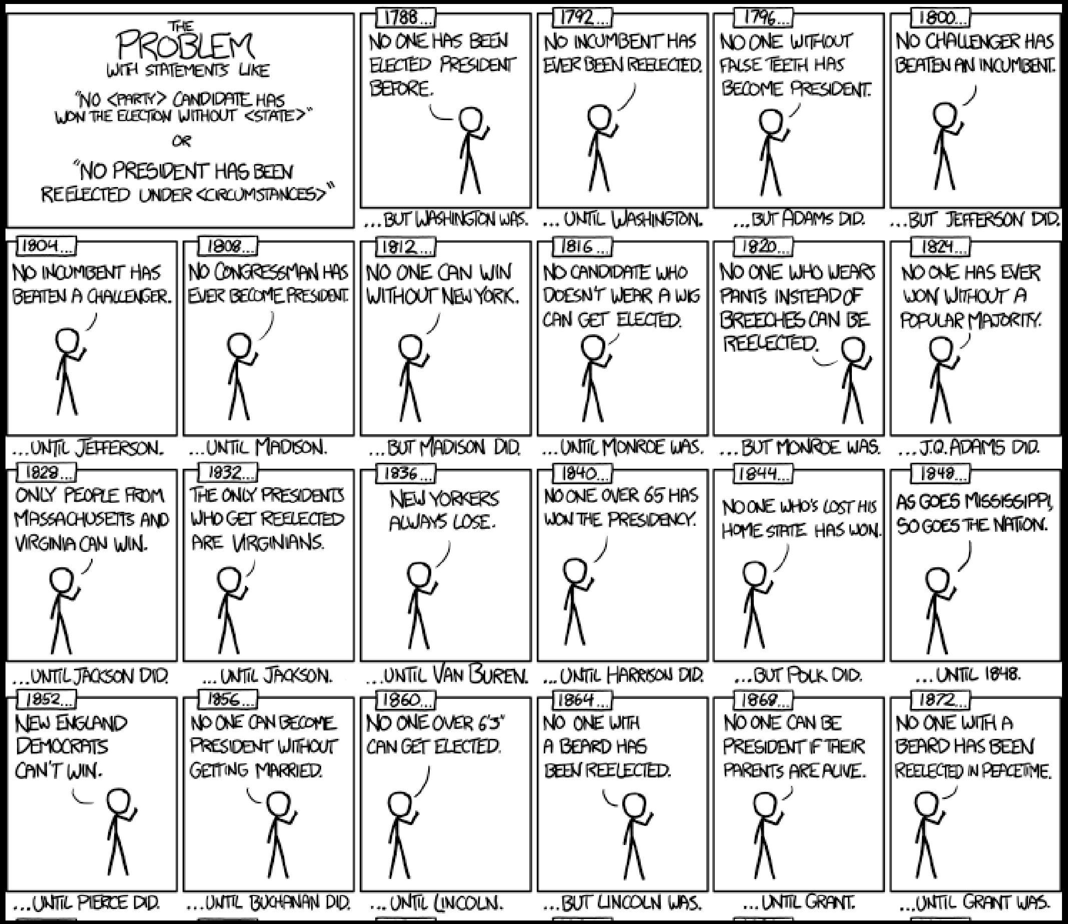
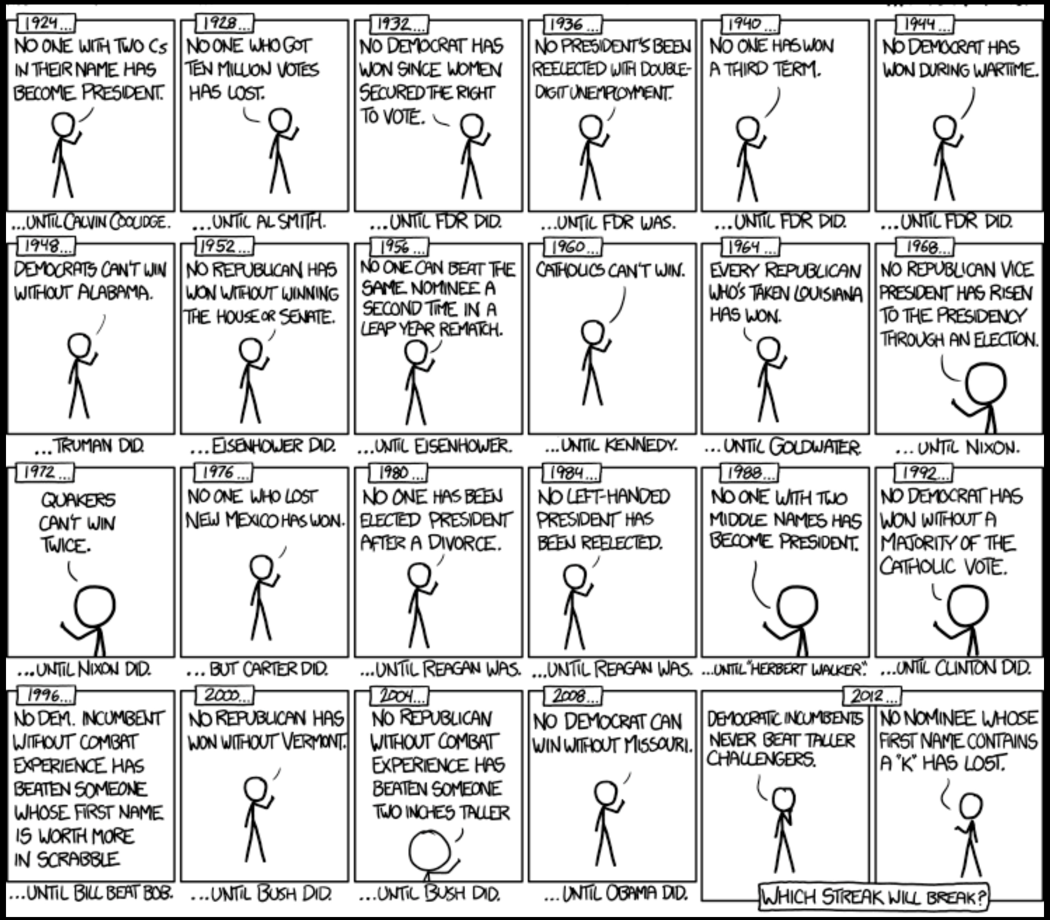
Overfitting, xkcd style





INFO 251: Applied Machine Learning

Nearest Neighbors

Announcements

- PS3 posted, due Feb 20
- Reminder: Please use Ed for all course-related communication, unless it's truly impossible that someone else in the class might have the same question
  - If the question is truly unique to your circumstances, then include me + Suraj + Satej in your communications

Course Outline

- Causal Inference and Research Design
  - Experimental methods
  - Non-experiment methods
- **Machine Learning**
  - **Design of Machine Learning Experiments**
  - Linear Models and Gradient Descent
  - Non-linear models
  - Fairness and Bias in ML
  - Neural models
  - Deep Learning
  - Practicalities
  - Unsupervised Learning
- Special topics

Key Concepts (last lecture)

- Representation
- Evaluation
- Optimization
- Supervised Learning
- Unsupervised Learning
- The curse of dimensionality
- Feature engineering
- Overfitting
- Generalization
- Cross-validation
- Bootstrap
- Accuracy, ROC, AUC, F-scores
- Baselines
- Error analysis
- Ablative analysis

Key Concepts (todays' lecture)

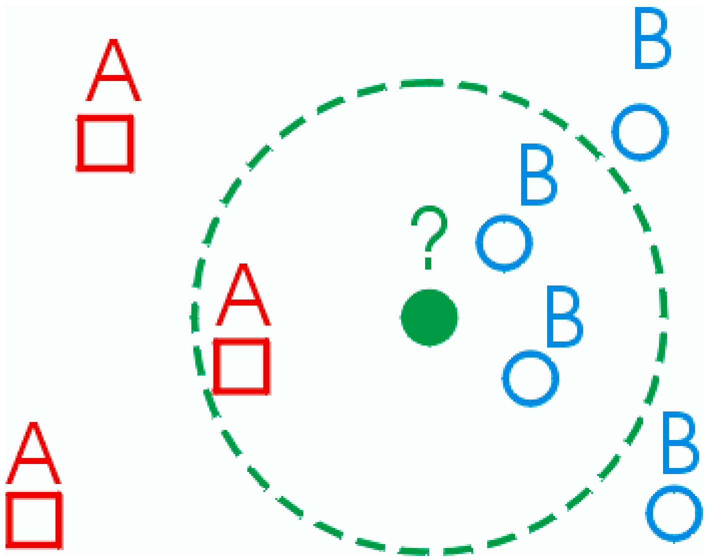
- Lazy learning
- Decision boundaries
- Voronoi diagrams
- (K-)Nearest Neighbors
- Similarity and Distance metrics
- Normalization and Standardization
- Feature weighting

Outline

- **Lazy learning**
- K-nearest neighbors
- Similarity and Distance metrics
- Curse of Dimensionality
- Case Study: Digit classification

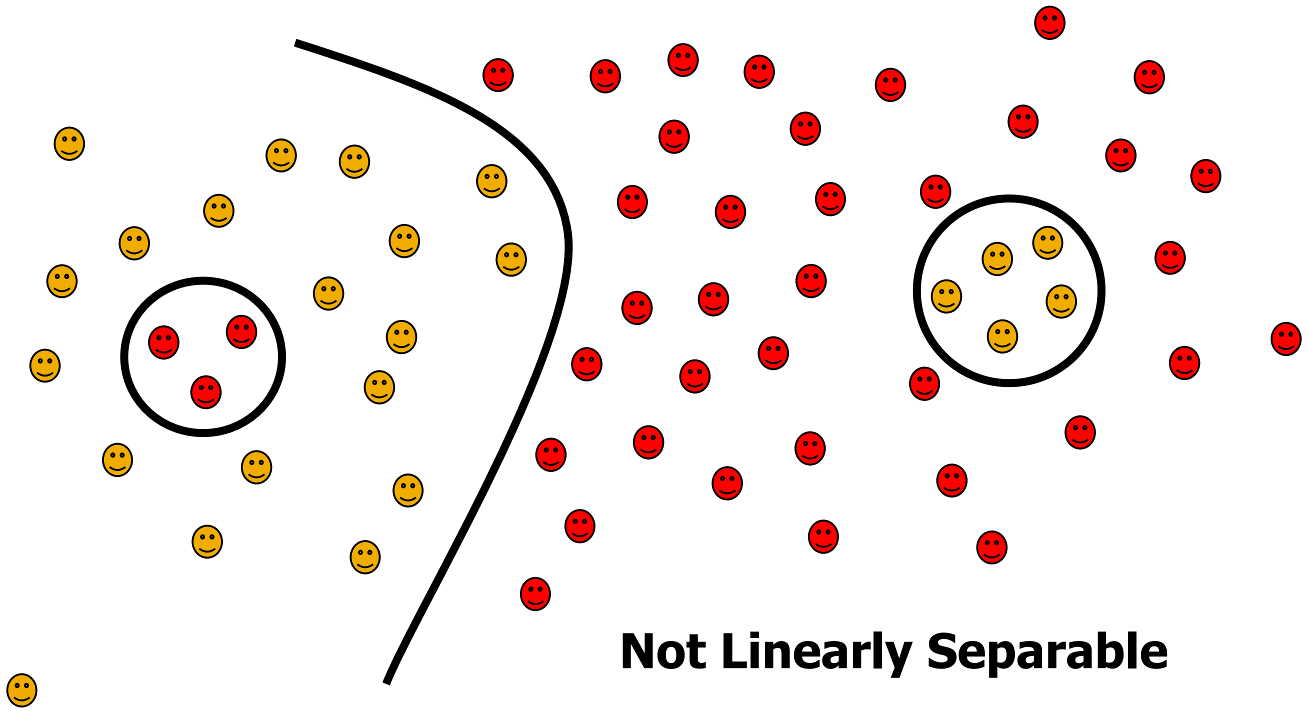
Instance-Based Learning

- "Lazy Learning"
  - Learn as little up front, make real-time decisions
- Nearest Neighbor (invented in 1950's!)
  - Given new datapoint x_i, find 'nearest neighbor' x_j to x_i, predict $f(x_i) = f(x_j)$



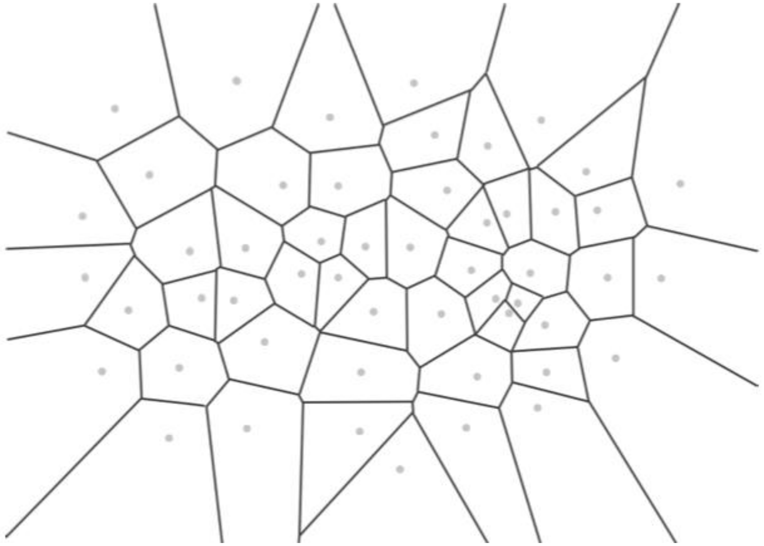
Why this approach?

- Can learn complex decision boundaries
  - Including data that is not linearly separable



Voronoi diagrams

- How does Nearest Neighbors divide hypothesis space?
  - A simple idea that induces a complex function
- Regions mark the space closest to each point



Outline

- Lazy learning
- **K-nearest neighbors**
- Similarity and Distance metrics
- Curse of Dimensionality
- Case Study: Digit classification

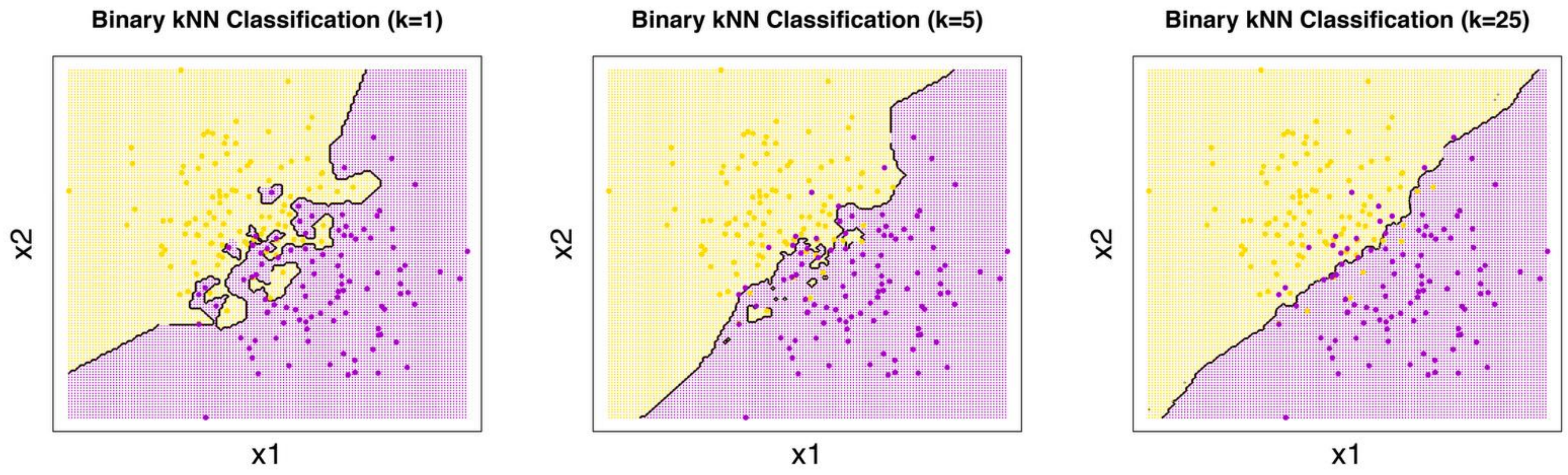
K-Nearest Neighbors (kNN)

- Nearest Neighbors is very unstable
  - Beware of overfitting!
- *K*-NN: Generalized version of NN
  - Given x_i, take vote among *K* nearest neighbors
- If output is discrete?
  - Majority prediction wins
- If continuous?
  - Take mean of *K* nearest neighbors

$$f(x_i) = \frac{1}{K}\sum_{j=1}^{K} f(x_j)$$

Decision boundaries

- *K*-NN has complex boundaries
- larger *K* smoothes the boundary
- How to determine *K*?
  - Use cross-validation!



K-NN example

- Predicting Default

Training data →

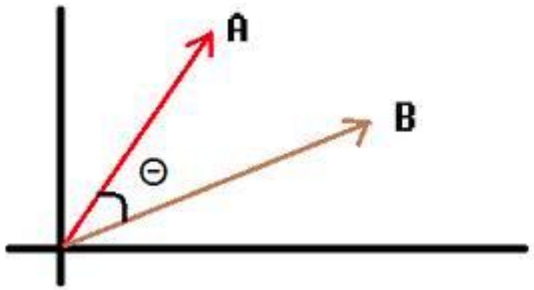
Age	Loan	Default
25	$40,000	N
35	$60,000	N
45	$80,000	N
20	$20,000	N
35	$120,000	N
40	$62,000	Y
60	$100,000	Y
48	$220,000	Y
33	$150,000	Y

- Test data:
  - Age=31
  - Loan=125,000
- Need to define distance
  - Closest by age?
  - Closest by loan amount?

Distance metrics

- For numeric features:
  - Euclidean and Manhattan Distance
  - L^n-Norm: $D^n(x_i, x_j) = \sqrt[n]{\sum_{m=1}^{M} |x_{im} - x_{jm}|^n}$
  - L^∞-Norm (Chebyshev)
- Cosine similarity:

$$\frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$



Distance metrics

- For symbolic features:
  - Hamming distance

Hamming distance = 3

A	1	0	1	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0	0	1	1

Hamming distance = 6

GAGCCTACTAACGGGAT
CATCGTAATGACGGCCT

    - Jaccard Similarity
    - Value distance measure, etc.
    - Mutual information, etc.

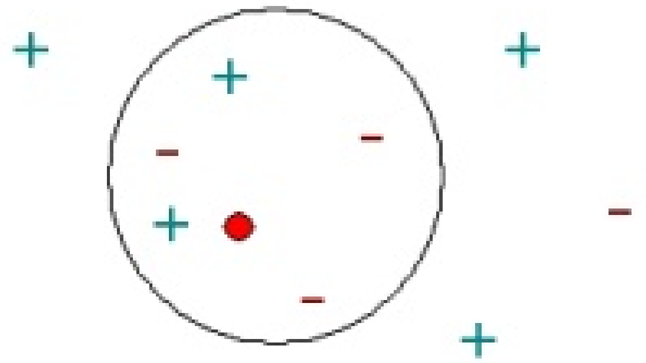
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

K-NN: Pros and Cons

- Advantages
  - Can learn complex functions
  - Training is very fast
  - No loss of information
- Disadvantages
  - Slow at query time
  - Storage requirements
  - Easy to fool

Weighted *K*-NN

- Why weight neighbors evenly?
- Distance-weighted *K*-Nearest Neighbors
  - $f(x_i) = \frac{\sum_{j=1}^{k} w_{ij} f(x_j)}{\sum_{j=1}^{k} w_{ij}}$
- Where
  - $w_{ij} = \frac{1}{d(x_i, x_j)} = \frac{1}{\sqrt[n]{\sum_{m=1}^{M} |x_{im} - x_{jm}|^n}}$
- In theory, can use all training examples
  - But in practice, we might not…

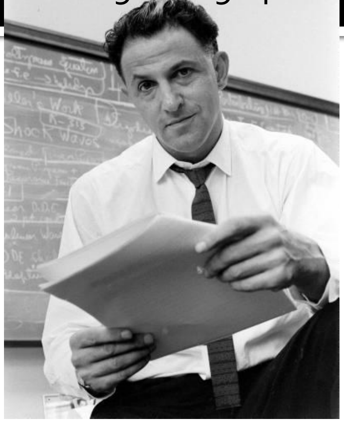


Outline

- Lazy learning
- K-nearest neighbors
- Similarity and Distance metrics
- **Curse of Dimensionality**
- Case Study: Digit classification

Curse of dimensionality

Richard Bellman
1920 - 1984



- Our brains get confused in high dimensions
  - Our intuitions are based on 2-D and 3-D spaces
  - "If we could see in high dimensions we wouldn't need ML"
- Adding dimensions increases space exponentially
  - 100 evenly spaced points on unit interval: $\bar{d} = 0.01$
  - In 10 dimensions, to have the same average distance between points, we would need 10,000,000,000,000,000,000 points!
  - 10 features isn't that many!
- Data in many dimensions is tricky
  - Poor sampling of the space
  - All points are far apart
  - Relative feature weights matter

Curse of dimensionality

- What can we do about it?
- **Normalize** numeric features
  - Standardized features: mean = 0, variance = 1
- Feature selection – we'll come back to this
  - A priori filtering – very "cheap" but sensitive
  - Forward selection – progressively add features
  - Backward selection – progressively remove features
- Dimensionality reduction
  - We'll come back to this as well

Feature weighting

- Keep features, but scale back influence
  - From before, weighted nearest neighbors starts with

$$f(x_i) = \frac{\sum_{j=1}^{k} w_{ij} f(x_j)}{\sum_{j=1}^{k} w_{ij}}$$

  - When we add feature weighting, each feature is assigned weight $\boldsymbol{\delta_m}$

$$w_{ij} = \frac{1}{D^n(x_i, x_j)} = \frac{1}{\sqrt[n]{\sum_{m=1}^{M} \boldsymbol{\delta_m} |x_{im} - x_{jm}|^n}}$$

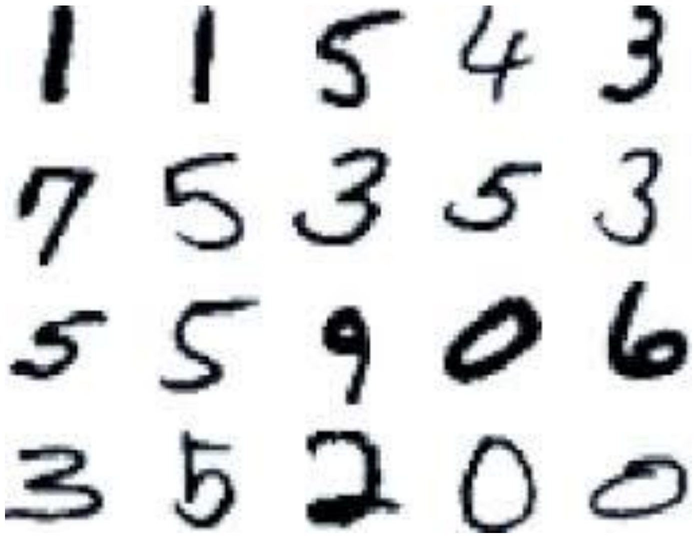
  - Setting $\boldsymbol{\delta_m}$ to zero eliminates that dimension
- How do we determine the $\boldsymbol{\delta_m}$?
  - Cross-validation (+ gradient descent)!

Outline

- Lazy learning
- K-nearest neighbors
- Similarity and Distance metrics
- Curse of Dimensionality
- **Case Study: Digit classification**

Case study: digit classification

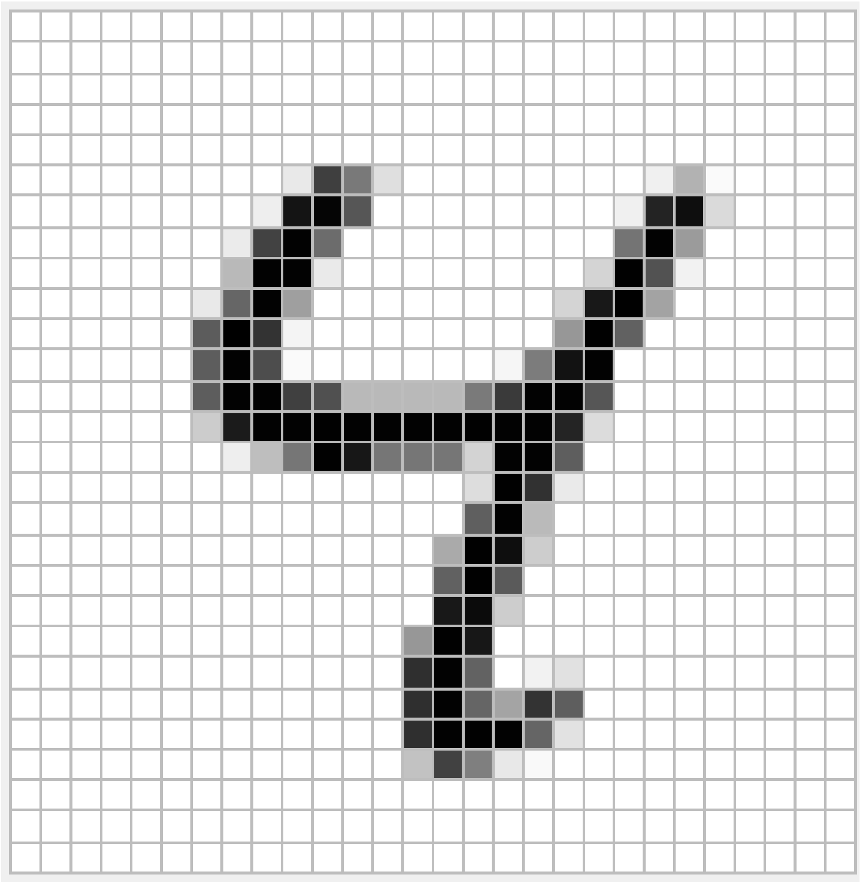
- MNIST data set
  - 70,000 labeled digits
  - 28 X 28 pixels each
  - Greyscale values (0-255)
  - Scaled and centered, but with plenty of variation



Feature representation

- What is a feature? i.e., what does a single x_i look like?
  - 28x28 grid of values
  - feature vector of length 784
  - Normalized to [0,1] scale
  - Note: pixel representation throws away locality – permutations are identical!

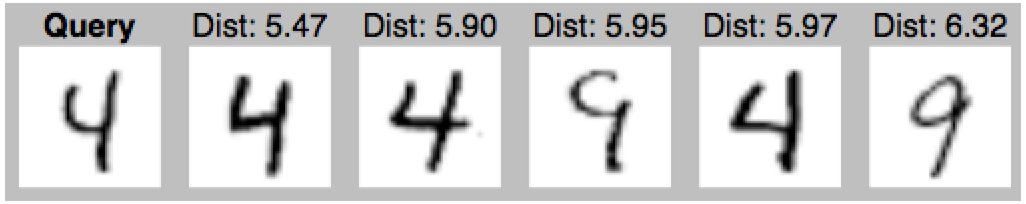
- What does the digit "4" look like, in terms of our feature space?
  - Our feature space is large and inappropriate



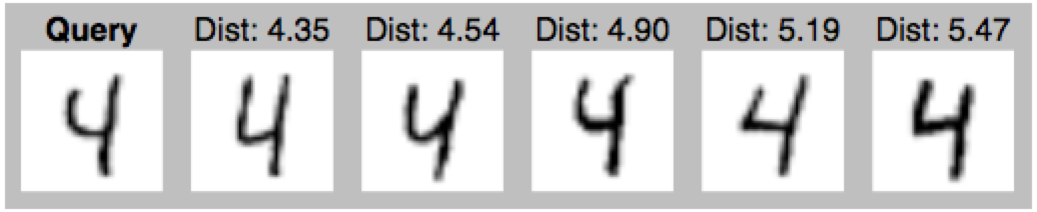
K-NN in practice

- Digit Neighbors

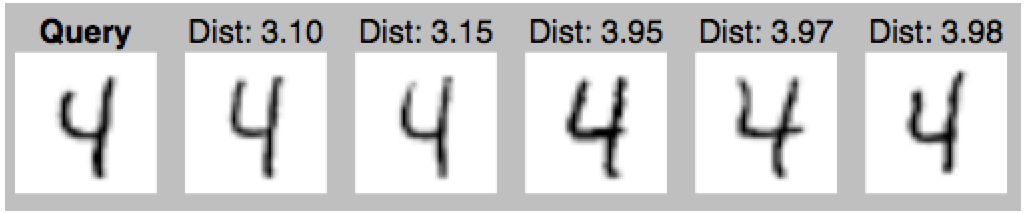
200 training examples



1000 training examples



10000 training examples



K-NN for digit classification

- Results

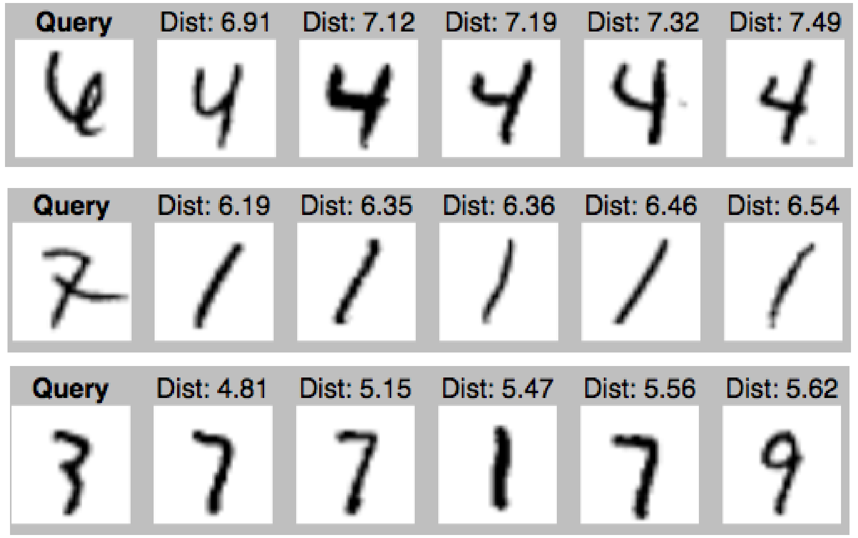
k=1; Euclidean (L2) distance

Training	Error%	Time
100	30.0	0.38
1000	12.1	2.34
10000	5.3	28.7
60000	2.7	2202

+ de-skewing	2.3
+ blurring	1.8
+ pixel-shifting	1.2

Edited *K*-NN

- Outliers (noise) may exist in the training set
  - Mislabeled examples
  - Unlearnable examples
- An easy solution: Remove outliers
  - If all neighbors are a different class



Instance-Based Learning

- Common forms of Instance-Based Learning
  - Lazy learning
  - ***K*-Nearest Neighbors**
  - **Locally-weighted regression**
  - Radial basis networks
  - Case-based reasoning
  - **Collaborative filtering**

For Next Class

- Read:
  - Daume, Chapter 7
  - Schutt & O'Neill, Chapter 5