CSC 311: Introduction to Machine Learning

Lecture 1 - Introduction

Anthony Bonner & Lisa Zhang

Based on slides by Amir-massoud Farahmand & Emad A.M. Andrews

This course

- Broad introduction to machine learning
  - First half: algorithms and principles for supervised learning
    - nearest neighbors, decision trees, ensembles, linear regression, logistic regression
  - 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.

Do I have the appropriate background?

This is where we use all the math that you learned, and then some!

- Linear algebra (MAT223/240): vector/matrix manipulations, properties.
- Calculus (MAT232): partial derivatives/gradient
- Probability, Statistics (STA256): common distributions, Bayes Rule; expectation, variance, covariance, median; maximum likelihood.
- Geometry and geometric intuition.
- Recommend preparation: Numerical Methods (CSC338) or Computational Statistics

Mathematical maturity will be assumed: you should be able to write and understand rigorous proofs.

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.

Requirements and Marking

- Three (3) assignments
  - Combination of pen & paper derivations and programming exercises
  - Due Oct 8, Nov 15, Dec 7 at 11:59pm
- Midterm
  - October 22, 8–9pm.
- Final Exam
  - Two (or Three) hours
  - Date and time TBA

More on Assignments

Collaboration on the assignments is not allowed. Each student is responsible for their own work. Discussion of assignments should be limited to clarification of the handout itself, and should not involve any sharing of pseudocode or code or simulation results. Violation of this policy is grounds for a semester grade of F, in accordance with university regulations.

The schedule of assignments will be posted on the course webpage.

Extensions will be granted only in special situations, and you will need a Student Medical Certificate or a written request approved by the course coordinator at least one week before the due date.

Related Courses and a Note on the Content

- More advanced ML courses such as **CSC413** (Neural Networks and Deep Learning) and **CSC412** (Probabilistic Learning and Reasoning) both build upon the material in this course.
- If you've already taken an applied statistics course, there will be some overlap.
- Warning: This is not a very difficult course, but it has a lot of content. To be successful, you need to work hard. For example, don't start working on your homework assignments just two days before the deadline.

What is learning?

"The activity or process of gaining knowledge or skill by studying, practicing, being taught, or experiencing something."

Merriam Webster dictionary

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Tom Mitchell

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

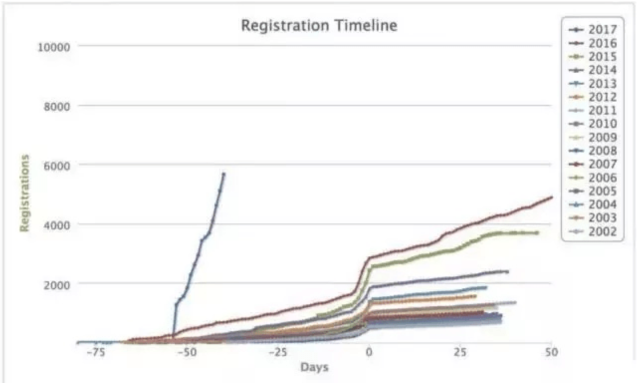
- Human learning is:
  - Very data efficient
  - An entire multitasking system (vision, language, motor control, etc.)
  - Takes at least a few years :)
- For serving specific purposes, machine learning doesn't have to look like human learning in the end.
- It may borrow ideas from biological systems,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

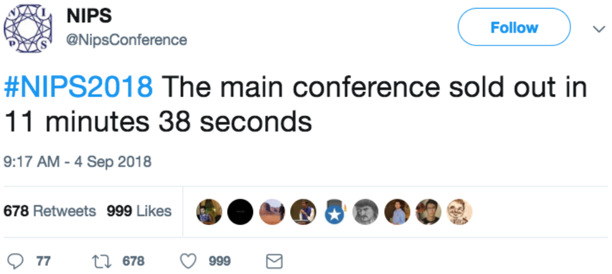
ML conferences selling out like Beyonce tickets.



Source: medium.com/syncedreview/

History of machine learning

ML conferences selling out like Beyonce tickets.


NIPS
@NipsConference

#NIPS2018 The main conference sold out in 11 minutes 38 seconds

9:17 AM - 4 Sep 2018

678 Retweets 999 Likes

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

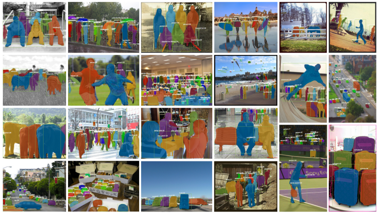
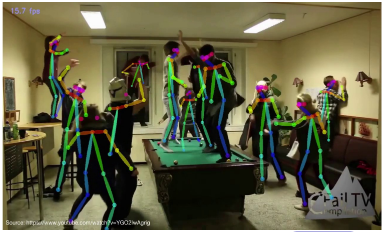
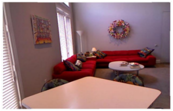


Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).



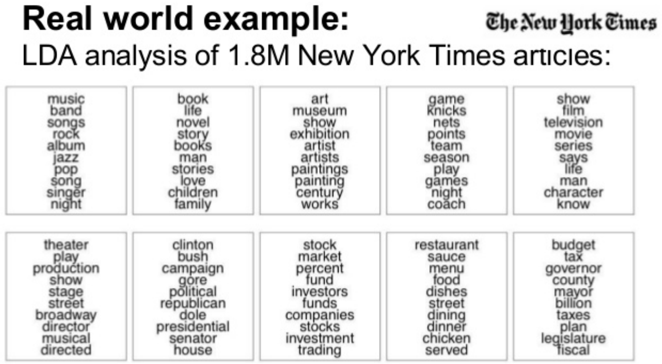

DAQUAR 1553
What is there in front of the sofa?
Ground truth: table
IMG+BOW: table (0.74)
2-VIS+BLSTM: table (0.88)
LSTM: chair (0.47)

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)

Instance segmentation - ▸ Link

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

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



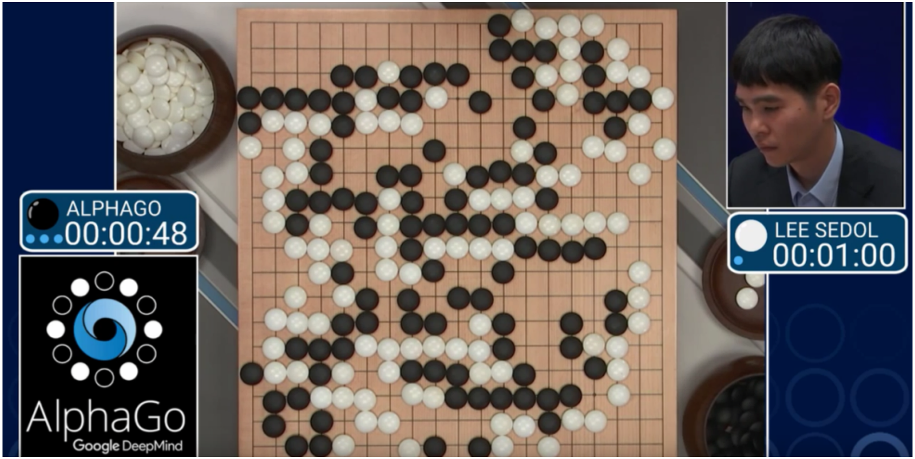
Real world example: The New York Times

LDA analysis of 1.8M New York Times articles:

music	book	art	game	show
band	life	museum	knicks	film
songs	novel	show	nets	television
rock	story	exhibition	points	movie
album	books	artist	team	series
jazz	man	artists	season	says
pop	stories	paintings	play	life
song	love	painting	games	man
singer	children	century	night	character
night	family	works	coach	know

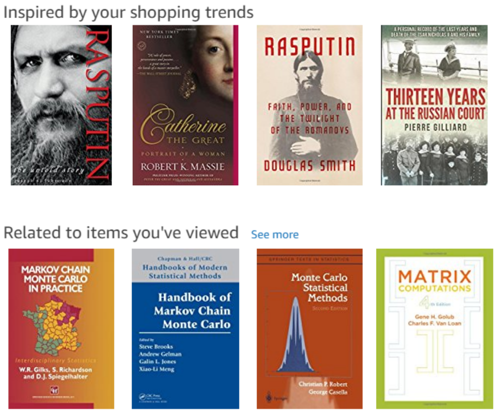
theater	clinton	stock	restaurant	budget
play	bush	market	sauce	tax
production	campaign	percent	menu	governor
show	gore	fund	food	county
stage	political	investors	dishes	mayor
street	republican	funds	street	billion
broadway	dole	companies	dining	taxes
director	presidential	stocks	dinner	plan
musical	senator	investment	chicken	legislature
directed	house	trading	served	fiscal

Playing Games



DOTA2 - ▸ Link

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


Inspired by your shopping trends
RASPUTIN
Catherine the Great
RASPUTIN
THIRTEEN YEARS AT THE RUSSIAN COURT
Related to items you've viewed
See more
MARKOV CHAIN MONTE CARLO IN PRACTICE
Handbook of Markov Chain Monte Carlo
Monte Carlo Statistical Methods
MATRIX COMPUTATIONS

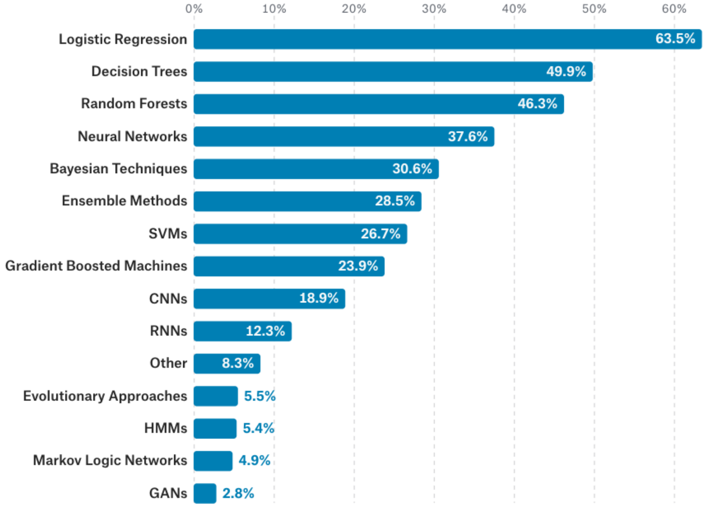
Why this class?

Why not jump straight to CSC412/413, 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.

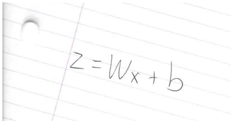
Why this class?

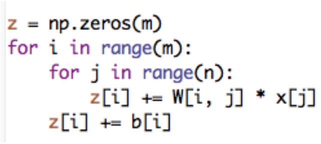
2017 Kaggle survey of data science and ML practitioners: what data science methods do you use at work?



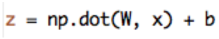
Implementing machine learning systems

- 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

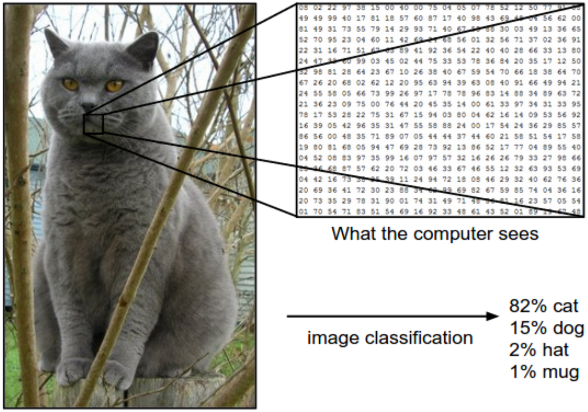
Introduction

- 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
⋮	⋮	⋮

Supervised Learning Example

What an image looks like to the computer:



[Image credit: Andrej Karpathy]

Representing the Input

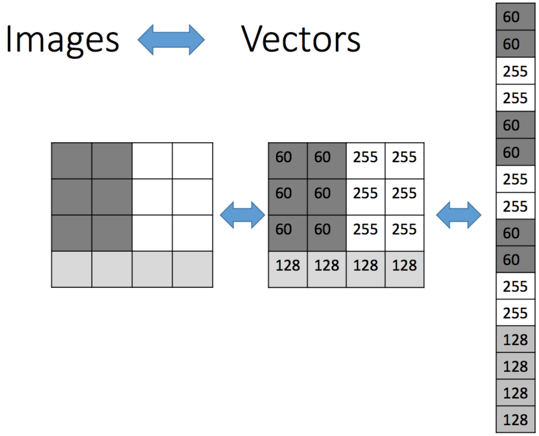
- Machine learning algorithms need to handle lots of types of data: images, text, audio waveforms, credit card transactions, etc.
- Common strategy: represent the input as an input vector in $\mathbb{R}^d$
  - Representation = mapping to another space that is easy to manipulate
  - Vectors are a great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

Supervised Learning Setup

- 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 $\{1, \dots, C\}$
  - These days, t is often a highly structured object (e.g. image)
- Denote the training set $\{(\mathbf{x}^{(1)}, t^{(1)}), \dots, (\mathbf{x}^{(N)}, t^{(N)})\}$
  - Note: these superscripts have nothing to do with exponentiation!

Input Vectors

Can use raw pixels:



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

Nearest Neighbour

Our first *supervised learning* model!

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.

Nearest Neighbor: Example

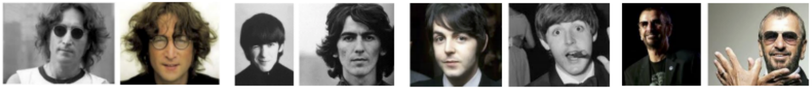
Example: We would like to classify which Beetle is in this image $\mathbf{x}$:

We find the *closest* image $\mathbf{x}^{(i)}$ in the training set, and look at its label.

Nearest Neighbor: Example

Example: We would like to classify which Beetle is in this image $\mathbf{x}$:

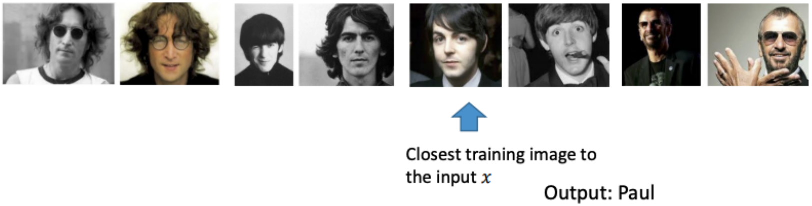
We find the *closest* image $\mathbf{x}^{(i)}$ in the training set, and look at its label.



Nearest Neighbor: Example

Example: We would like to classify which Beetle is in this image $\mathbf{x}$:

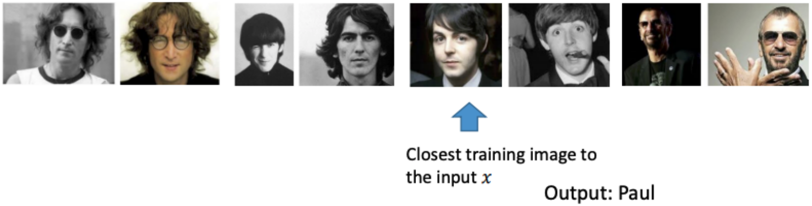
We find the *closest* image $\mathbf{x}^{(i)}$ in the training set, and look at its label.



Nearest Neighbor: Example

Example: We would like to classify which Beetle is in this image $\mathbf{x}$:

We find the *closest* image $\mathbf{x}^{(i)}$ in the training set, and look at its label.



But how do we determine what is the "closest" image?

The 1-Nearest Neighbor Algorithm

- 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:

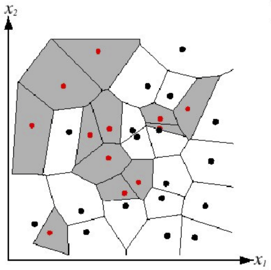
1. Given
   - input $\mathbf{x}$ (that we would like to classify),
   - training set $(\mathbf{x}^{(1)}, t^{(1)}), (\mathbf{x}^{(2)}, t^{(2)}), \ldots$.

   Find an example $(\mathbf{x}^{(*)}, t^{(*)})$ in the training set closest to $\mathbf{x}$, i.e. that minimizes distance$(\mathbf{x}^{(*)}, \mathbf{x})$.
2. Output $y = t^{(*)}$

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

Nearest Neighbors: Decision Boundaries

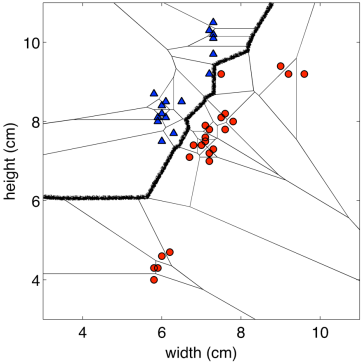
We can visualize the behaviour in the classification setting using a Voronoi diagram.



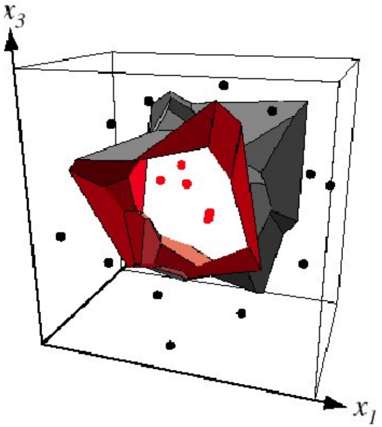
Here, our data set $\mathbf{x} \in \mathbb{R}^2$ (e.g. a 2-pixel image), and there are two possible labels t either red or blue.

Nearest Neighbors: Decision Boundaries

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



Nearest Neighbors: Decision Boundaries



Example: Decision boundary in a 3D data set.

Estimating Model Accuracy

- Typically, we want to have some sense of how well our model performs.
- For example, we want to use this model in a real-world application, and want to know how reliable the predictions will be.
- There are several ways to measure model performance. The simplest is the accuracy mesaure:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

The Test Set

What might be an issue with computing the accuracy using the training set?

The Test Set

What might be an issue with computing the accuracy using the training set?

- We will always get perfect performance in our 1-nearest neighbor model (a data point is always closest to itself!)
- Not a realistic representation of how our model will perform on unseen data.

The Test Set

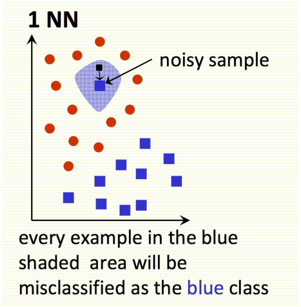
What might be an issue with computing the accuracy using the training set?

- We will always get perfect performance in our 1-nearest neighbor model (a data point is always closest to itself!)
- Not a realistic representation of how our model will perform on unseen data.

Solution: set aside a test set (from our set of labeled data) so we can estimate how well our model will generalize.

Nearest Neighbors

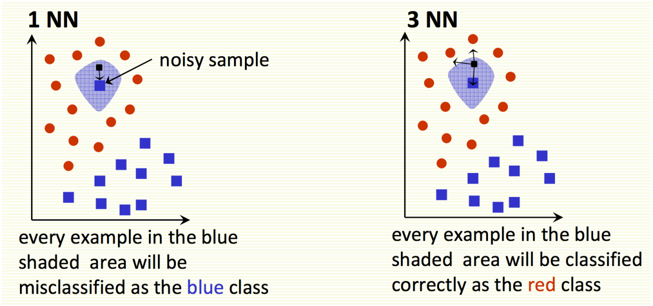
[Pic by Olga Veksler]


1 NN
noisy sample
every example in the blue shaded area will be misclassified as the blue class

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

k-Nearest Neighbors

[Pic by Olga Veksler]


1 NN
noisy sample
every example in the blue shaded area will be misclassified as the blue class
3 NN
every example in the blue shaded area will be classified correctly as the red class

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

the K-Nearest neighbors Algorithm

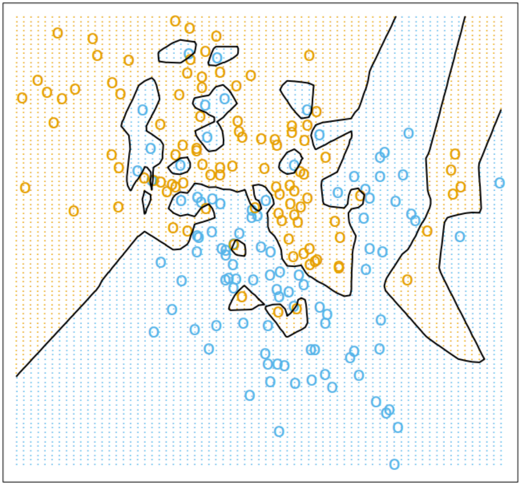
Algorithm (kNN):

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

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

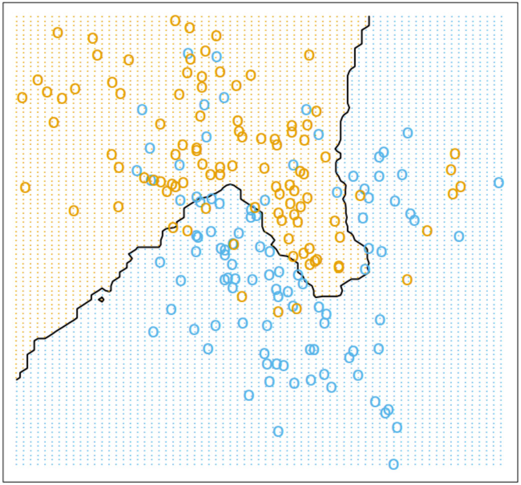
$\mathbb{I}\{\text{statement}\}$ is the identity function and is equal to one whenever the statement is true. We could also write this as $\delta(t^{(z)}, t^{(i)})$ with $\delta(a, b) = 1$ if $a = b$, 0 otherwise. $\mathbb{I}\{1\}$.

K-Nearest neighbors (k=1)



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

K-Nearest neighbors (k=15)



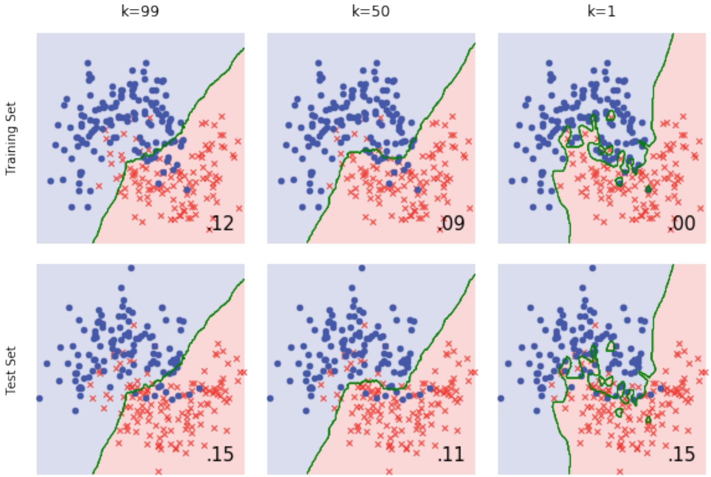
[Image credit: "The Elements of Statistical Learning"]

How to choose 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 will explain another way to choose k using our data.

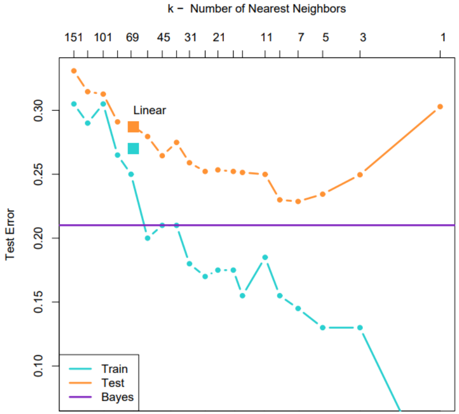
Underfitting and Overfitting



[Image credit: https://cambridgecoding.wordpress.com/2016/03/24/misleading-modelling-overfitting-cross-validation-and-the-bias-variance-trade-off/

Choice of k: Generalization

- 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 the test set.



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

The problem with using the test set

Here, k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself.

What are potential issues with using the test set to choose k?

The problem with using the test set

Here, k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself.

What are potential issues with using the test set to choose k?

- The test set is meant to simulate unseen data, in order to estimate how well the model generalizes.
- We therefore shouldn't use the test set to make any decisions about any aspects of the model!
- Keep the test set unseen, so that we can estimate model accuracy.

The problem with using the test set

Here, k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself.

What are potential issues with using the test set to choose k?

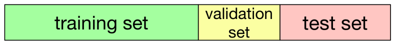
- The test set is meant to simulate unseen data, in order to estimate how well the model generalizes.
- We therefore shouldn't use the test set to make any decisions about any aspects of the model!
- Keep the test set unseen, so that we can estimate model accuracy.

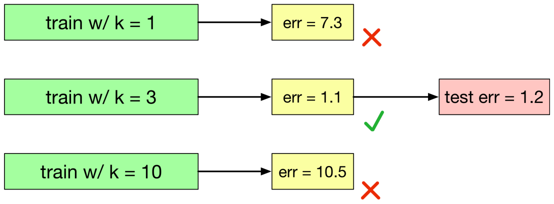
What can we do instead?

Validation and Test Sets

Partition the data into three subsets

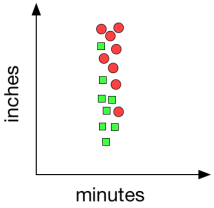
- Training set: used to train the model
- Validation set: used to tune hyperparameters
- Test set: used to evaluate final performance *once*, after hyperparameters are chosen

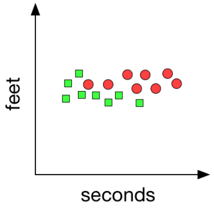




Pitfalls: Normalization

- 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

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

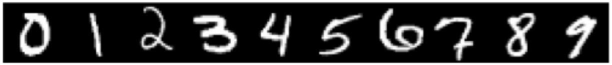
- Caution: depending on the problem, the scale might be important!

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

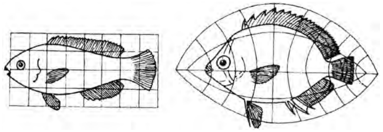


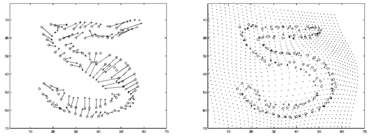
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	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.

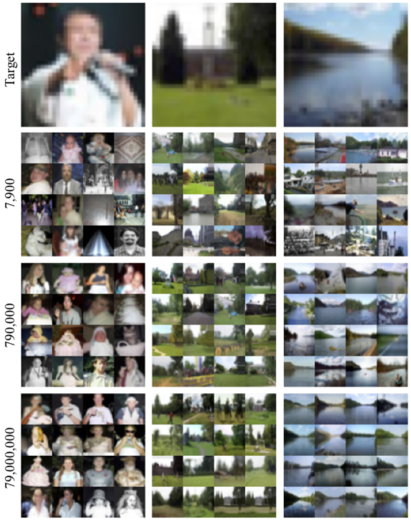




[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]

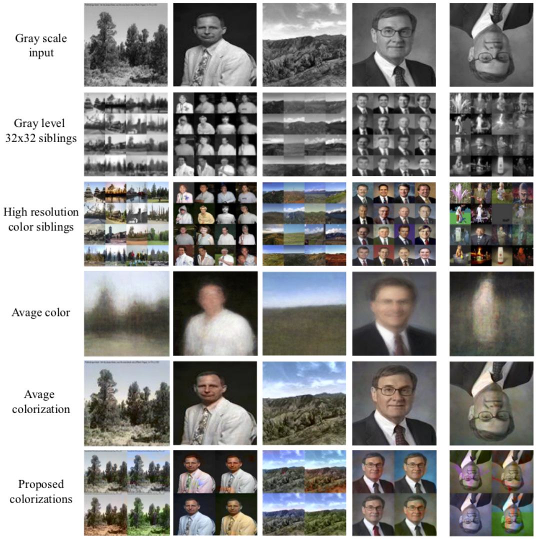
Example: 80 Million Tiny Images

- 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: Colorizing 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
- Next time: decision trees, another approach to regression and classification

Questions?

?