① We'll start at 1:10

② Math diagnostic due on Friday → if enrolled → MARKUS
→ if waitlisted → course email.

③ All information + announcements will be on the course website

CSC 311: Introduction to Machine Learning

Lecture 1 - Introduction and Nearest Neighbors

Rahul G. Krishnan

University of Toronto, Fall 2023

Outline

1. Introductions

2. Admin Details

3. What is Machine Learning?

4. Preliminaries and Nearest Neighbor Methods

Instructors

Rahul G. Krishnan

Rahul G. Krishnan

- Grew up and went to high school in Mumbai, India
- Undergrad in ECE at UofT
- MS in CS at NYU
- PhD in EECS at MIT
- Senior Researcher at Microsoft Research New England
- Assistant Professor in CS at UofT
- Research in Machine Learning and Healthcare

Meet Your Peers

Find someone you don't know and introduce yourself.
Make a new friend!

Marking Scheme

Component	**% Final Grade**
1 Mathematics Pre-assessment	3%
3 Assignments	27%, 9% each
Embedded Ethics assignments	5%
Project	10%
Midterm Exam	20%
Final Exam	35% (40% auto-fail threshold)

5% embedded ethics assignments:

Assignment	**% Final Grade**	**Marking**
Pre-module survey	1%	Full credit for submitting.
Class participation	0.5%	Full credit for 90% attendance.
Written Reflection	2%	Full credit for a good-faith effort.
Post-module survey	1.5%	Full credit for submitting.

Recommended Textbooks

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

Here are some recommended textbooks.

- Bishop: Pattern Recognition and Machine Learning.
- Hastie, Tibshirani, and Friedman: The Elements of Statistical Learning.
- MacKay: Information Theory, Inference, and Learning Algorithms.
- Barber: Bayesian Reasoning and Machine Learning.
- Sutton and Barto: Reinforcement Learning: An Introduction.
- Deisenroth, Faisal, and Ong: Math for ML.
- Shalev-Shwartz and Ben-David: Understanding Machine Learning: From Theory to Algorithms.
- Kevin Murphy: Machine Learning: a Probabilistic Perspective.

Course Components

- Math Diagnostic (Due September 15 2023)
- Assignments
- Project
- Midterm
- Final

Assignments

- Theoretical and programming questions.
- Due on MarkUs before 11:59 pm on due date.
- Late Policy: Deadlines are firm. Each person gets three grace days they can use throughout the semester. No credit after deadline+grace days.
- Collaboration Policy: You should absolutely help each other learn the course material. Discussing strategies to solve the homework is OK but the work you submit must be your own work. Do not give out your written material or use someone elses.

Project

- Groups of 2-3.
- 4 weeks.
- Implement and evaluate several algorithms from the course.
- Propose and evaluate an extension of one algorithm or choose your own adventure!
- Will post instructions and starter code before reading week.

Exams

- Conceptual questions.
- Midterm
  - Tutorial time slots on Oct 26 and 27.
  - Must attend your registered section.
  - Can bring one double-sided reference sheet.
- Final Exam
  - 3-hour exam.
  - Date/time will be released around late November.
  - Do not book travel plans until your final exam schedule is released.

Academic Integrity

- Cheating only cheats yourself!
  - Consult U of T Code of Behaviour on Academic Matters
- What you should do for assignments:
  - Ask questions during office hours.
  - Discuss ideas and code examples with others.
  - Write code on your own.
  - Say no to sending code to others.
- What you should do for tests and exams:
  - Create practice questions.
  - Test yourself/each other under time pressure.

Generative AI and Learning TLDR

See full policy on course website.

- You do not need to use any such tools to succeed in the class.
- Your knowledge of the concepts will be tested on the final exam and midterm exam (55% of your grade) where you will *not* have access to generative AI tools.
- Use generative AI models for learning. But, any use of GPT4/ChatGPT must be documented.
- Use GPT4/ChatGPT to help understand and personalize concepts taught in homework and lectures!
- Models can hallucinate and can fail in unpredictable ways.
- Using GPT4/ChatGPT's output directly in any material handed in for homework constitutes an academic violation. You are expected to write own homework assignments even if such tools were used as aids to learn concepts.

Strategies for Success

- **Time Management**
    1. The best time to do something was yesterday, the next best time is today, don't wait till tomorrow.
    2. Hard skill to master but will serve you well throughout your career.
- **Study groups**: Virtual or in-person, they're a great way to keep yourself and your peers accountable. Teaching your peers is a good way to make sure you understand the foundational concepts.
- **Leverage resources**: Go to TA/instructor office hours *regularly* and not just before the tests/midterms/finals.

Special Considerations Policy

- Missing an assessment due to extraordinary circumstances?
  Send a form and supporting documentation to the course email.
- Acceptable reasons:
  - Late course enrollment
  - Medical conditions: physical/mental health, hospitalizations, injury, accidents
  - Non-medical conditions (i.e., family/personal emergency)
- Unacceptable reasons: heavy course loads, multiple assignments/tests during the same period, time management issues
- Accessibility students: Accommodations are listed in Accessibility documentation

Remark Requests

- A marking error on assignment/test.
- Submit within two weeks after marks are released.
- For assignment, submit on MarkUs.
  For midterm, fill out a form and send it to course email.

Course Information

Course Website: Almost Everything.
`http://www.cs.toronto.edu/~rahulgk/courses/csc311_f23/index.html`

Quercus: Announcements and Grades.

Piazza: Discussions.
`https://piazza.com/utoronto.ca/fall2023/csc3112023fall`

MarkUs: Assignments.
`https://markus.teach.cs.toronto.edu/2023-09`

CrowdMark: Exams.

Getting in Touch

Piazza:
`https://piazza.com/utoronto.ca/fall2023/csc3112023fall`

- Course related and no sensitive info → public post
- Course related and sensitive info → private post
- TAs will respond within 48 hours. Ask early!

Course email: csc311-2023-09@cs.toronto.edu

- Special considerations requests.
- Remark requests for midterm.
- Any other matter.

Only email us directly with non-CSC311 questions.

- Course-related questions will get a faster response through the course email instead of emailing us individually.

Office Hours

Instructors' Office Hours

- Rahul G. Krishnan, Tuesdays Wednesdays, 9:30- 11:00 am at PT 286,
- Instructor OH will prioritize conceptual questions about the course material.

TAs will hold office hours to help with assignments and the project, as well as preparing for the midterm and final exams.

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.

Types of Machine Learning

- **Supervised learning:** have labeled examples of the correct behavior
- **Unsupervised learning:** no labeled examples – instead, looking for "interesting" patterns in the data
- **Reinforcement learning:** learning system (agent) interacts with the "world" and learns to maximize a scalar reward signal

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

1. Introductions

2. Admin Details

3. What is Machine Learning?
    - Examples of Machine Learning
    - Why This Class?

4. Preliminaries and Nearest Neighbor Methods

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).

Source: https://www.youtube.com/watch?v=YGO2lwAgrig

Instance segmentation - ▸ Link

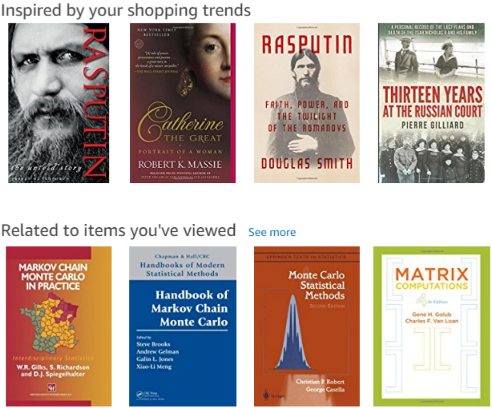
DAQUAR 1553
What is there in front of the sofa?
Ground truth: table
IMG+BOW: table (0.74)
2-VIS+BLSTM: table (0.88)
LSTM: chair (0.47)

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)

Natural Language Processing

Machine translation, sentiment analysis, topic modeling, spam filtering, general purpose chatbots (ChatGPT/GPT4).

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



Inspired by your shopping trends

Related to items you've viewed

See more

1. Introductions

2. Admin Details

3. What is Machine Learning?
    - Examples of Machine Learning
    - Why This Class?

4. Preliminaries and Nearest Neighbor Methods

Related Courses

- Was CSC411 previously.
- CSC412 (Probabilistic Learning and Reasoning) and CSC413 (Neural Networks and Deep Learning) build upon this course.
- Overlap with Applied Statistics course.

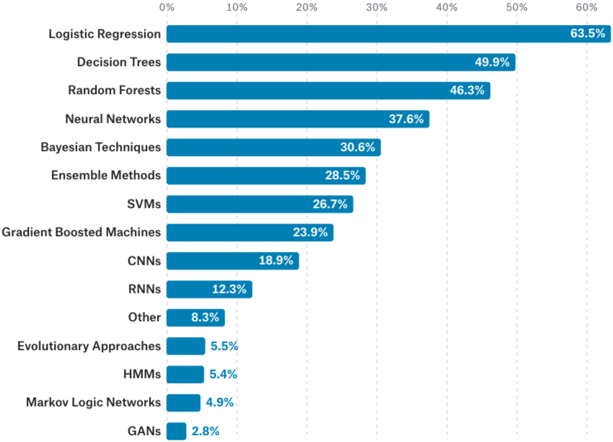
Why This Class?

Why not jump straight to CSC 412/413, 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 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.

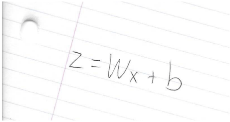
Why This Class?

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

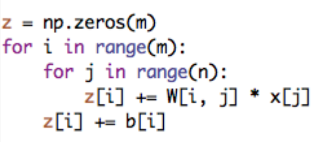


Implementing Machine Learning Systems

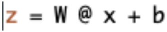
- Derive an algorithm (with pencil and paper), and then translate the math into code.
- Array processing (NumPy)
  - **vectorize** computations (express them in terms of matrix/vector operations) to exploit hardware efficiency
  - This also makes your code cleaner and more readable!



$Z = Wx + b$



```
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!

1. Introductions
2. Admin Details
3. What is Machine Learning?
4. Preliminaries and Nearest Neighbor Methods

Introduction

$$D = \{ \ldots (\overset{\text{input}}{x_i}, \overset{\text{output.}}{t_i}) \ldots \}_{i=1}^{N}$$

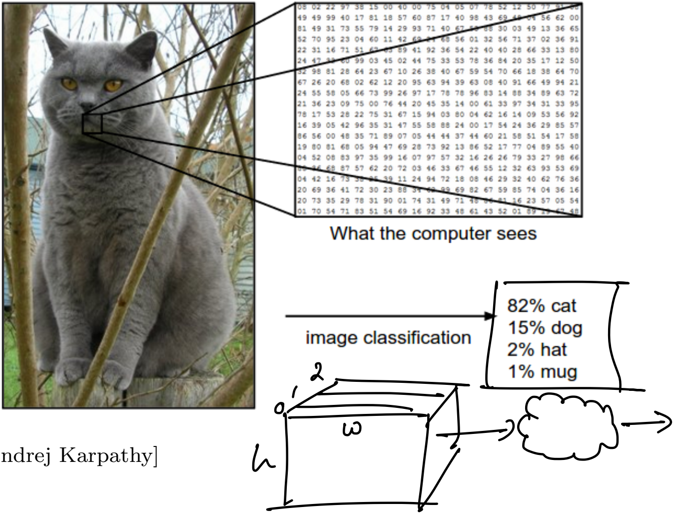
$$\left.\begin{array}{l} t_i / y_i \sim \text{output} \\ x_i \sim \text{input} \end{array}\right\} \text{Notation}$$

- 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 $\{1 \ldots K\}$
image captioning	image	caption
document classification	text	document category
speech-to-text	audio waveform	text $\{0, 1\}$
⋮	⋮	⋮

Input Vectors

What an image looks like to the computer:



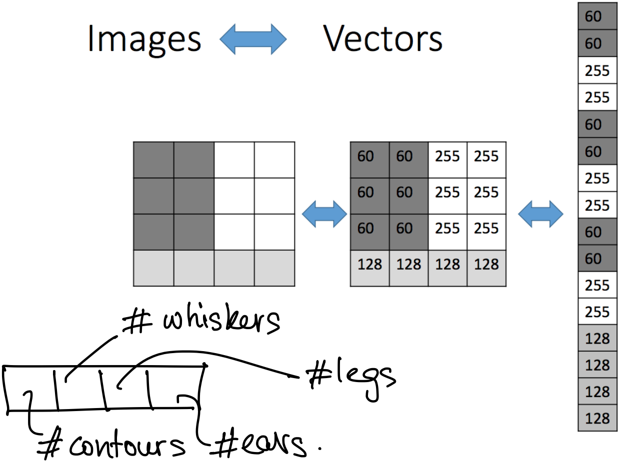
[Image credit: Andrej Karpathy]

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:



60	60	255	255
60	60	255	255
60	60	255	255
128	128	128	128

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 $\{1, \dots, C\}$
    - 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!

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:

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

2. Output $y = t^*$

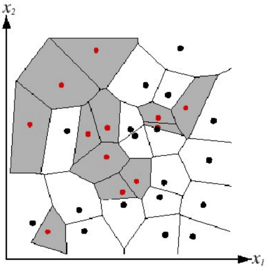
- Note: we don't need to compute the square root. Why?

Nearest Neighbors: Decision Boundaries

red black .

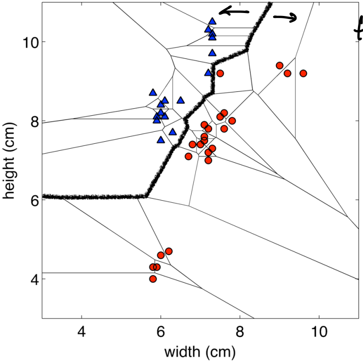
$\{0,1\}$.

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



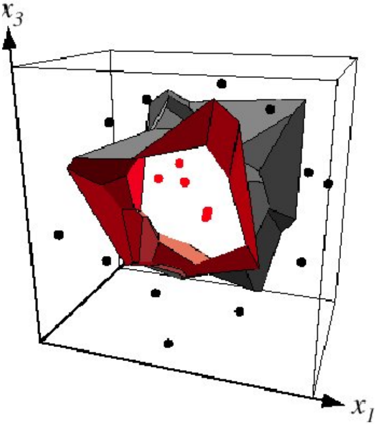
Nearest Neighbors: Decision Boundaries

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



classify depending on whether a test point is on the left or right of this boundary

Nearest Neighbors: Decision Boundaries

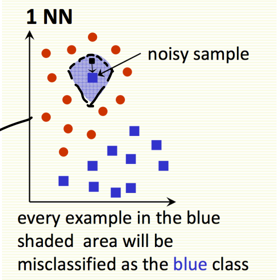


Example: 2D decision boundary

Nearest Neighbors

- Sensitive to noise or mis-labeled data ("class noise"). Solution?

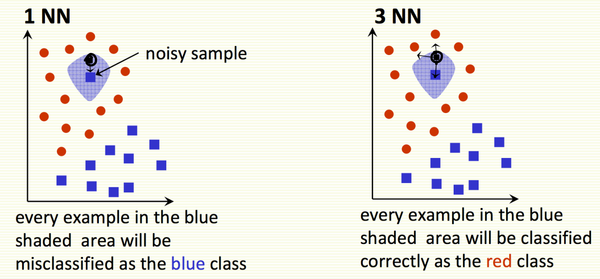
Dtr



[Pic by Olga Veksler]

Nearest Neighbors

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



[Pic by Olga Veksler]

K-Nearest Neighbors

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

Algorithm (kNN):

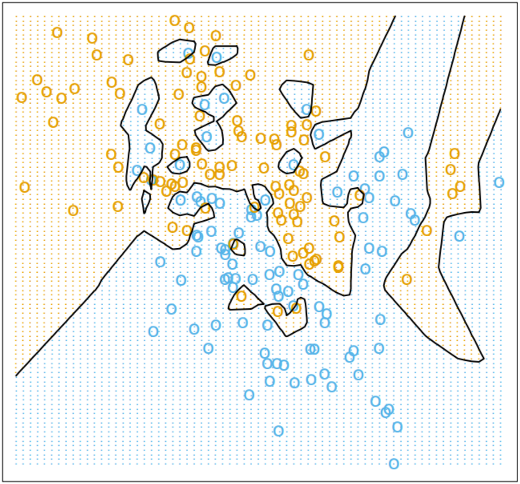
1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class.

$$y^* = \max_{t^{(z)} \in \text{class labels}} \sum_{i=1}^{k} \mathbb{I}(t^{(z)} = t^{(i)})$$

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

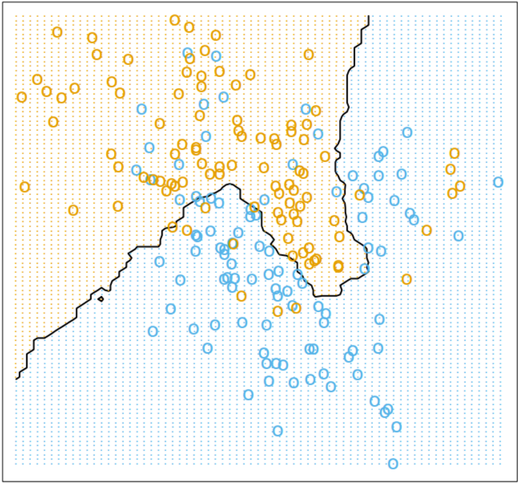
k=1



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

K-Nearest Neighbors

k=15



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

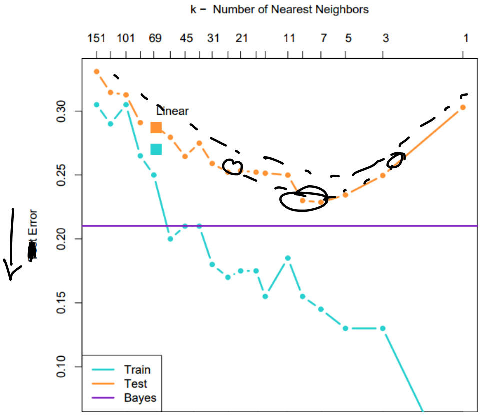
K-Nearest Neighbors

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

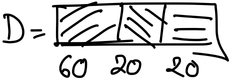
- We would like our algorithm to generalize to data it hasn't seen before.
- We can measure the generalization error (error rate on new examples) using a test set.



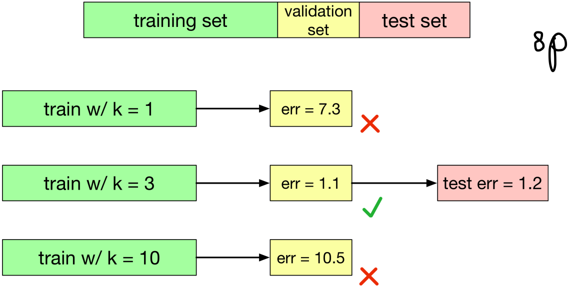
[Image credit: "The Elements of Statistical Learning"]

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:



is a typical split .



training set | validation set | test set

train w/ k = 1 → err = 7.3 ✗

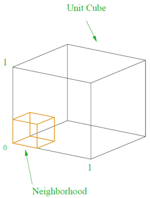
train w/ k = 3 → err = 1.1 ✓ → test err = 1.2

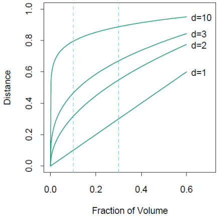
train w/ k = 10 → err = 10.5 ✗

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

Pitfalls: The Curse of Dimensionality

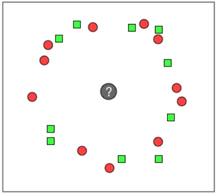
- Low-dimensional visualizations are misleading! In high dimensions, "most" points are far apart.
- Excercise: We want the nearest neighbor of *any* query x to be closer than ϵ. How many points do we in our space to guarantee it? circle $O(\varepsilon^2)$
- The volume of a single ball of radius ϵ around each point is e.g $\mathcal{O}(\epsilon^d)$ sphere $O(\varepsilon^3)$
- 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.
- If $\epsilon = 0.1$, each increase of dimension means we need to have 10x more balls to cover the volume.





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. (Will show this in a homework question...)
- Picture to keep in mind:

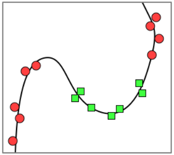


Pitfalls: The Curse of Dimensionality

- Nearest neighbors says that if two points are close in input space, their outputs must be close (the same).
- As dimension increases, all points appear equidistance – so how do we select the label for a test point?
- As dimension increases, so too do the number of irrelevant dimensions that nearest neighbor will compute distances based on.

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.



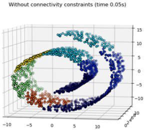
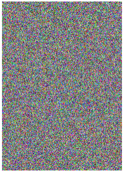
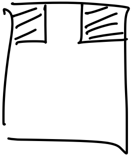


Image credit: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html

- 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.

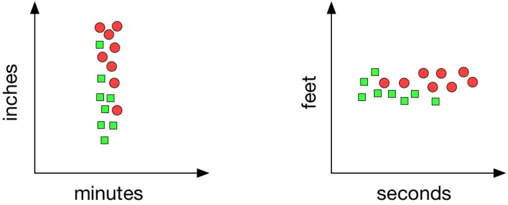
$2^{(h \times w)}$





Pitfalls: Normalization

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



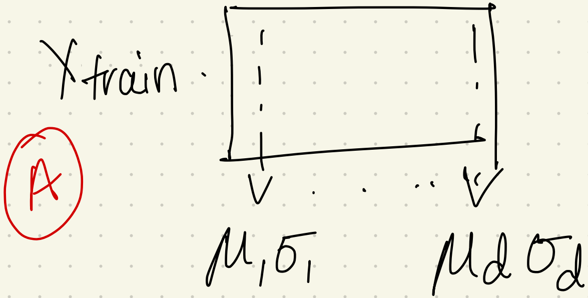
- Simple fix: normalize each dimension to be zero mean and unit variance. I.e., compute the mean μ_j and standard deviation σ_j, and take

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

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

Steps for normalization

Ⓐ



- Calculate d means & standard deviations (one for each feature)

Modify training set :

Ⓑ for each row, dimension j in training set do

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j} \quad ①$$

Ⓒ Need to apply the same transformation in ① to test points before classification

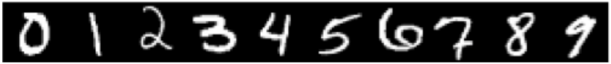
Pitfalls: Computational Cost

- 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

x_i 28x28

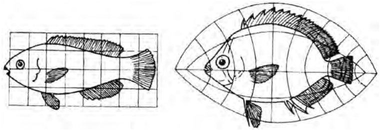


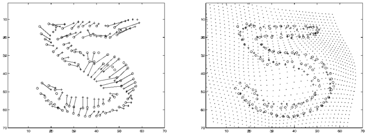
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: d = 784
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	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.]

Announcements

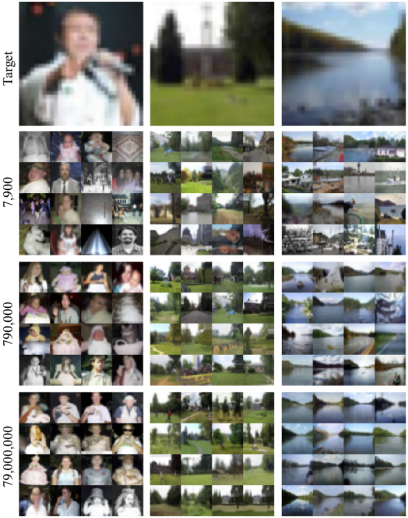
① Math diagnostics returned in ~1.5 weeks.

② HW 1 out today / tomorrow → Due in ~2 weeks.

TA OH will be posted

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.]

Learning with Generative AI

Here are some prompts (tried out on GPT4) to revise some of the content you learned in lecture today.

- Explain the k-nearest neighbor algorithm to me as if I were a (ten year old) or (first year undergraduate student) or (without using any mathematics) or (using formal notation).
- What is the curse of dimensionality in machine learning?
- Create pseudocode for the k-nearest neighbor algorithm. (After model output.) Go line by line and explain what each line does to me.

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.