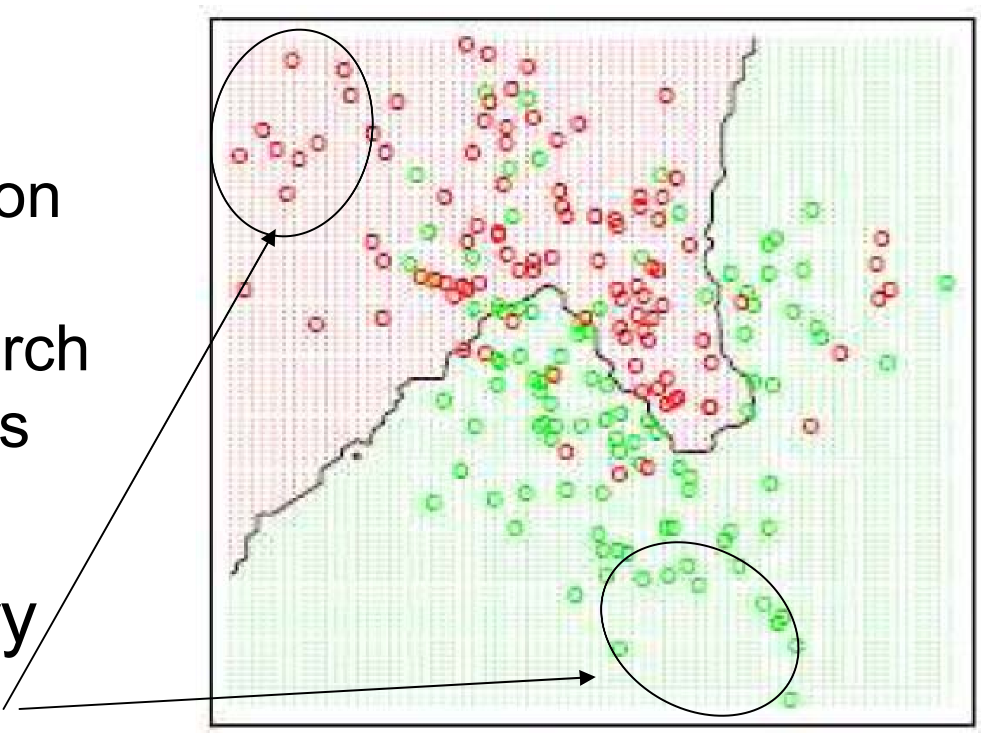
Features should be normalized to have the same range of values (e.g., [0,+1]), otherwise features with larger ranges will be treated as more important.

Practical issues with KNN

- Our data may also contain the GPAs
- Should we include this attribute into the calculate?
- When collecting data, people tend to collect as much information as possible regardless whether they are useful for the question in hand
- Recognize and remove such attributes when building your classification models

Other issues

- It can be computationally expensive to find the nearest neighbors!
  - Speed up the computation by using smart data structures to quickly search for approximate solutions
- For large data set, it requires a lot of memory
  - Remove unimportant examples



Final words on KNN

- KNN is what we call *lazy learning (vs. eager learning)*
  - Lazy: learning only occur when you see the test example
  - Eager: learn a model before you see the test example, training examples can be thrown away after learning
- Advantage:
  - Conceptually simple, easy to understand and explain
  - Very flexible decision boundaries
  - Not much learning at all!
- Disadvantage
  - It can be hard to find a good distance measure
  - Irrelevant features and noise can be very detrimental
  - Typically can not handle more than 30 attributes
  - Computational cost: requires a lot computation and memory