CSC411/2515 Lecture 2: Nearest Neighbors

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

University of Toronto

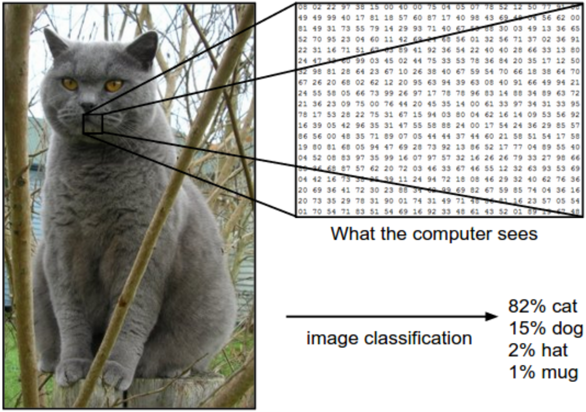
Introduction

- Today (and for the next 5 weeks) we're focused on supervised learning.
- This means we're given a training set consisting of inputs and corresponding labels, e.g.

Task	Inputs	Labels
object recognition	image	object category
image captioning	image	caption
document classification	text	document category
speech-to-text	audio waveform	text
⋮	⋮	⋮

Input Vectors

What an image looks like to the computer:



[Image credit: Andrej Karpathy]

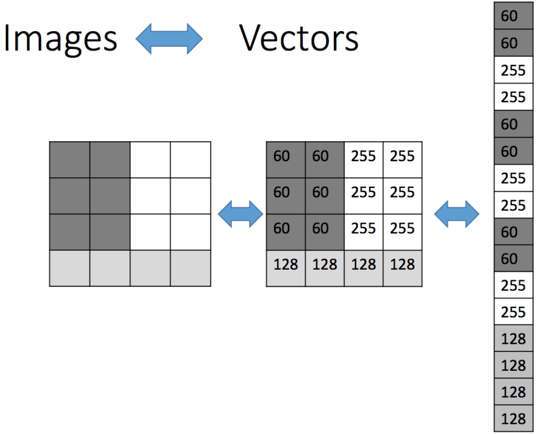
Input Vectors

- Machine learning algorithms need to handle lots of types of data: images, text, audio waveforms, credit card transactions, etc.
- Common strategy: represent the input as an input vector in $\mathbb{R}^d$
  - Representation = mapping to another space that's easy to manipulate
  - Vectors are a great representation since we can do linear algebra!

VECTOR INSTITUTE

Input Vectors

Can use raw pixels:



Can do much better if you compute a vector of meaningful features.

Input Vectors

- Mathematically, our training set consists of a collection of pairs of an input vector $\mathbf{x} \in \mathbb{R}^d$ and its corresponding target, or label, t
    - Regression: t is a real number (e.g. stock price)
    - Classification: t is an element of a discrete set $\{1, \ldots, C\}$
    - These days, t is often a highly structured object (e.g. image)
- Denote the training set $\{(\mathbf{x}^{(1)}, t^{(1)}), \ldots, (\mathbf{x}^{(N)}, t^{(N)})\}$
    - Note: these superscripts have nothing to do with exponentiation!

Nearest Neighbors

- Suppose we're given a novel input vector $\mathbf{x}$ we'd like to classify.
- The idea: find the nearest input vector to $\mathbf{x}$ in the training set and copy its label.
- Can formalize "nearest" in terms of Euclidean distance

$$||\mathbf{x}^{(a)} - \mathbf{x}^{(b)}||_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

Algorithm:

1. Find example $(\mathbf{x}^*, t^*)$ (from the stored training set) closest to $\mathbf{x}$. That is:

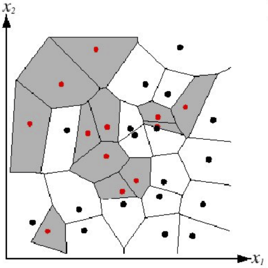
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\text{argmin}} \; \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

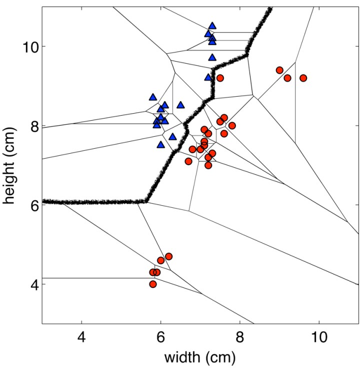
- Note: we don't need to compute the square root. Why?

Nearest Neighbors: Decision Boundaries

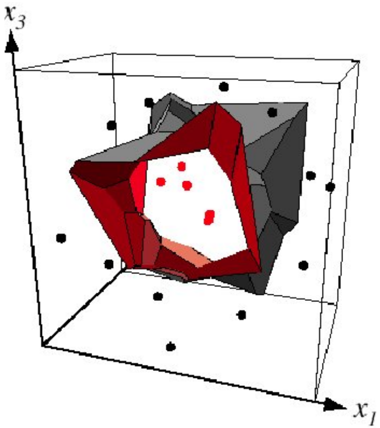
We can visualize the behavior in the classification setting using a Voronoi diagram.



Decision boundary: the boundary between regions of input space assigned to different categories.



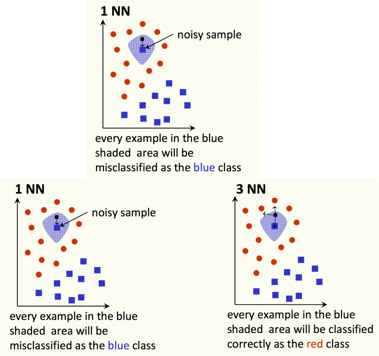
Nearest Neighbors: Decision Boundaries



Example: 3D decision boundary

k-Nearest Neighbors

[Pic by Olga Veksler]



- Nearest neighbors sensitive to noise or mis-labeled data ("class noise"). Solution?
- Smooth by having k nearest neighbors vote

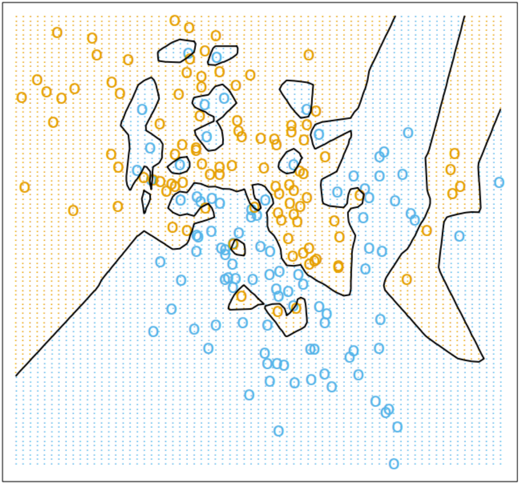
Algorithm (kNN):

1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

$$y = arg \max_{t^{(z)}} \sum_{r=1}^{k} \delta(t^{(z)}, t^{(r)})$$

K-Nearest neighbors

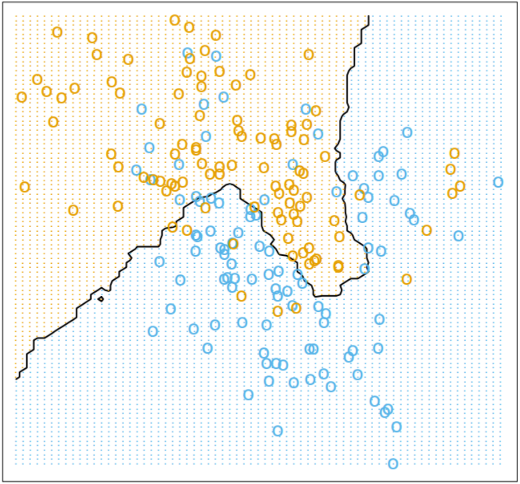
k=1



[Image credit: "The Elements of Statistical Learning"]

K-Nearest neighbors

k=15



[Image credit: "The Elements of Statistical Learning"]

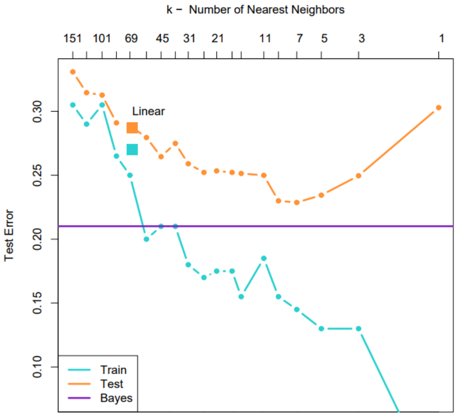
k-Nearest Neighbors

Tradeoffs in choosing k?

- Small k
  - Good at capturing fine-grained patterns
  - May overfit, i.e. be sensitive to random idiosyncrasies in the training data
- Large k
  - Makes stable predictions by averaging over lots of examples
  - May underfit, i.e. fail to capture important regularities
- Rule of thumb: $k < sqrt(n)$, where n is the number of training examples

K-Nearest neighbors

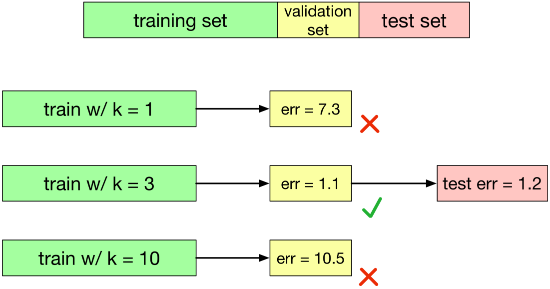
- We would like our algorithm to generalize to data it hasn't before.
- We can measure the generalization error (error rate on new examples) using a test set.



[Image credit: "The Elements of Statistical Learning"]

Validation and Test Sets

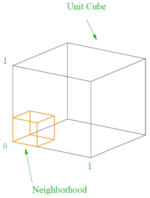
- k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself
- We can tune hyperparameters using a validation set:

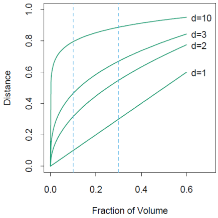


- The test set is used only at the very end, to measure the generalization performance of the final configuration.

Pitfalls: The Curse of Dimensionality

- Low-dimensional visualizations are misleading! In high dimensions, "most" points are far apart.
- If we want the nearest neighbor to be closer then ϵ, how many points do we need to guarantee it?
- The volume of a single ball of radius ϵ is $\mathcal{O}(\epsilon^d)$
- The total volume of $[0,1]^d$ is 1.
- Therefore $\mathcal{O}\left((\frac{1}{\epsilon})^d\right)$ balls are needed to cover the volume.

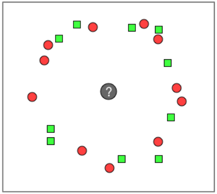




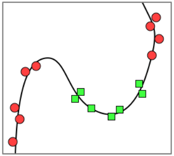
[Image credit: "The Elements of Statistical Learning"]

Pitfalls: The Curse of Dimensionality

- In high dimensions, “most” points are approximately the same distance. (Homework question coming up...)

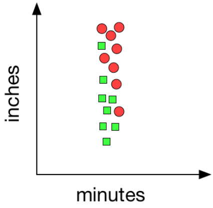


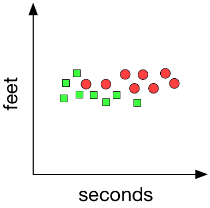
- Saving grace: some datasets (e.g. images) may have low intrinsic dimension, i.e. lie on or near a low-dimensional manifold. So nearest neighbors sometimes still works in high dimensions.



Pitfalls: Normalization

- Nearest neighbors can be sensitive to the ranges of different features.
- Often, the units are arbitrary:





- Simple fix: normalize each dimension to be zero mean and unit variance. I.e., compute the mean μ_j and standard deviation σ_j, and take

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j}$$

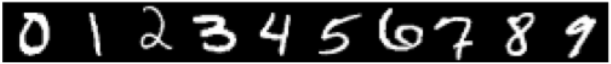
- Caution: depending on the problem, the scale might be important!

Pitfalls: Computational Cost

- Number of computations at training time: 0
- Number of computations at test time, per query (naïve algorithm)
  - Calculate D-dimensional Euclidean distances with N data points: $\mathcal{O}(ND)$
  - Sort the distances: $\mathcal{O}(N \log N)$
- This must be done for *each* query, which is very expensive by the standards of a learning algorithm!
- Need to store the entire dataset in memory!
- Tons of work has gone into algorithms and data structures for efficient nearest neighbors with high dimensions and/or large datasets.

Example: Digit Classification

- Decent performance when lots of data

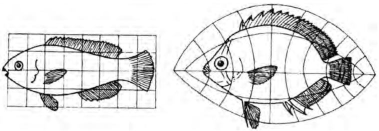


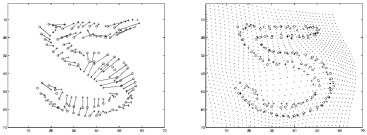
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: d = 784
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Example: Digit Classification

- KNN can perform a lot better with a good similarity measure.
- Example: shape contexts for object recognition. In order to achieve invariance to image transformations, they tried to warp one image to match the other image.
  - Distance measure: average distance between corresponding points on *warped* images
- Achieved 0.63% error on MNIST, compared with 3% for Euclidean KNN.
- Competitive with conv nets at the time, but required careful engineering.

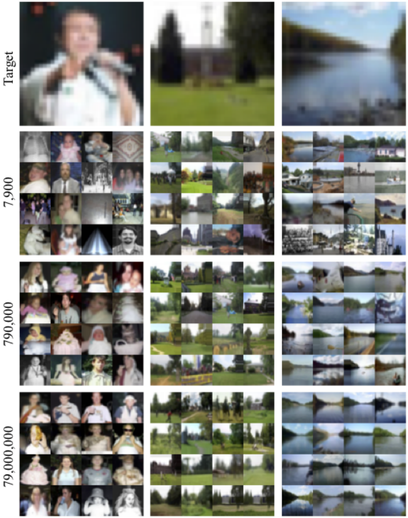




[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]

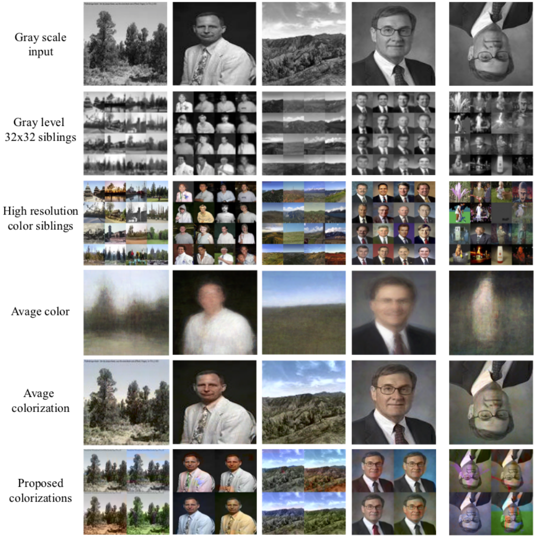
Example: 80 Million Tiny Images

- 80 Million Tiny Images was the first extremely large image dataset. It consisted of color images scaled down to 32×32.
- With a large dataset, you can find much better semantic matches, and KNN can do some surprising things.
- Note: this required a carefully chosen similarity metric.



[Torralba, Fergus, and Freeman, 2007. 80 Million Tiny Images.]

Example: 80 Million Tiny Images



[Torralba, Fergus, and Freeman, 2007. 80 Million Tiny Images.]

Conclusions

- Simple algorithm that does all its work at test time — in a sense, no learning!
- Can control the complexity by varying k
- Suffers from the Curse of Dimensionality
- Next time: decision trees, another approach to regression and classification

Questions?

?