CSC 311: Introduction to Machine Learning

Lecture 1 - Introduction

Amir-massoud Farahmand & Emad A.M. Andrews

University of Toronto

This course

- Broad introduction to machine learning
  - First half: algorithms and principles for supervised learning
    - nearest neighbors, decision trees, ensembles, linear regression, logistic regression, SVMs
  - Unsupervised learning: PCA, K-means, mixture models
  - Basics of reinforcement learning
- Coursework is aimed at advanced undergrads. We will use multivariate calculus, probability, and linear algebra.

Course Information

Recommended readings will be given for each lecture. But the following will be useful throughout the course:

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning, Second Edition, 2009.
- Christopher M. Bishop, Pattern Recognition and Machine Learning, 2006
- Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, Second Edition, 2018.
- Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, 2016
- Kevin Murphy, Machine Learning: A Probabilistic Perspective, 2012.
- Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, An Introduction to Statistical Learning, 2017.
- Shai Shalev-Shwartz and Shai Ben-David, Understanding Machine Learning: From Theory to Algorithms, 2014.
- David MacKay, Information Theory, Inference, and Learning Algorithms, 2003.

There are lots of freely available, high-quality ML resources.

What is machine learning?

- For many problems, it is difficult to program the correct behaviour by hand
  - recognizing people and objects
  - understanding human speech
- Machine learning approach: program an algorithm to automatically learn from data, or from experience
- Why might you want to use a learning algorithm?
  - hard to code up a solution by hand (e.g. vision, speech)
  - system needs to adapt to a changing environment (e.g. spam detection)
  - want the system to perform *better* than the human programmers
  - privacy/fairness (e.g. ranking search results)

What is machine learning?

- It is similar to statistics...
  - Both fields try to uncover patterns in data
  - Both fields draw heavily on calculus, probability, and linear algebra, and share many of the same core algorithms
- But it is not statistics!
  - Stats is more concerned with helping scientists and policymakers draw good conclusions; ML is more concerned with building autonomous agents
  - Stats puts more emphasis on interpretability and mathematical rigor; ML puts more emphasis on predictive performance, scalability, and autonomy

Relations to AI

- Nowadays, “machine learning” is often brought up with “artificial intelligence” (AI)
- AI does not often imply a learning based system
  - Symbolic reasoning
  - Rule based system
  - Tree search
  - etc.
- Learning based system $\rightarrow$ learned based on the data $\rightarrow$ more flexibility, good at solving pattern recognition problems.

Relations to human learning

- Human learning is:
  - Very data efficient
  - An entire multitasking system (vision, language, motor control, etc.)
  - Takes at least a few years :)
- For serving specific purposes, machine learning doesn't have to look like human learning in the end.
- It may borrow ideas from biological systems, e.g., neural networks.
- It may perform better or worse than humans.

What is machine learning?

- Types of machine learning
  - **Supervised learning:** access to labeled examples of the correct behaviour
  - **Reinforcement learning:** learning system (agent) interacts with the world and learn to maximize a reward signal
  - **Unsupervised learning:** no labeled examples – instead, looking for “interesting” patterns in the data

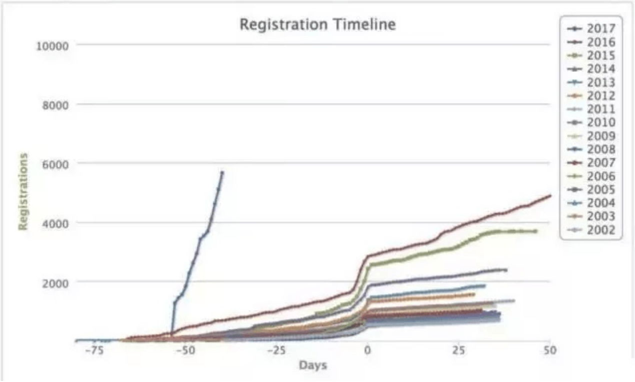
History of machine learning

- 1957 — Perceptron algorithm (implemented as a circuit!)
- 1959 — Arthur Samuel wrote a learning-based checkers program that could defeat him
- 1969 — Minsky and Papert's book *Perceptrons* (limitations of linear models)
- 1980s — Some foundational ideas
  - Connectionist psychologists explored neural models of cognition
  - 1984 — Leslie Valiant formalized the problem of learning as PAC learning
  - 1986 — Backpropagation (re-)discovered by Geoffrey Hinton and colleagues
  - 1988 — Judea Pearl's book *Probabilistic Reasoning in Intelligent Systems* introduced Bayesian networks

History of machine learning

- 1990s — the "AI Winter", a time of pessimism and low funding
- But looking back, the '90s were also sort of a golden age for ML research
  - Markov chain Monte Carlo
  - variational inference
  - kernels and support vector machines
  - boosting
  - convolutional networks
- 2000s — applied AI fields (vision, NLP, etc.) adopted ML
- 2010s — deep learning
  - 2010–2012 — neural nets smashed previous records in speech-to-text and object recognition
  - increasing adoption by the tech industry
  - 2016 — AlphaGo defeated the human Go champion

ML conferences selling out like Beyonce tickets.



Source: medium.com/syncedreview/

History of machine learning

ML conferences selling out like Beyonce tickets.

Computer vision: Object detection, semantic segmentation, pose estimation, and almost every other task is done with ML.

Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

Instance segmentation - Link

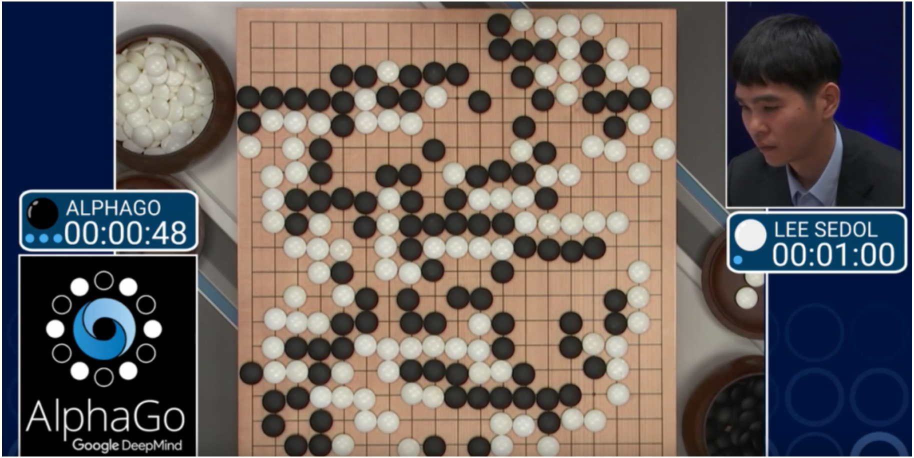
DAQUAR 1553
What is there in front of the sofa?
Ground truth: table
IMG+BOW: table (0.74)
2-VIS+BLSTM: table (0.88)
LSTM: chair (0.47)

COCOQA 5078
How many leftover donuts is the red bicycle holding?
Ground truth: three
IMG+BOW: two (0.51)
2-VIS+BLSTM: three (0.27)
BOW: one (0.29)

Speech: Speech to text, personal assistants, speaker identification...

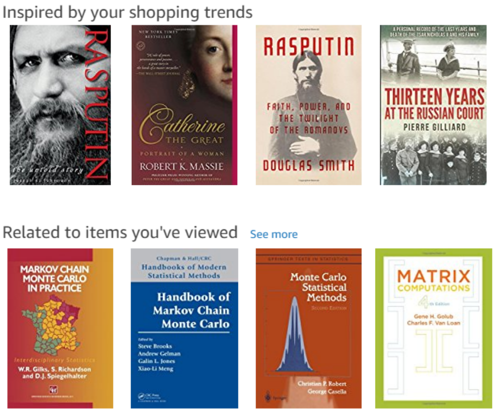
NLP: Machine translation, sentiment analysis, topic modeling, spam filtering.

Playing Games



DOTA2 - Link

E-commerce & Recommender Systems : Amazon, Netflix, ...



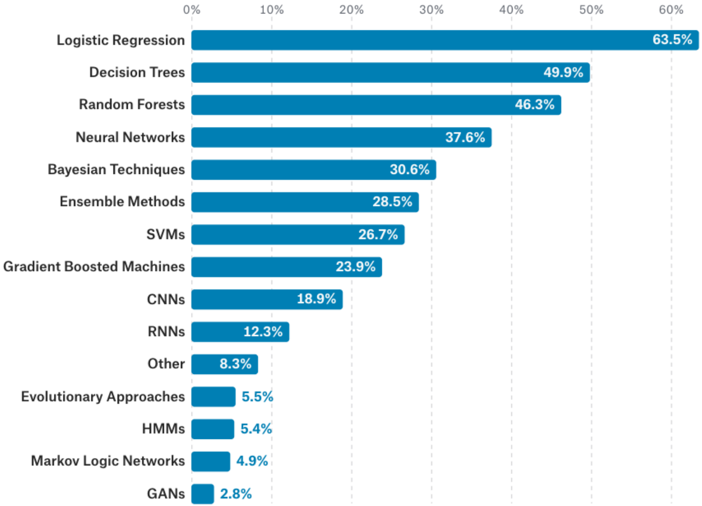
Why this class?

Why not jump straight to CSC412/421, and learn neural nets first?

- The principles you learn in this course will be essential to really understand neural nets.
- The techniques in this course are still the first things to try for a new ML problem.
  - For example, you should try applying logistic regression before building a deep neural net!
- There is a whole world of probabilistic graphical models.

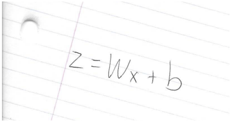
Why this class?

2017 Kaggle survey of data science and ML practitioners: what data science methods do you use at work?



Implementing machine learning systems

- You will often need to derive an algorithm (with pencil and paper), and then translate the math into code.
- Array processing (NumPy)
  - **vectorize** computations (express them in terms of matrix/vector operations) to exploit hardware efficiency
  - This also makes your code cleaner and more readable!




```
z = np.zeros(m)
for i in range(m):
    for j in range(n):
        z[i] += W[i, j] * x[j]
    z[i] += b[i]
```

```
z = np.dot(W, x) + b
```

Preliminaries and Nearest Neighbourhood Methods

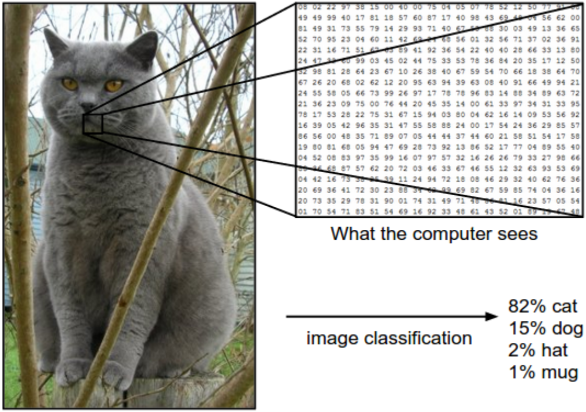
Introduction

- Today (and for the next 5-6 lectures) we focus on supervised learning.
- This means we are 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:



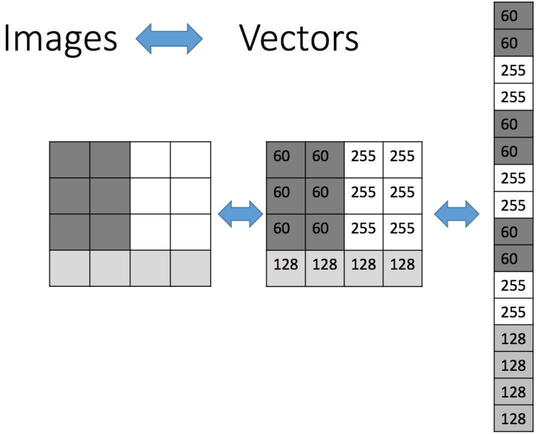
[Image credit: Andrej Karpathy]

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 is easy to manipulate
  - Vectors are a great representation since we can do linear algebra

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:

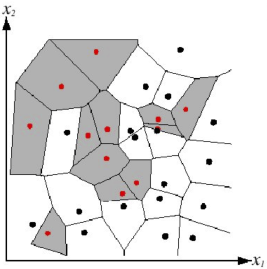
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\operatorname{argmin}} \quad \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

- Note: we do not need to compute the square root. Why?

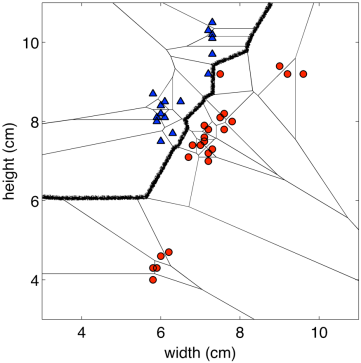
Nearest Neighbors: Decision Boundaries

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

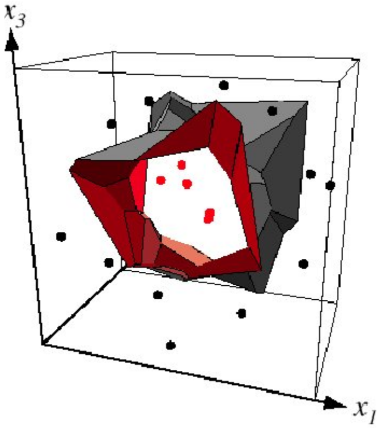


Nearest Neighbors: Decision Boundaries

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

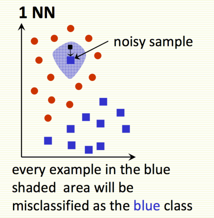


Nearest Neighbors: Decision Boundaries



Example: 2D decision boundary

Nearest Neighbors

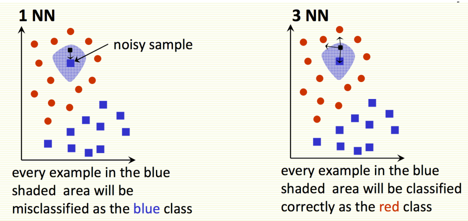


[Pic by Olga Veksler]

- Nearest neighbors sensitive to noise or mis-labeled data ("class noise"). Solution?

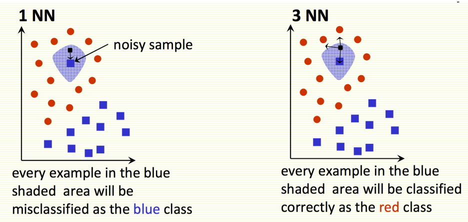
k-Nearest Neighbors

[Pic by Olga Veksler]



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

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):

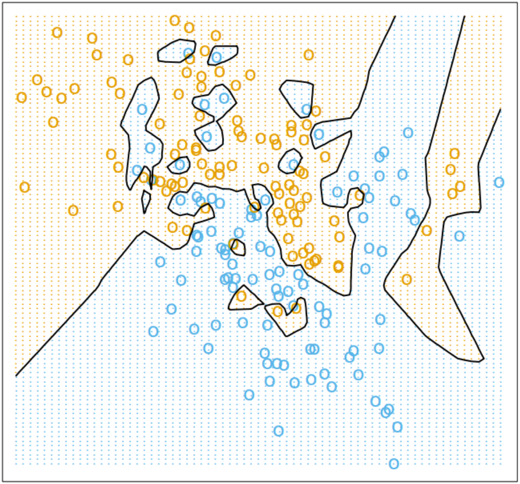
1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

$$y = \underset{t^{(z)}}{\operatorname{argmax}} \sum_{i=1}^{k} \mathbb{I}\{t^{(z)} = t^{(i)}\}$$

$\mathbb{I}\{\text{statement}\}$ is the identity function and is equal to one whenever the statement is true. We could also write this as $\delta(t^{(z)}, t^{(i)})$ with $\delta(a,b) = 1$ if $a = b$, 0 otherwise. $\mathbb{I}\{1\}$.

K-Nearest neighbors

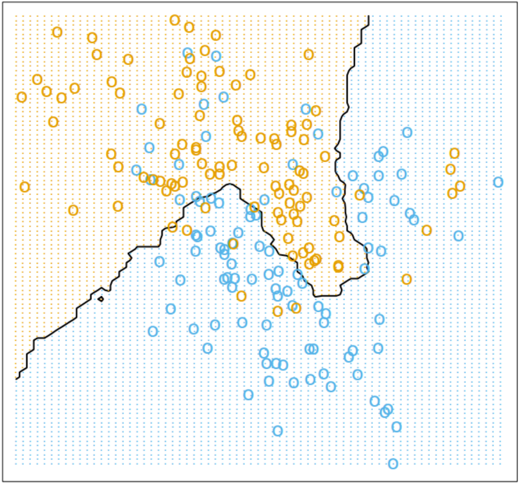
k=1



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

K-Nearest neighbors

k=15



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

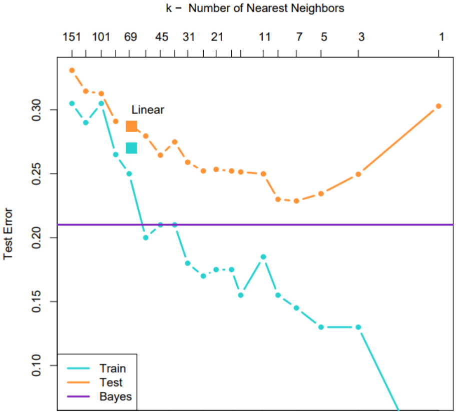
k-Nearest Neighbors

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
- Balancing k:
  - The optimal choice of k depends on the number of data points n.
  - Nice theoretical properties if $k \to \infty$ and $\frac{k}{n} \to 0$.
  - Rule of thumb: Choose $k = n^{\frac{2}{2+d}}$.
  - We explain an easier way to choose k using data.

K-Nearest neighbors

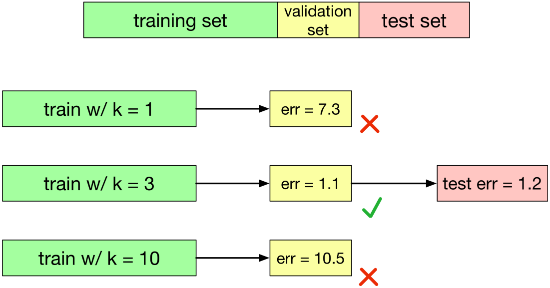
- We would like our algorithm to generalize to data it hasn't seen before.
- We can measure the generalization error (error rate on new examples) using a test set.



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

Validation and Test Sets

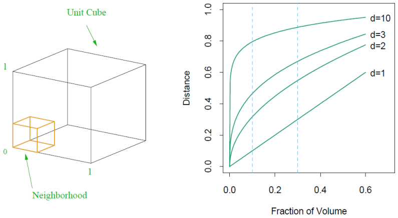
- k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself
- We can tune hyperparameters using a validation set:



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

Pitfalls: The Curse of Dimensionality

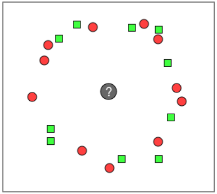
- Low-dimensional visualizations are misleading! In high dimensions, "most" points are far apart.
- If we want the nearest neighbor to be closer than ϵ, how many points do we need to guarantee it?
- The volume of a single ball of radius ϵ is $\mathcal{O}(\epsilon^d)$
- The total volume of $[0,1]^d$ is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



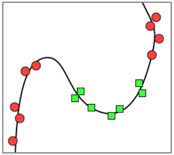
[Image credit: "The Elements of Statistical Learning"]

Pitfalls: The Curse of Dimensionality

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

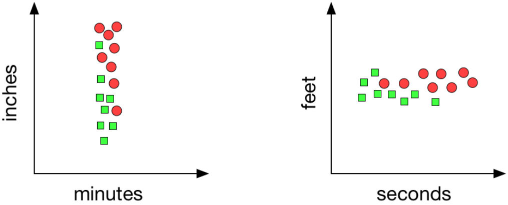


- Saving grace: some datasets (e.g. images) may have low intrinsic dimension, i.e. lie on or near a low-dimensional manifold. So nearest neighbors sometimes still works in high dimensions.



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

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

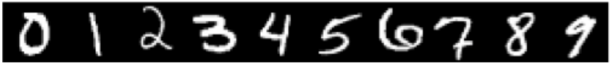
- Caution: depending on the problem, the scale might be important!

Pitfalls: Computational Cost

- Number of computations at training time: 0
- Number of computations at test time, per query (naïve algorithm)
  - Calculuate 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

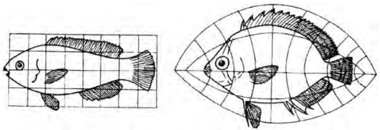


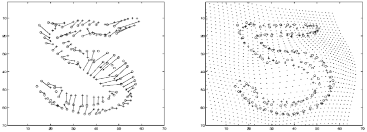
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	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.

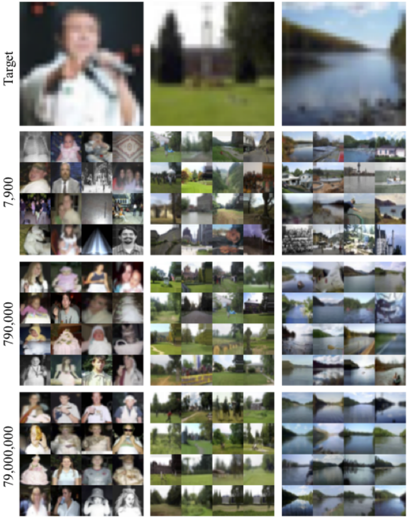




[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]

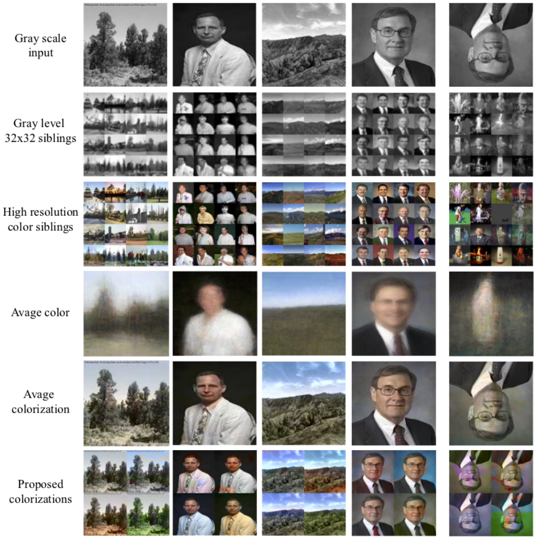
Example: 80 Million Tiny Images

- 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 be used for regression too, which we encounter later.
- Can control the complexity by varying k
- Suffers from the Curse of Dimensionality
- Next time: decision trees, another approach to regression and classification

Questions?

?