Nearest neighbor & VC-dimension

Lecture 9, 10/5/17

David Sontag

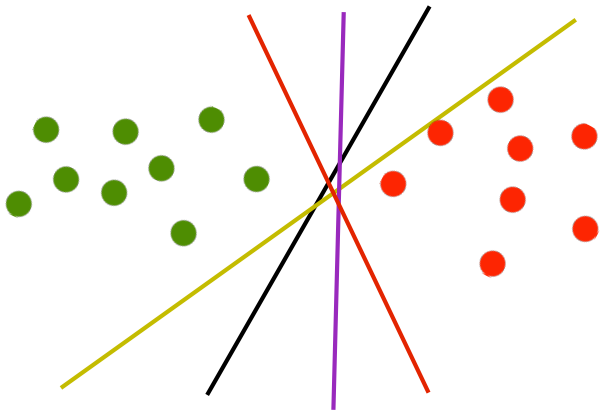
Massachusetts
Institute of
Technology

Acknowledgement: several slides adapted from Carlos Guestrin, Luke Zettlemoyer, Tom Mitchell, and Sanjoy Dasgupta

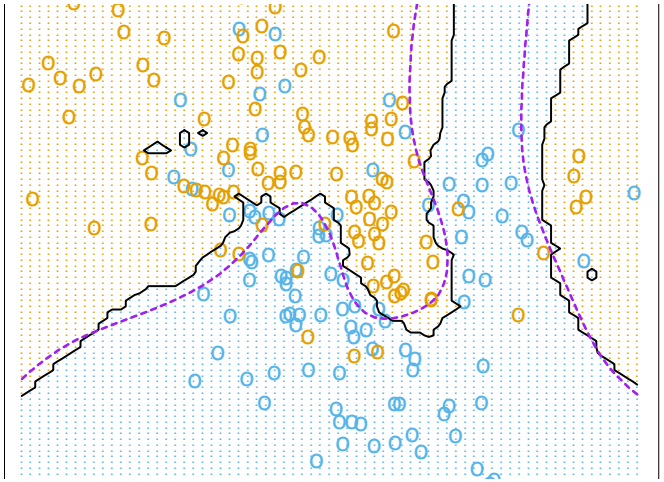
Course announcements

- Exam 1 is in two weeks: Thursday 10/19 at 7:30pm
  - Includes material up to and including today's lecture
  - Students who have a conflict must write to course staff by this Monday
- Project Milestone 1 due tonight
- Added two additional books to information sheet
  - *Understanding Machine Learning: From Theory to Algorithms*, Shalev-Shwartz and Ben-David, 2014.
  - *The Elements of Statistical Learning*, Hastie, Tibshirani, Friedman, 2009.
  - **Both available for free online**

Today's lecture:
Tale of two simple classifiers



Linear classifier



Nearest neighbor classifier

What are the statistical trade-offs of these two learning algorithms?

Today's lecture: Outline

- k-Nearest Neighbor (k-NN) algorithm
  - Algorithmic details
  - Example on MNIST (handwritten characters)
  - Discussion
- VC-dimension of classifiers
  - How much data do we need to *generalize* well?
  - Analysis for linear and 1-NN classifiers

Nearest Neighbor Algorithm

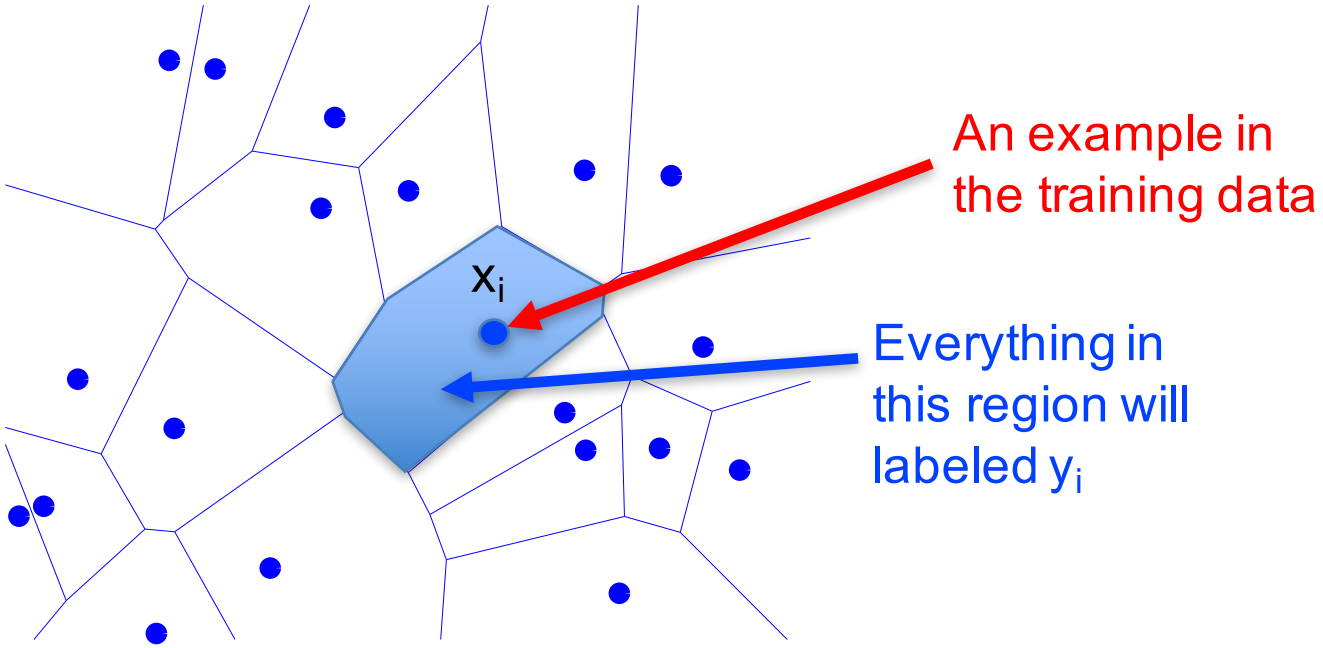
- Learning Algorithm:
  - Store training examples
- Prediction Algorithm:
  - To classify a new example **x** by finding the training example $(\mathbf{x}_i, y_i)$ that is *nearest* to **x**
  - Guess the class $y = y_i$

Must remember the whole dataset

(Naive implementation) Running time for prediction is O(N)

Example of a *non-parametric* method

- Decision boundary (called a "Voronoi diagram"):

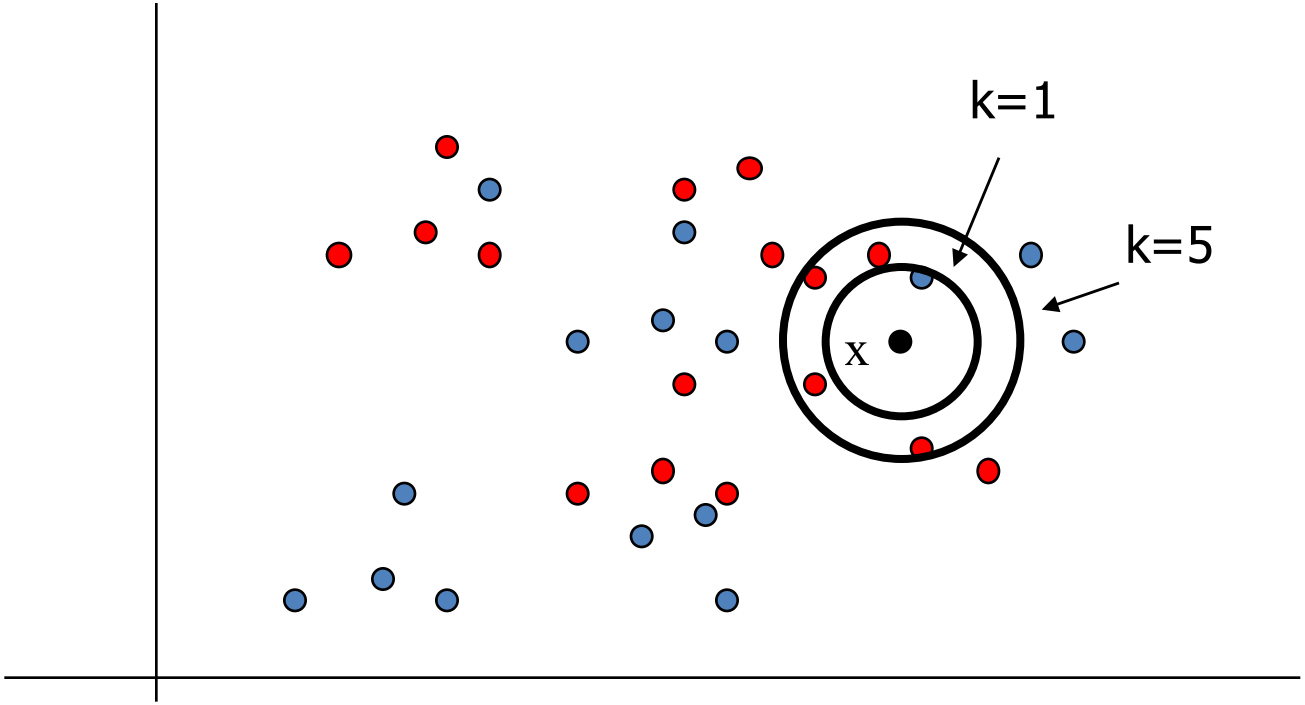


[Figure from Shalev-Shwartz & Ben-David, '14]

- The more examples that are stored, the more complex the decision boundaries can become

k-Nearest neighbor (*k*-NN) algorithm

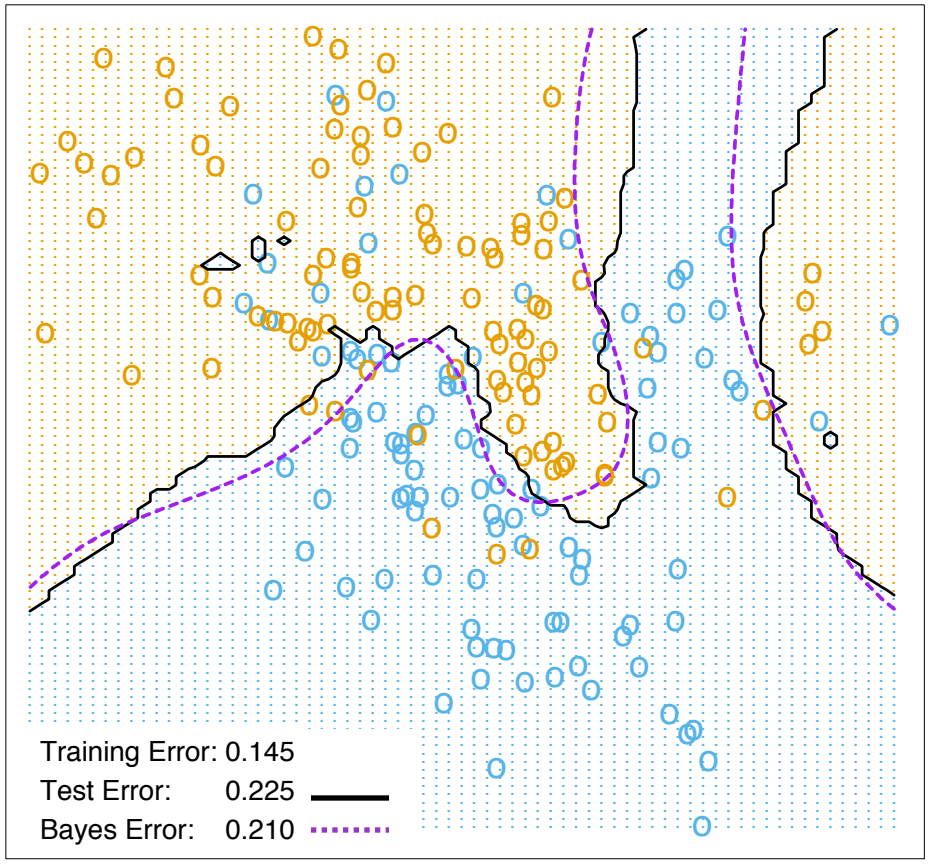
- To classify new input **x**, find *k* closest data points and assign the most frequently occurring class:



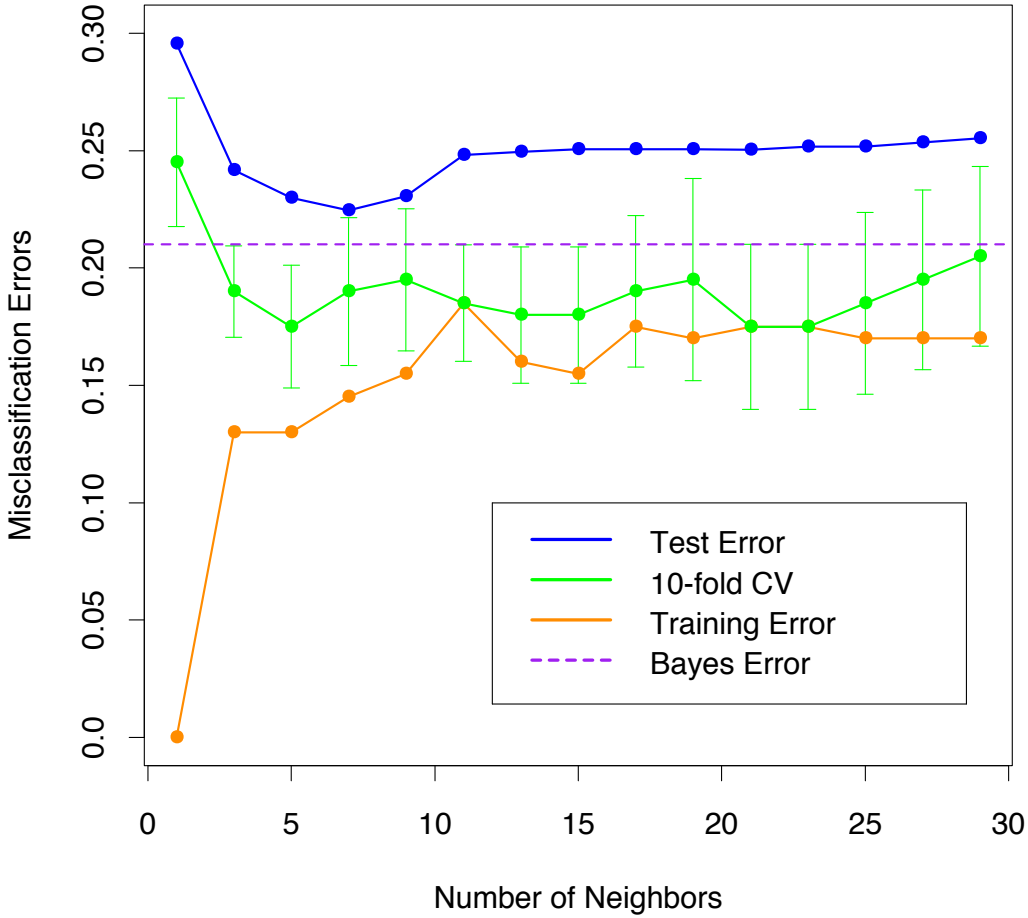
- Increasing *k* reduces variance, increases bias

Behavior of *k*-NN as *k* increases

7-Nearest Neighbors



[Hastie and Tibshirani, Chapter 13]



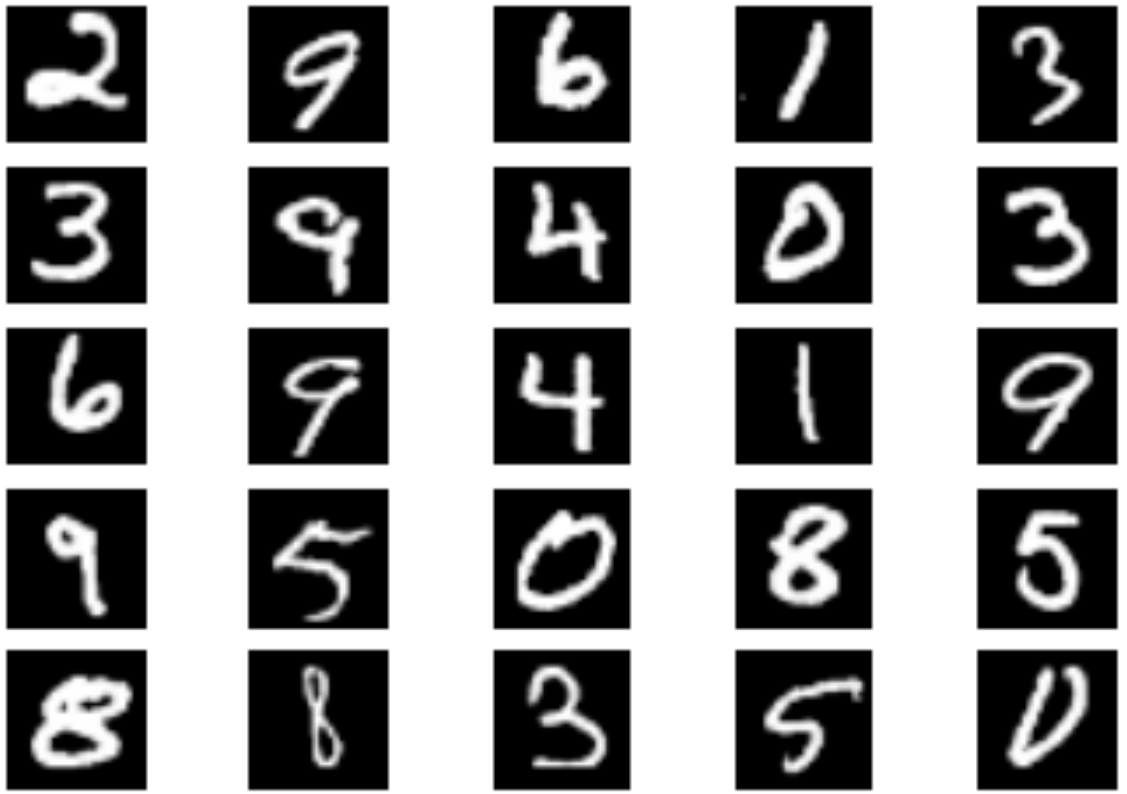
Application to handwritten character recognition

MNIST dataset

Goal: classify each handwritten character into {0, 1, 2, …, 9}

60K training points

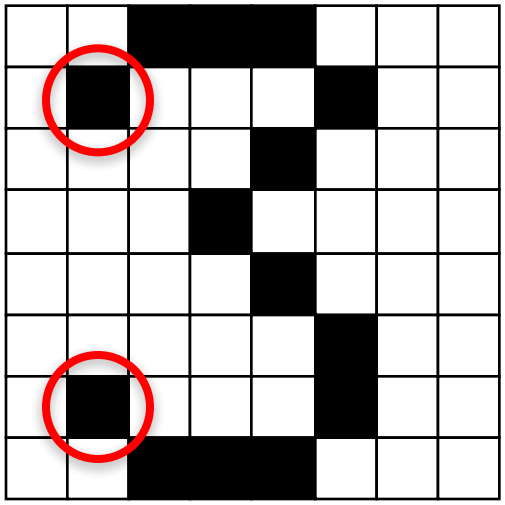
Random sample from MNIST



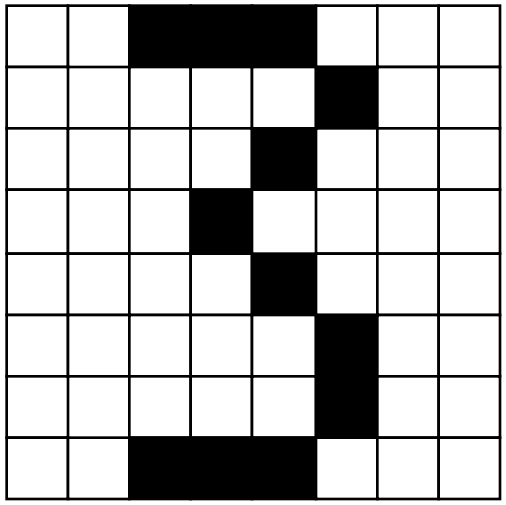
[LeCun et al., 1998]

Application to handwritten character recognition: **distance function**

How should we measure the distance between two images? (each image is 28 x 28 grayscale)



x_1



x_2

Euclidean distance on 784-dimensional vector:

$$d(x_1, x_2) = ||x_1 - x_2||_2 = \sqrt{2}$$

Application to handwritten character recognition: **confusion matrix on test data (1-NN)**

Predicted label

True label

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	972	1	1	0	0	1	3	1	0	0
1	0	1129	3	0	1	1	1	0	0	0
2	7	6	992	5	1	0	2	16	3	0
3	0	1	2	970	1	19	0	7	7	3
4	0	7	0	0	944	0	3	5	1	22
5	1	1	0	12	2	860	5	1	6	4
6	4	2	0	0	3	5	944	0	0	0
7	0	14	6	2	4	0	0	992	0	10
8	6	1	3	14	5	13	3	4	920	5
9	2	5	1	6	10	5	1	11	1	967

https://www.linkedin.com/pulse/nearest-neighbour-classification-techniques-mnist-indurthi/

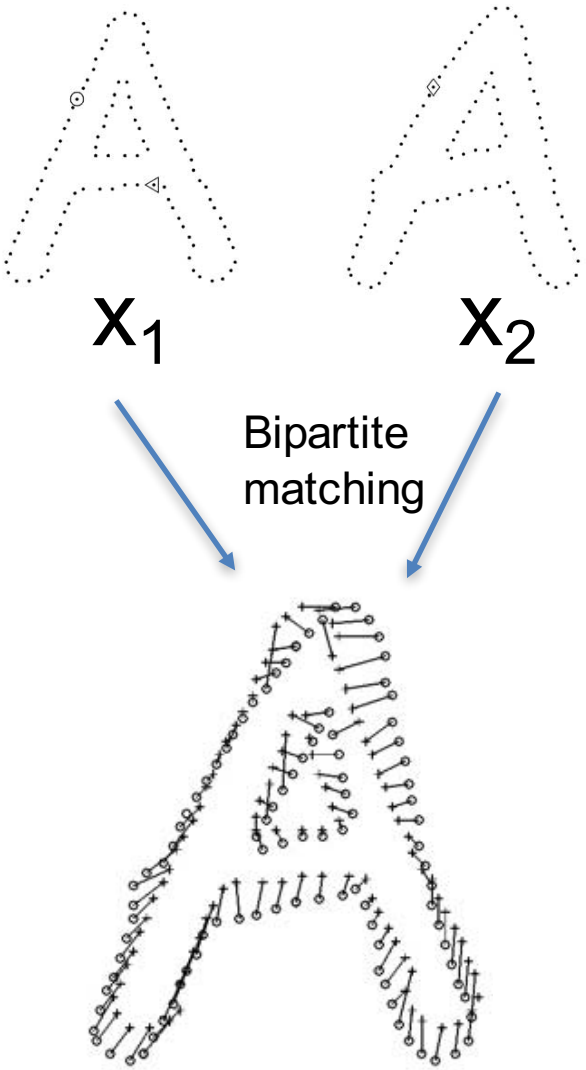
Application to handwritten character recognition: **can we design a better distance function?**

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 24, APRIL 2002 509

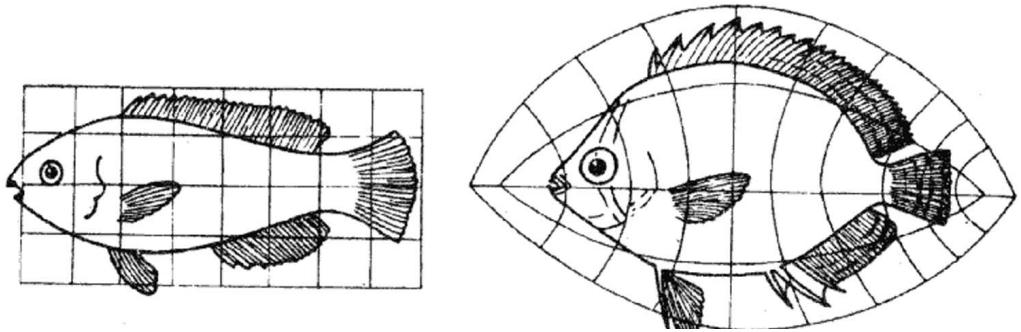
Shape Matching and Object Recognition Using Shape Contexts

Serge Belongie, *Member*, *IEEE*, Jitendra Malik, *Member*, *IEEE*, and Jan Puzicha

Abstract—We present a novel approach to measuring similarity between shapes and exploit it for object recognition. In our framework, the measurement of similarity is preceded by 1) solving for correspondences between points on the two shapes, 2) using the correspondences to estimate an aligning transform. In order to solve the correspondence problem, we attach a descriptor, the *shape context*, to each point. The shape context at a reference point captures the distribution of the remaining points relative to it, thus offering a globally discriminative characterization. Corresponding points on two similar shapes will have similar shape contexts, enabling us to solve for correspondences as an optimal assignment problem. Given the point correspondences, we estimate the transformation that best aligns the two shapes; regularized thin-plate splines provide a flexible class of transformation maps for this purpose. The dissimilarity between the two shapes is computed as a sum of matching errors between corresponding points, together with a term measuring the magnitude of the aligning transform. We treat recognition in a nearest-neighbor classification framework as the problem of finding the stored prototype shape that is maximally similar to that in the image. Results are presented for silhouettes, trademarks, handwritten digits, and the COIL data set.

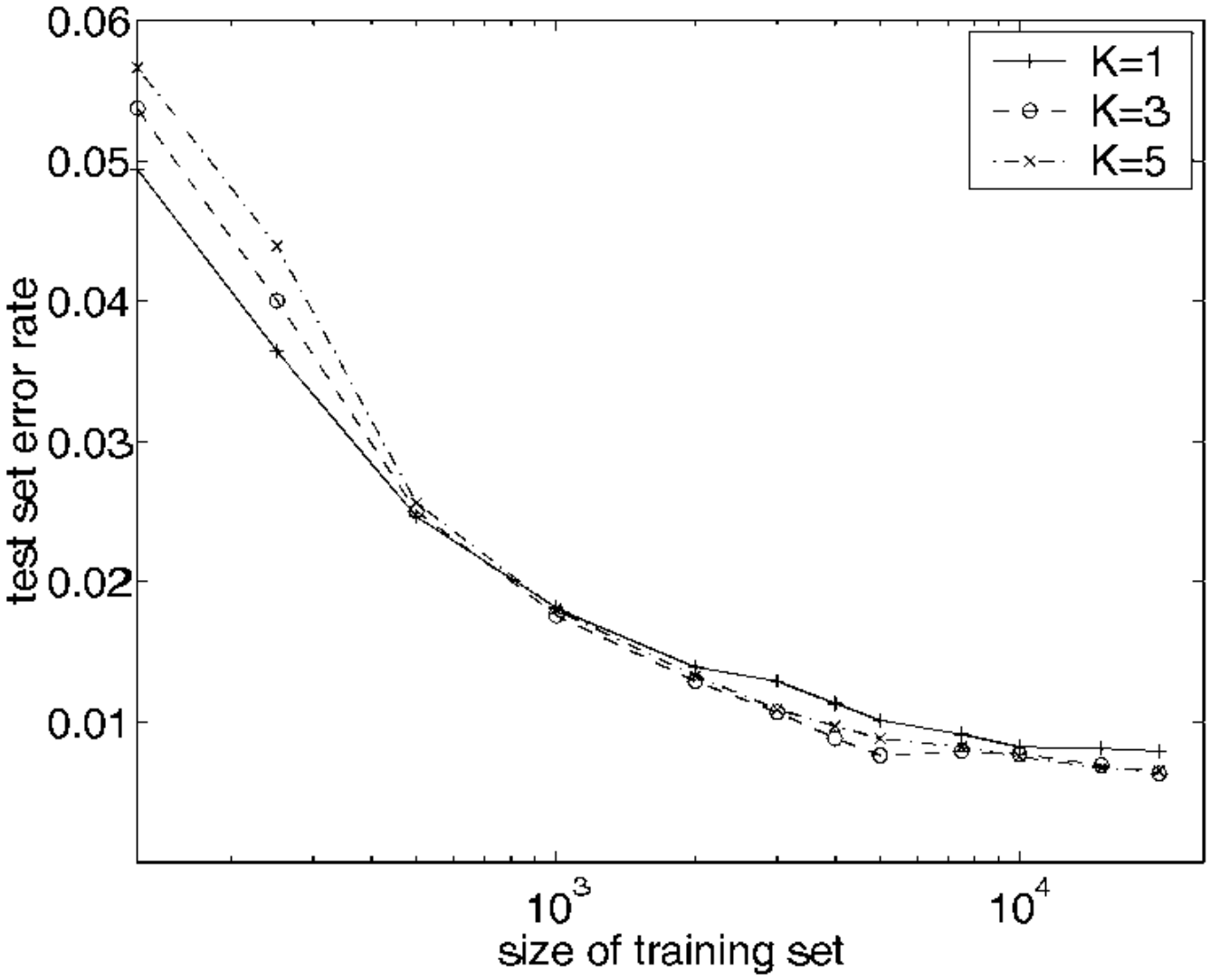


Intuition from a classic work from 1917:



D'Arcy Thompson's *On Growth and Form*

Application to handwritten character recognition: **results on MNIST with shape distance**



3-NN with Euclidean distance:
2.9% error

3-NN with shape distance:
0.63% error

Convolutional neural network:
0.35% error

When to use nearest neighbor algorithms

When to Consider

- Have a good feature extractor / distance function
- Lots of training data

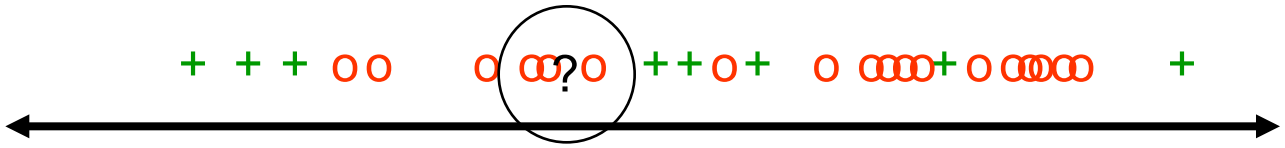
Advantages

- Training is very fast
- Learn complex target functions
- *Interpretability*

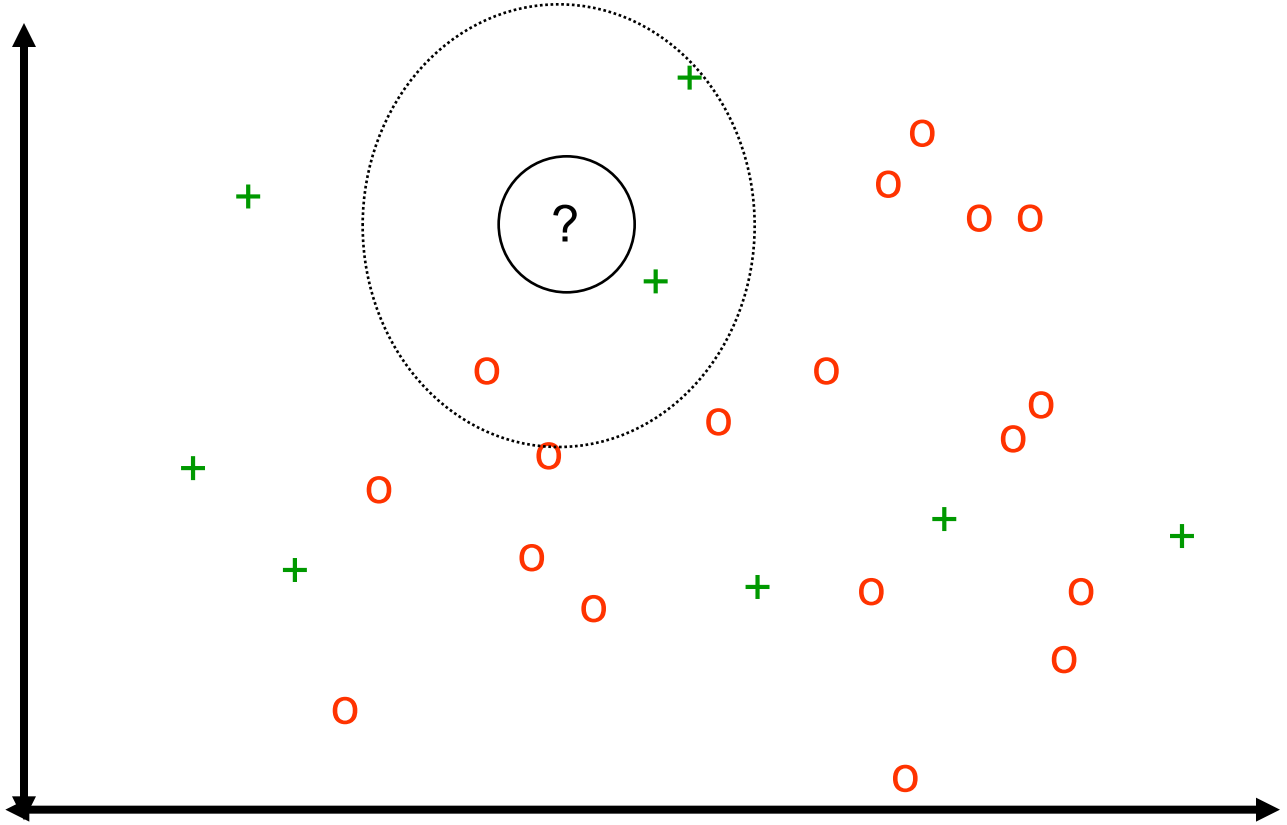
Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

k-NN and irrelevant features



k-NN and irrelevant features



Q: What would a linear classifier do in this setting?
A: It would put zero weight on the irrelevant feature

Q: How can we get the best of both worlds?
A: *Learn* the distance function

Relationship of *k*-NN to SVM with RBF kernel

- Consider the following generalization of the *k*-NN algorithm:

$$\hat{y}(\vec{x}) \leftarrow \text{sign}\left(\sum_{i=1}^{N} y_i d(\vec{x}_i, \vec{x})\right) \quad \text{with} \quad d(\vec{x}_i, \vec{x}) = \exp\left(-\frac{||\vec{x}_i - \vec{x}||_2^2}{2\sigma^2}\right)$$

- Looks at all training points (i.e., *k*=N), but weighs *i*'th training point depending on distance of $\mathbf{x}_i$ from $\mathbf{x}$

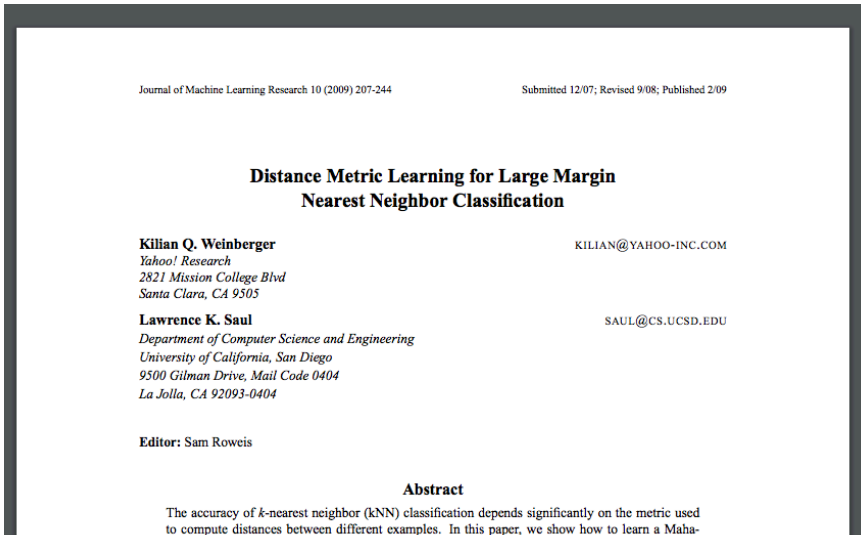
- Compare to classification with SVM and RBF kernel (using the dual):

$$\hat{y}(\vec{x}) \leftarrow \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x}_i, \vec{x})\right) \quad \text{with} \quad K(\vec{u}, \vec{v}) = \exp\left(-\frac{||\vec{u} - \vec{v}||_2^2}{2\sigma^2}\right) \qquad 0 \leq \alpha_i \leq C$$

- The functions are identical, except that the SVM has additional parameters α_i that can be learned

Discussion about nearest neighbor methods

- In high-dimensional spaces, Euclidean distance may not be very informative



Journal of Machine Learning Research 10 (2009) 207-244 Submitted 12/07; Revised 9/08; Published 2/09

Distance Metric Learning for Large Margin Nearest Neighbor Classification

Kilian Q. Weinberger KILIAN@YAHOO-INC.COM
Yahoo! Research
2821 Mission College Blvd
Santa Clara, CA 9505

Lawrence K. Saul SAUL@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, Mail Code 0404
La Jolla, CA 92093-0404

Editor: Sam Roweis

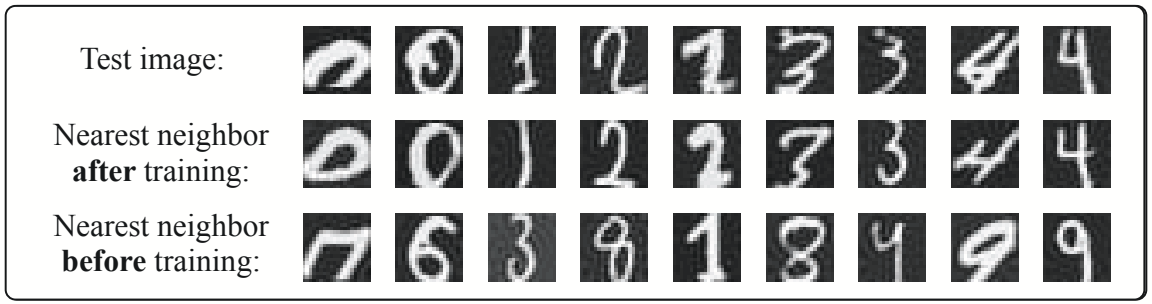
Abstract

The accuracy of *k*-nearest neighbor (kNN) classification depends significantly on the metric used to compute distances between different examples. In this paper, we show how to learn a Maha-

Key idea:
Compute Euclidean distance after performing a *linear transformation* of the feature vector

Called **Mahalanobis distance metric**:

$$\mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j)$$



Discussion about nearest neighbor methods

- In high-dimensional spaces, Euclidean distance may not be very informative
- Motivates fast algorithms for (approximate) nearest neighbor computation

1. Locality sensitive hashing
   Collection of special hash functions $h_1, \ldots, h_m : \mathcal{X} \to \mathbb{Z}$.
   Search for nearest neighbor in

$$\bigcup_{i=1}^{m} \{x \in S : h_i(x) = h_i(q)\}$$

   This set is smaller than S, and is likely to contain the nearest neighbor of q.
2. Tree-based search
   Build tree structure on S, and use it to discard subsets of S that are far from a query q.
   Common options: k-d tree, PCA tree, cover tree.

Discussion about nearest neighbor methods

- In high-dimensional spaces, Euclidean distance may not be very informative
- Motivates fast algorithms for (approximate) nearest neighbor computation
- Powerful concept that shows up elsewhere in ML, e.g. one-shot learning

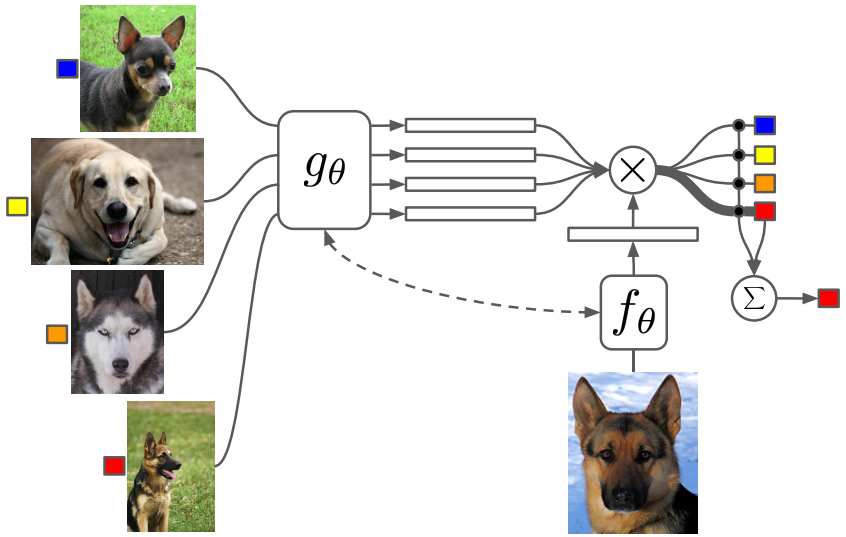


Figure 1: Matching Networks architecture

[Vinyals et al., 2016]

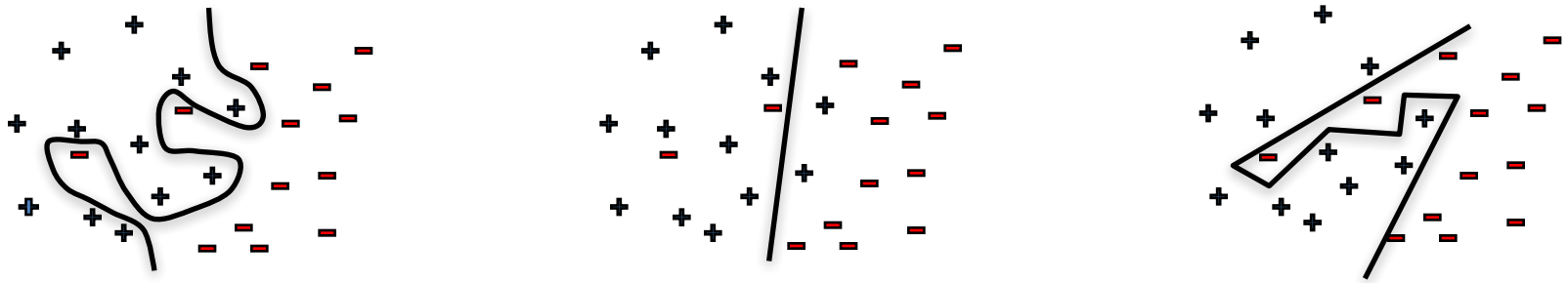
Today's lecture: Outline

- k-Nearest Neighbor (k-NN) algorithm
  - Algorithmic details
  - Example on MNIST (handwritten characters)
  - Discussion

- **VC-dimension of classifiers**
  - How much data do we need to *generalize* well?
  - Analysis for linear and 1-NN classifiers

Warmup: finite hypothesis classes

- Suppose that have a small set of classifiers *H* that you have to choose from, e.g. these $|H|=3$:



- Empirical risk minimization: *select the one with smallest error on our dataset*
- How well does it perform on held-out data?

Application: how big should your validation set be?

- You go off to industry, eager to apply your newly attained ML chops
- In comes a new dataset. You run SVM with different regularization constants, choice of kernels, … each run on the training set gives you a different classifier. Say **|H|=**40
- You know training error is not reflective of test error, so you use a held-out validation set of **m** data points to choose among these 40 classifiers
- How large does **m** need to be for us to be confident that the empirical risk is reflective of the population risk?

Generalization bound for |H| hypothesis

Theorem: Hypothesis space *H* finite, dataset *D* with *m* i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis *h*:

$$\Pr(\text{error}_{true}(h) - \text{error}_D(h) > \epsilon) \leq |H|e^{-2m\epsilon^2}$$

Proof sketch: Apply Hoefding's inequality to each individual hypothesis, and then apply the Union bound

PAC bound and Bias-Variance tradeoff

for all h, with probability at least 1-δ:

$$\text{error}_{true}(h) \leq \underbrace{\text{error}_D(h)}_{\text{“bias”}} + \underbrace{\sqrt{\frac{\ln |H| + \ln \frac{1}{\delta}}{2m}}}_{\text{“variance”}}$$

- For large |H|
  - low bias (assuming we can find a good h)
  - high variance (because bound is looser)
- For small |H|
  - high bias (is there a good h?)
  - low variance (tighter bound)

What about continuous hypothesis spaces?

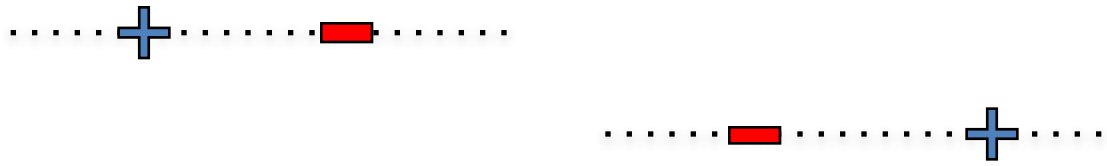
$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln |H| + \ln \frac{1}{\delta}}{2m}}$$

- Continuous hypothesis space:
  - $|H| = \infty$
  - Infinite variance???

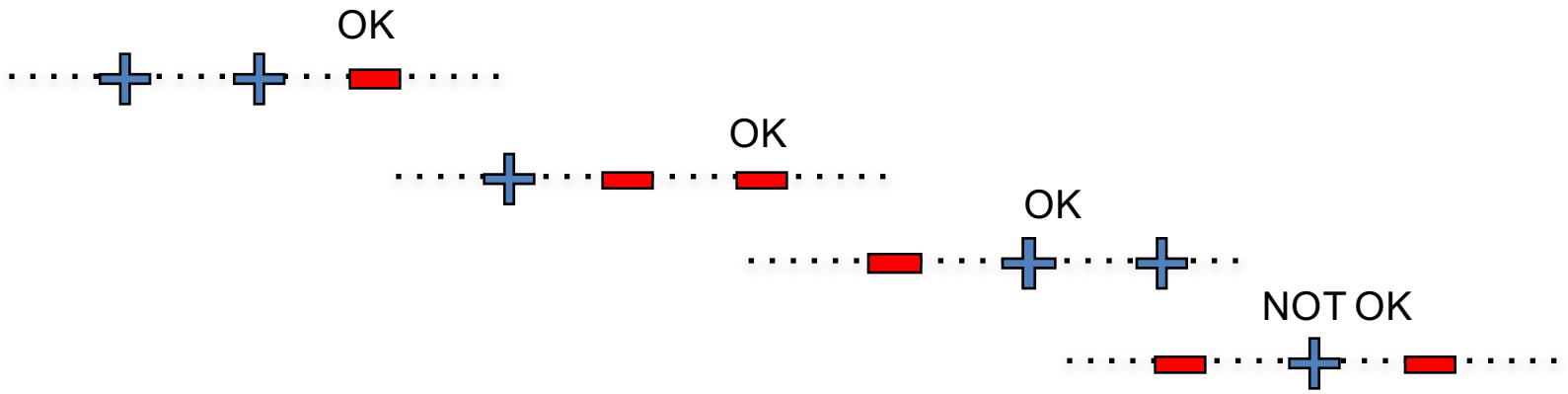
- **Only care about the maximum number of points that can be labeled any possible way!**

How many points can a linear boundary classify exactly? (*d*=1)

2 Points: Yes!!



3 Points: No…



etc (8 total)

Shattering and Vapnik–Chervonenkis Dimension

A **set of points** is *shattered* by a hypothesis space H iff:

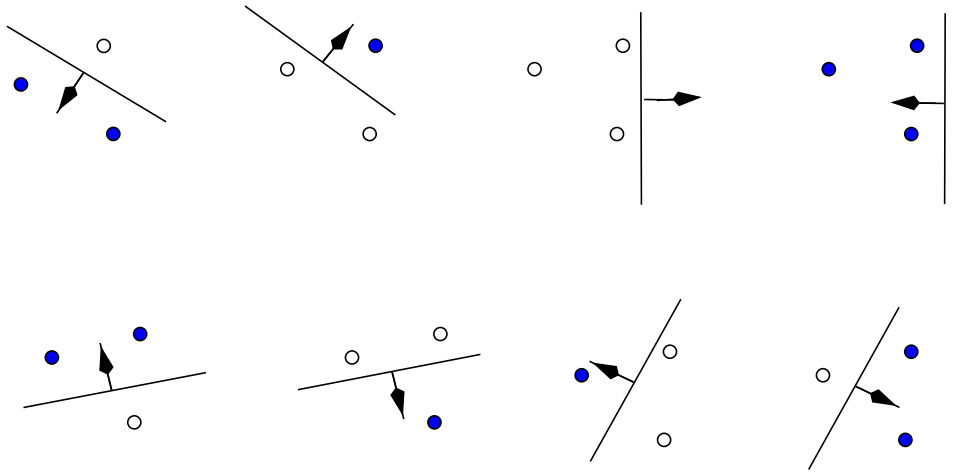
- For all ways of *splitting* the examples into positive and negative subsets
- There exists some *consistent* hypothesis h

The *VC-dimension* of H over input space

- The size of the *largest* number of points (we get to choose!) that can be shattered by H

How many points can a linear boundary classify exactly? (*d*=2)

3 Points: Yes!!



4 Points: No…

etc.

[Figure from Chris Burges]

How many points can a linear boundary classify exactly? (in *d* dimensions)

- A linear classifier $\sum_{j=1..d} w_j x_j + b$ can represent all assignments of possible labels to d+1 points
  - But not d+2!
  - Bias term b required
- Thus, VC-dimension of d-dimensional linear classifiers is d+1

- Question: Can we get a bound for error as a function of the VC-dimension?

PAC bound using VC dimension

- VC dimension: number of training points that can be classified exactly (shattered) by hypothesis space H
  - Measures relevant size of hypothesis space

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{VC(H)\left(\ln \frac{2m}{VC(H)} + 1\right) + \ln \frac{4}{\delta}}{m}}$$

- Same bias / variance tradeoff as always
  - Now, just a function of VC(H)

VC-dimension of nearest-neighbor classifiers

- In contrast, the VC-dimension of a 1-NN classifier is infinite!

- As a consequence of what you show in Exercise 4 (question 3), hard-margin SVM with RBF kernel also has infinite VC-dimension

Discussion

- We are guaranteed that linear classifiers (of not too high dimension) *generalize well*, regardless of regularization
- Suggestive of why L1 regularization of linear classifiers can be a good idea
- Analyzing *gap-tolerant* classifiers provides theory to explain why L2 regularization (large margin) is a good idea
- What we described is for binary classification
  - Can be generalized to multi-class and also regression
  - More convenient complexity measures such as Rademacher complexity