CSC 311: Introduction to Machine Learning

Lecture 1 - Introduction and Nearest Neighbors

Roger Grosse   Rahul G. Krishnan   Guodong Zhang

University of Toronto, Fall 2021

This course

- Broad introduction to machine learning
  - 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

Course Website:
`https://www.cs.toronto.edu/~rgrosse/courses/csc311_f21/`
Main source of information is the course webpage; check regularly!

Announcements, grades, & links: Quercus.

- Did you receive the announcement?

Discussions: Piazza.

- Sign up: `piazza.com/utoronto.ca/fall2021/csc311`
- Your grade **does not depend on your participation on Piazza**. It's just a good way for asking questions, discussing with your instructor, TAs and your peers. We will only allow questions that are related to the course materials/assignments/exams.

Office hours: Gather Town (see Quercus for link)

- Roger Grosse, Monday 10am–noon
- Rahul G. Krishnan, Monday 6pm-7pm
- Guodong Zhang, Monday 8–9pm

You only **need** to pay attention to the course website for content and Quercus for links.

Course Information

- For the first two weeks of class, lectures will be delivered synchronously via Zoom, and recorded for asynchronous viewing by enrolled students.
- Current plan is to resume in-person lectures starting Sept. 23.
  - This will depend on how the COVID-19 situation evolves, as well as on guidance from the University and from public health authorities.
  - One section will be held virtually. You won't be at a disadvantage if you choose to attend the virtual section.
  - If you attend in-person, **you must attend your assigned section**. To request to switch sections, please see the course website for instructions.
- Tutorials and office hours will be virtual throughout the term.
- All information about attending virtual lectures, tutorials, and office hours will be sent to enrolled students through Quercus.
- Last year's video lectures are also available through Quercus.

Course Information

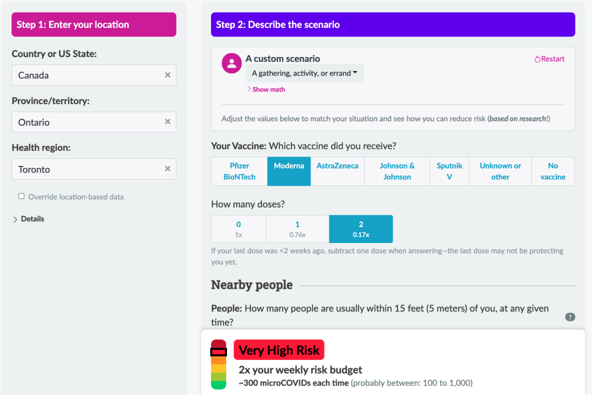
- You may download recorded lectures for your own academic use, but you should not copy, share, or use them for any other purpose.
- During lecture, please keep yourself on mute unless called upon.
- In case of illness, you should fill out the absence declaration form on ACORN and notify the instructors to request special consideration.
- For accessibility services: If you require additional academic accommodations, please contact UofT Accessibility Services as soon as possible, `studentlife.utoronto.ca/as`.

In-Person Attendance?

- Still in Step 3 of Ontario's reopening plan, but educational institutions were granted an exception to capacity limits on in-person gatherings. `https://www.ontario.ca/page/reopening-ontario`
- This decision was made from a public health perspective. This is not the same thing as safety to you as an individual.
- Consider your individual risk tolerance (especially if you live with someone vulnerable!).
- Continue to follow University and public health guidelines.

In-Person Attendance?

The MicroCovid Calculator (`https://www.microcovid.org/`) is helpful for managing your own level of risk.



Course Information

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

- Hastie, Tibshirani, and Friedman: “The Elements of Statistical Learning”
- Christopher Bishop: “Pattern Recognition and Machine Learning”, 2006.
- Kevin Murphy: “Machine Learning: a Probabilistic Perspective”, 2012.
- David Mackay: “Information Theory, Inference, and Learning Algorithms”, 2003.
- Shai Shalev-Shwartz & Shai Ben-David: “Understanding Machine Learning: From Theory to Algorithms”, 2014.
- David Barber: ”Bayesian Reasoning and Machine Learning”, 2012.
- Richard S. Sutton and Andrew G. Barto: ”Reinforcement Learning: An Introduction” (2nd ed.), 2018.

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

Requirements and Marking

- (35%) 3 homework assignments
  - Combination of pen & paper derivations and programming exercises
  - Weighted equally
- (5%) minor assignments (e.g. embedded ethics)
  - Good faith effort = full credit
- (20%) project
  - Groups of 2–3
  - Implement and evaluate some ML algorithms, try out a new idea, write a report
- (40%) midterm test and final exam
  - Both online
  - See website for times and dates
  - Weighting: higher of (15% midterm, 25% final) or (10% midterm, 30% final)

More on Assignments

Collaboration on the assignments is not allowed. Each student is responsible for his/her 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.

Assignments should be handed in by deadline; a late penalty of 10% per day will be assessed thereafter (up to 3 days, then submission is blocked).

Extensions will be granted only in special situations, and you will need to complete an absence declaration form and notify us to request special consideration, or otherwise have a written request approved by the course instructors at least one week before the due date.

Related Courses

- 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.
- This is the third academic year this course is listed only as an undergrad course. Previously it was CSC411, with a bit more content and heavier workload. We borrow liberally from the previous editions.

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's difficult to program the correct behavior 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)

Relations to statistics

- It's 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
- it's not *exactly* 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
- ...but machine learning and statistics rely on similar mathematics.

What is machine learning?

- Types of machine learning
  - **Supervised learning:** have labeled examples of the correct behavior
  - **Reinforcement learning:** learning system (agent) interacts with the world and learns to maximize a scalar 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
  - 1988 — 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
  - reinforcement learning
- 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
  - 2018-now — generating photorealistic images and videos
  - 2020 — GPT3 language model
- now — increasing attention to ethical and societal implications

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).

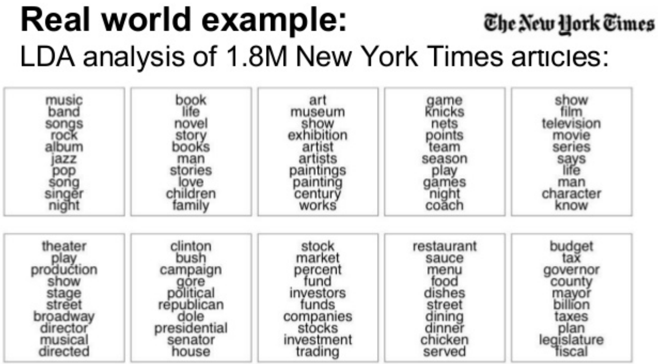
Instance segmentation - Link

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.

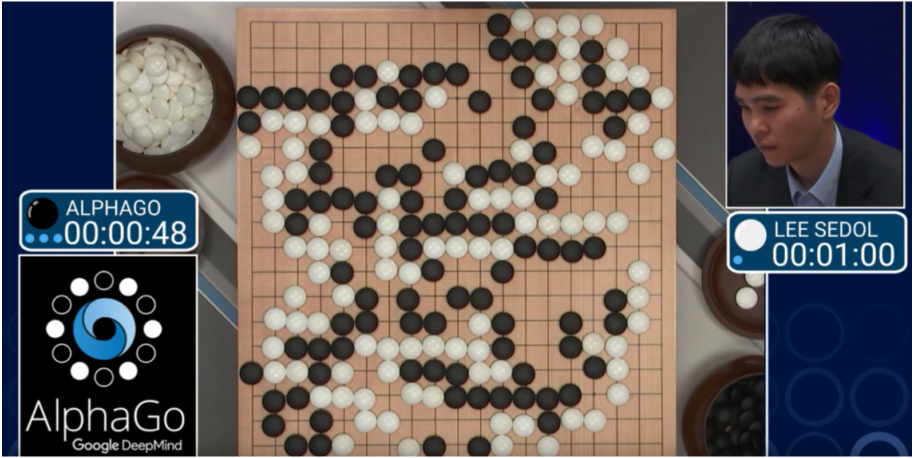


Real world example: The New York Times

LDA analysis of 1.8M New York Times articles:

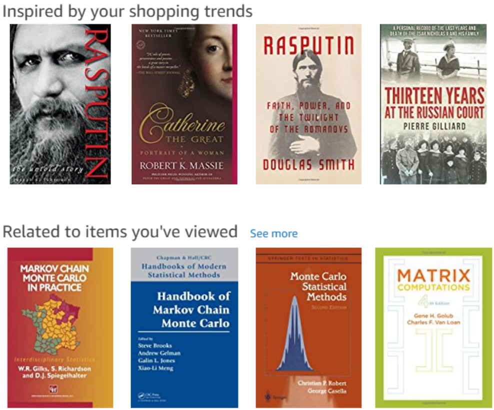
music band songs rock album jazz pop song singer night	book life novel story books man stories love children family	art museum show exhibition artist artists paintings painting century works	game knicks nets points team season play games night coach	show film television movie series says life man character know
theater play production show stage street broadway director musical directed	clinton bush campaign gore political republican dole presidential senator house	stock market percent fund investors funds companies stocks investment trading	restaurant sauce menu food dishes street dining dinner chicken served	budget tax governor county mayor billion taxes plan legislature fiscal

Playing Games



DOTA2 - Link

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



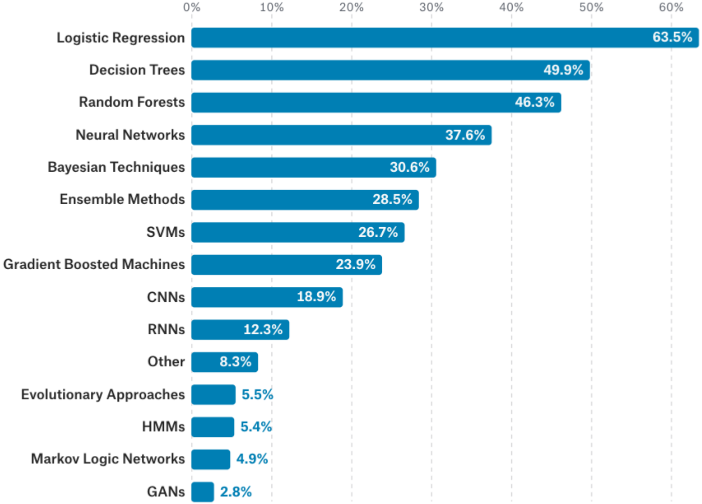
Why this class?

Why not jump straight to csc412/413, and learn neural nets first?

- The principles you learn in this course will be essential to understand and apply neural nets.
- The techniques in this course are still the first things to try for a new ML problem.
  - E.g., try logistic regression before building a deep neural net!
- There's a whole world of probabilistic graphical models.

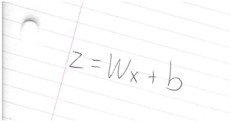
Why this class?

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 = W @ x + b
```

Implementing machine learning systems

- Neural net frameworks: PyTorch, TensorFlow, JAX, etc.
  - automatic differentiation
  - compiling computation graphs
  - libraries of algorithms and network primitives
  - support for graphics processing units (GPUs)
- Why take this class if these frameworks do so much for you?
  - So you know what to do if something goes wrong!
  - Debugging learning algorithms requires sophisticated detective work, which requires understanding what goes on beneath the hood.
  - That's why we derive things by hand in this class!

Preliminaries and Nearest Neighbor Methods

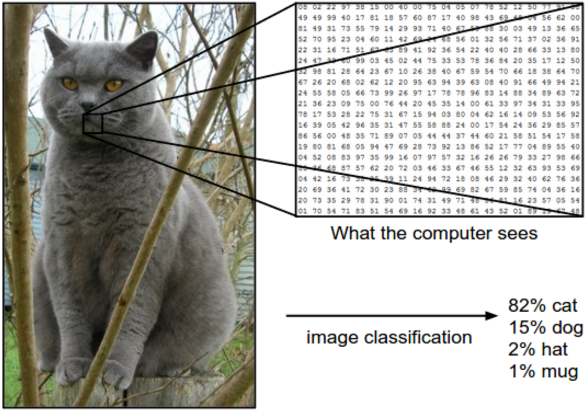
Introduction

- Today (and for much of this course) 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:



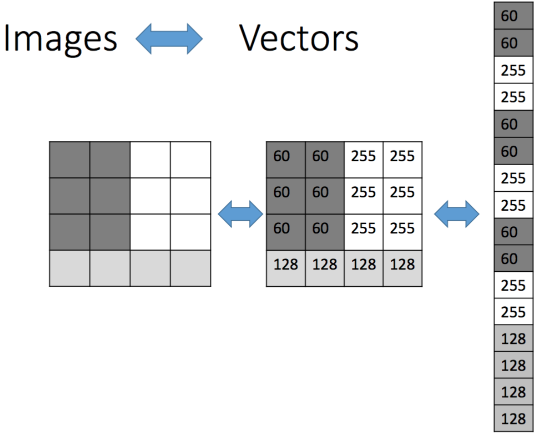
[Image credit: Andrej Karpathy]

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's 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:

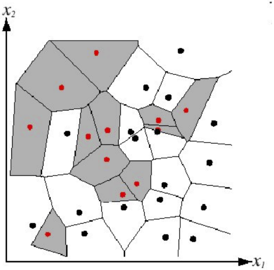
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\operatorname{argmin}} \ \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

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

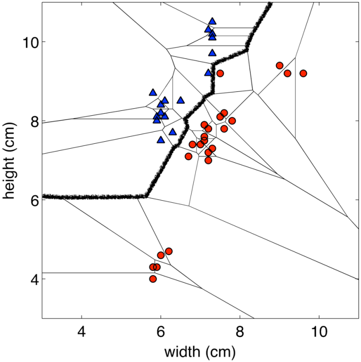
Nearest Neighbors: Decision Boundaries

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

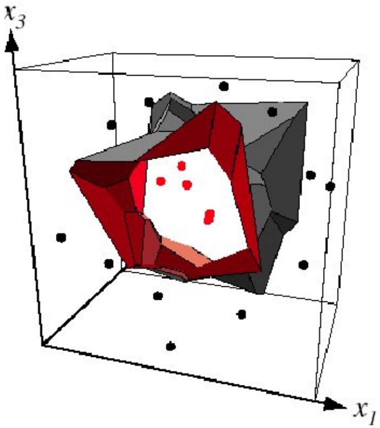


Nearest Neighbors: Decision Boundaries

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

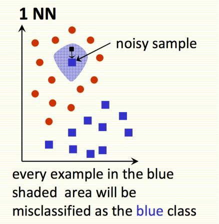


Nearest Neighbors: Decision Boundaries



Example: 2D decision boundary

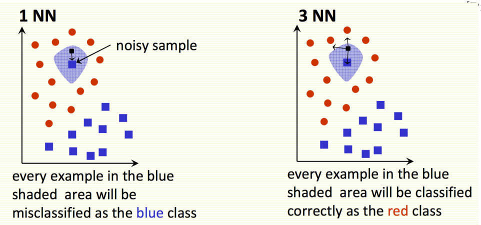
Nearest Neighbors



[Pic by Olga Veksler]

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

k-Nearest Neighbors

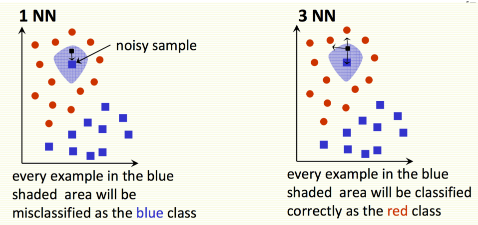


[Pic by Olga Veksler]

- 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):

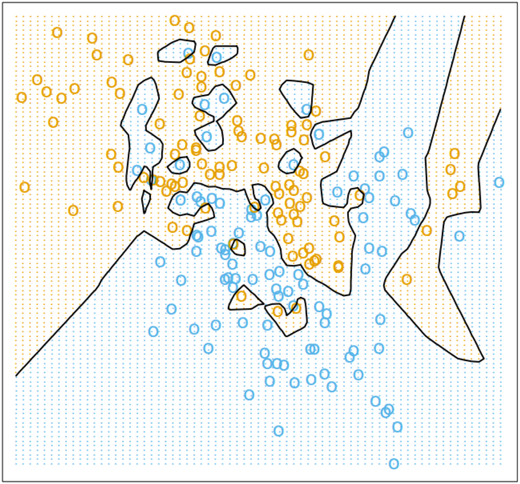
1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

$$y = arg \max_{t^{(z)}} \sum_{i=1}^{k} \mathbb{I}(t^{(z)} = t^{(i)})$$

$\mathbb{I}\{\text{statement}\}$ 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.

K-Nearest neighbors

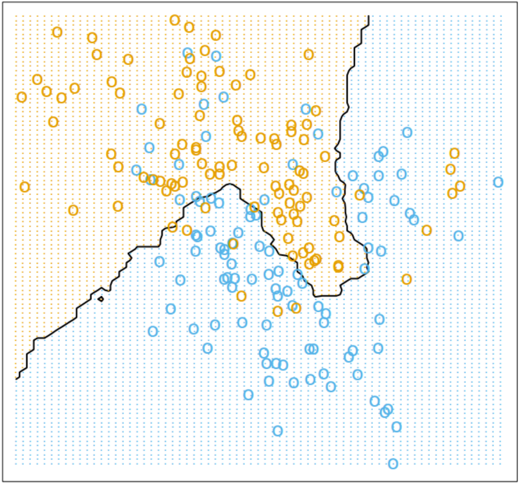
k=1



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

K-Nearest neighbors

k=15



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

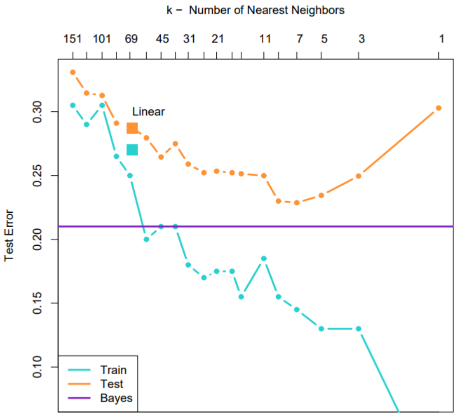
k-Nearest Neighbors

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
  - Optimal choice of k depends on number of data points n.
  - Nice theoretical properties if $k \to \infty$ and $\frac{k}{n} \to 0$.
  - Rule of thumb: choose $k < \sqrt{n}$.
  - We can choose k using validation set (next slides).

K-Nearest neighbors

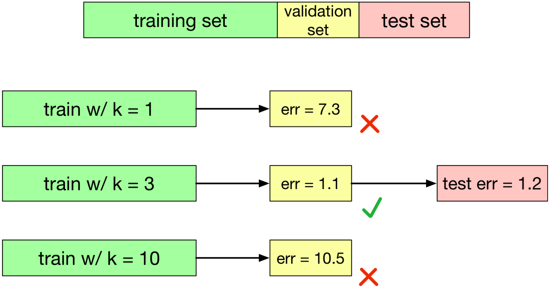
- We would like our algorithm to generalize to data it hasn't seen before.
- We can measure the generalization error (error rate on new examples) using a test set.



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

Validation and Test Sets

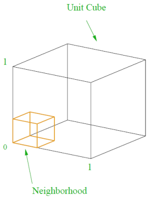
- k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself
- We can tune hyperparameters using a validation set:

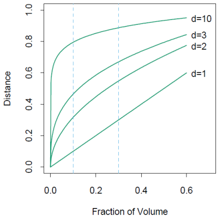


- The test set is used only at the very end, to measure the generalization performance of the final configuration.

Pitfalls: The Curse of Dimensionality

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

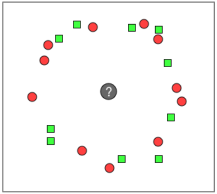




[Image credit: "The Elements of Statistical Learning"]

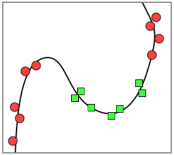
Pitfalls: The Curse of Dimensionality

- In high dimensions, “most” points are approximately the same distance.
- We can show this by applying the rules of expectation and covariance of random variables in surprising ways. (“optional” homework question coming up...)
- Picture to keep in mind:



Pitfalls: The Curse of Dimensionality

- Saving grace: some datasets (e.g. images) may have low intrinsic dimension, i.e. lie on or near a low-dimensional manifold.



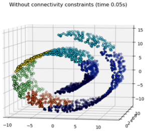
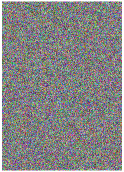


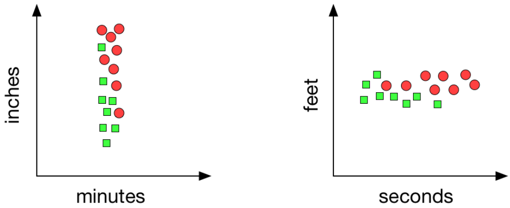
Image credit:
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html

- The neighborhood structure (and hence the Curse of Dimensionality) depends on the intrinsic dimension.
- The space of megapixel images is 3 million-dimensional. The true number of degrees of freedom is much smaller.



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

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

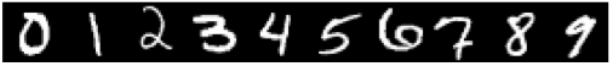
- Caution: depending on the problem, the scale might be important!

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

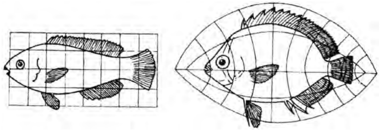


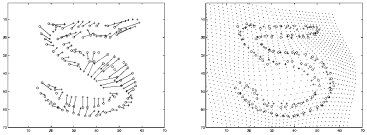
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: d = 784
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	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.

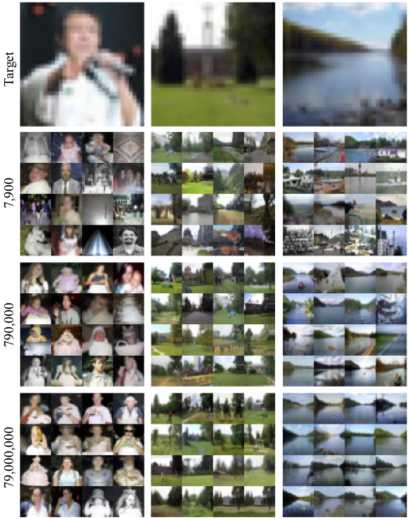




[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]

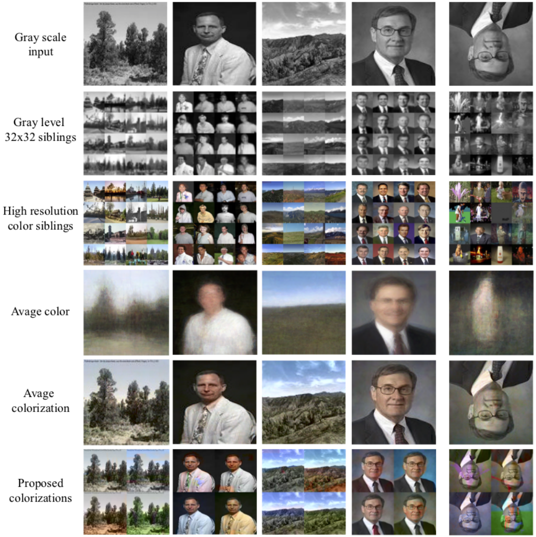
Example: 80 Million Tiny Images

- 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.
- 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: parametric models, which learn a compact summary of the data rather than referring back to it at test time.

Questions?

?