

# CSC 2515: Introduction to Machine Learning

## Lecture 1: Introduction and K-Nearest Neighbours Method

Amir-massoud Farahmand<sup>1</sup>

University of Toronto and Vector Institute

---

<sup>1</sup>Credit for slides goes to many members of the ML Group at the U of T, and beyond, including (recent past): Roger Grosse, Murat Erdogdu, Richard Zemel, Juan Felipe Carrasquilla, Emad Andrews, and myself.

# Table of Contents

## 1 Introduction

- Examples

## 2 K-Nearest Neighbour Methods

- Input Vectors
- K-Nearest Neighbours Method
- Training, Validation, and Test Sets
- The Curse of Dimensionality

## 3 Logistics

# How can we make an intelligent agent?

- What does it mean to have an intelligent agent?
- What do we need to create it?

# An AI Agent



Figure: An agent ...

# An AI Agent



Figure: ... observes the world ...

# An AI Agent

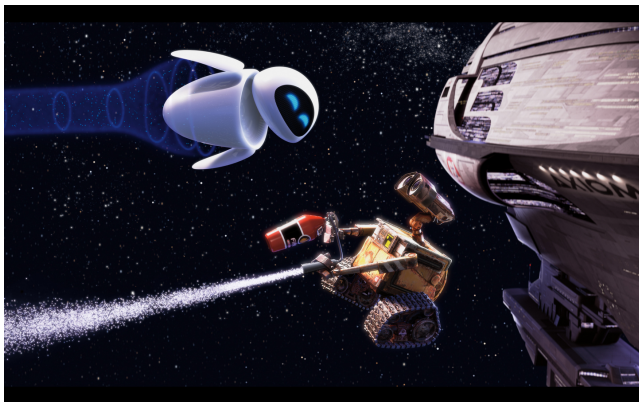


Figure: ... takes an action and its states changes ...

# An AI Agent

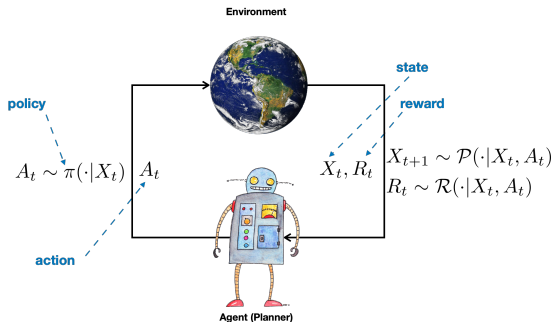


Figure: ... with the goal of achieving long-term rewards.

# An AI Agent: Some Requirements

This agent has to

- **predict** how the world works, e.g.,
  - ▶ classify different objects
  - ▶ estimate the probability of certain events happening in the future
- **plan** its actions in order to achieve its long-term goals





# 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

- Nowadays, “machine learning” is often brought up with “artificial intelligence” (AI)
- AI does not necessarily imply a learning-based system
  - ▶ Symbolic reasoning
  - ▶ Rule based system
  - ▶ Tree search
  - ▶ etc.

# 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

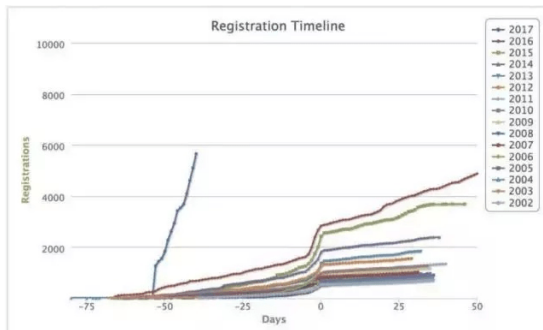
# History of machine learning

- 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



# History of machine learning

ML conferences selling out like Beyonce tickets.



Source: [medium.com/syncedreview/](https://medium.com/syncedreview/)

# History of machine learning

ML conferences selling out like Beyonce tickets.



 **NIPS**  
@NipsConference [Follow](#)

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

9:17 AM - 4 Sep 2018

678 Retweets 999 Likes

77 678 999

The image shows a tweet from the NIPS (@NipsConference) account. The tweet text reads: "#NIPS2018 The main conference sold out in 11 minutes 38 seconds". The tweet is dated "9:17 AM - 4 Sep 2018". Below the text, it shows "678 Retweets" and "999 Likes". At the bottom of the tweet, there are icons for replies (77), retweets (678), likes (999), and a direct message icon.

# Applications in Computer Vision

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

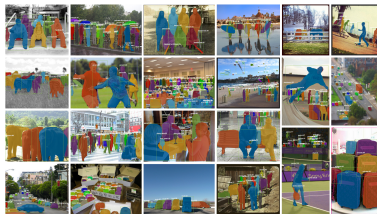


Figure 4. More results of Mask R-CNN on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).



15.7 fps

Source: <http://www.princeton.edu/~cs599v/YG00wA9ig>



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)

# Applications in Speech Recognition

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



# Applications in Natural Language Processing

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

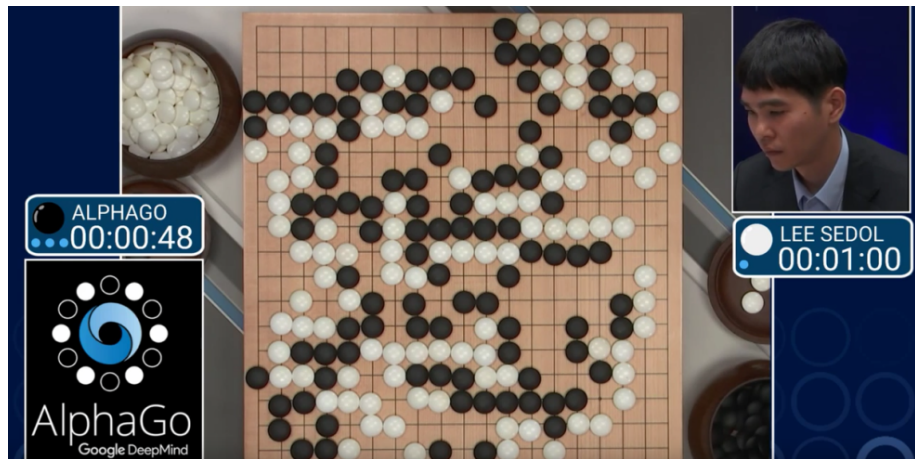
## Real world example:

*The New York Times*

LDA analysis of 1.8M New York Times articles:



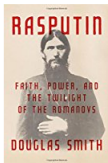
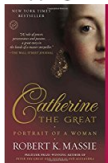
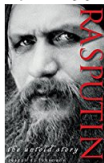
# Applications in Playing Games



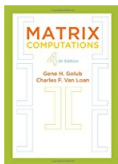
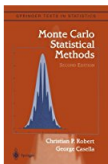
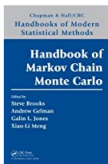
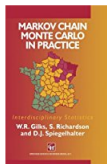
DOTA2 - [Link](#)

# Applications in E-commerce & Recommender Systems : Amazon, Netflix, ...

Inspired by your shopping trends



Related to items you've viewed [See more](#)



# Why this class?

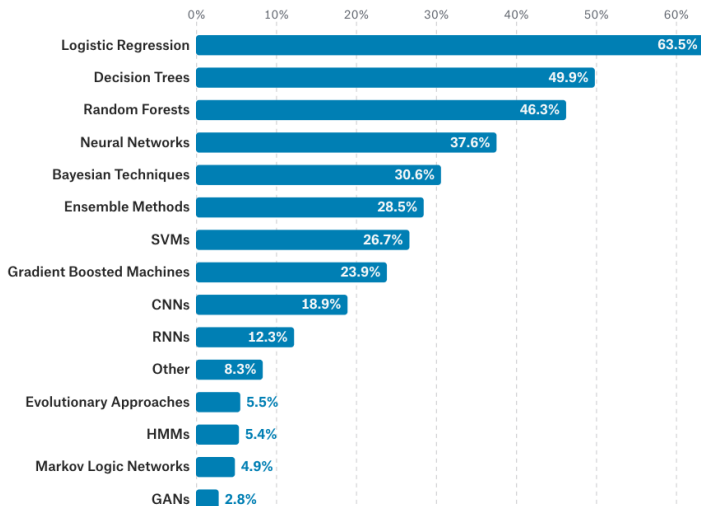
Why not jump straight to CSC413/2516 (Neural Networks), and learn neural nets first?

- 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?

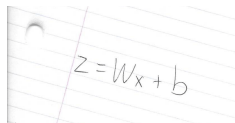


ML workflow sketch:

1. Should I use ML on this problem?
  - ▶ Is there a pattern to detect?
  - ▶ Can I solve it analytically?
  - ▶ Do I have data?
2. Gather and organize data.
  - ▶ Preprocessing, cleaning, visualizing.
3. Establishing a baseline.
4. Choosing a model, loss, regularization, ...
5. Optimization (could be simple, could be a PhD!).
6. Hyperparameter search.
7. Analyze performance and mistakes, and iterate back to step 4 (or 2).

# 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!



A photograph of a piece of lined paper with a hole punch on the left. The equation  $z = Wx + b$  is handwritten in black ink on the paper.

```
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
```

# Implementing machine learning systems

- Machine Learning framework: scikit-learn
- Neural net frameworks: PyTorch, TensorFlow, Jax-derivatives, 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 is why we derive things by hand in this class!

# K-Nearest Neighbour Methods

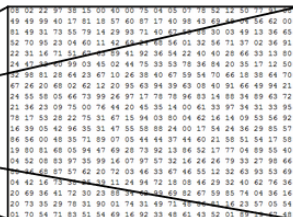
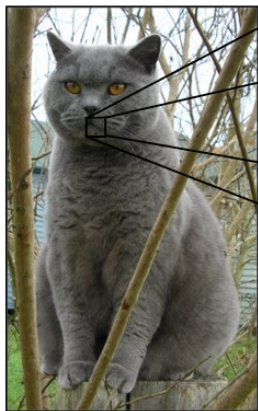
# Introduction

- Today (and for the next 5–6 weeks) we focus on **supervised learning**.
- This means we are given a **training set** consisting of **inputs** and corresponding **labels** (or **target, output**), 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:



What the computer sees

image classification

82% cat  
15% dog  
2% hat  
1% mug

[Image credit: Andrej Karpathy]

# Input Vectors

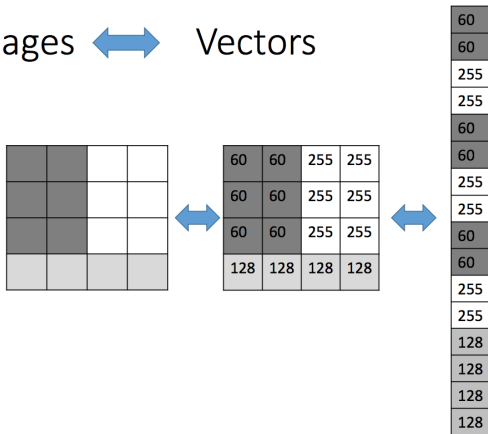
- Machine learning algorithms need to handle many different types of data: images, text, audio waveforms, credit card transactions, etc.
- Common strategy: represent the input as an **input vector** in  $\mathbb{R}^d$ 
  - ▶ **Representation**: mapping to another space that is easy to manipulate
  -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



# Input Vectors

Can use raw pixels:

Images  $\longleftrightarrow$  Vectors



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

# Dataset

- Mathematically, our training dataset 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 (ex: stock price, performance of a design)
  - ▶ **Classification**:  $t$  is an element of a discrete set  $\{1, \dots, C\}$  (ex: object recognition, health diagnosis given symptoms)
  - ▶ These days,  $t$  is often a highly structured object (ex: image, sentence, etc.)
- Denote the training set  $\{(\mathbf{x}^{(1)}, t^{(1)}), \dots, (\mathbf{x}^{(N)}, t^{(N)})\}$ 
  - ▶ Note: these superscripts have nothing to do with exponentiation!
- **Question**: Other examples of regression and classification problems?

# Developing our first ML algorithm

- Suppose that we are given the training set  $\{(\mathbf{x}^{(1)}, t^{(1)}), \dots, (\mathbf{x}^{(N)}, t^{(N)})\}$ , for a classification problem.
- Suppose we are given a novel input vector  $\mathbf{x}$ , and we are asked to classify it.
- **Question:** How can we do it?

# Nearest Neighbours

- Suppose that we are given a novel input vector  $\mathbf{x}$  we would 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:

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

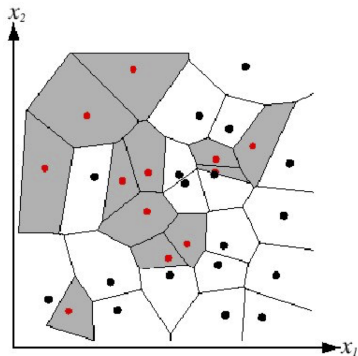
2. Output  $y = t^*$

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

# Nearest Neighbours: Decision Boundaries

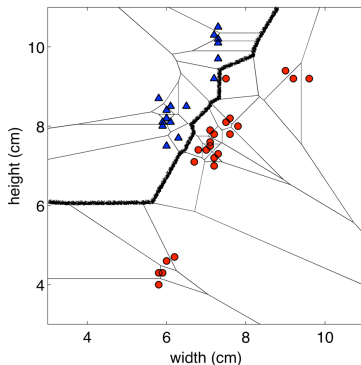
We can visualize the behaviour in the classification setting using a **Voronoi diagram**.

- Voronoi cell: The set of points in the space that are closest to a given seed point (a data point, in our case)

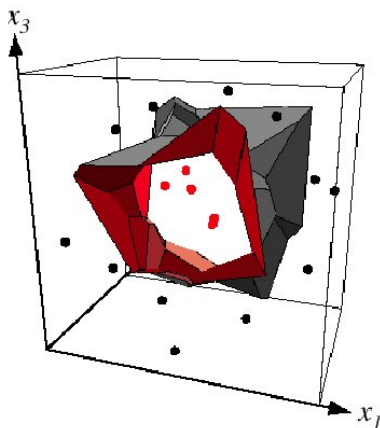


# Nearest Neighbours: Decision Boundaries

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



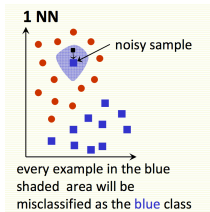
# Nearest Neighbours: Decision Boundaries



Example: 2D decision boundary

# Nearest Neighbours

[Pic by Olga Veksler]

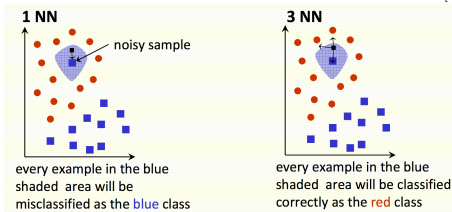


- Nearest neighbours are sensitive to noise or mis-labeled data (“class noise”). Solution?



# K-Nearest Neighbours

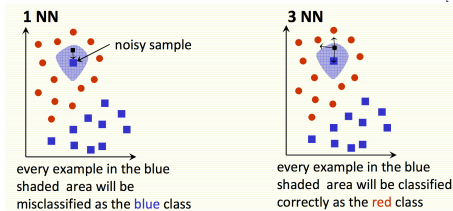
[Pic by Olga Veksler]



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

# K-Nearest Neighbours

[Pic by Olga Veksler]



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

## Algorithm (kNN):

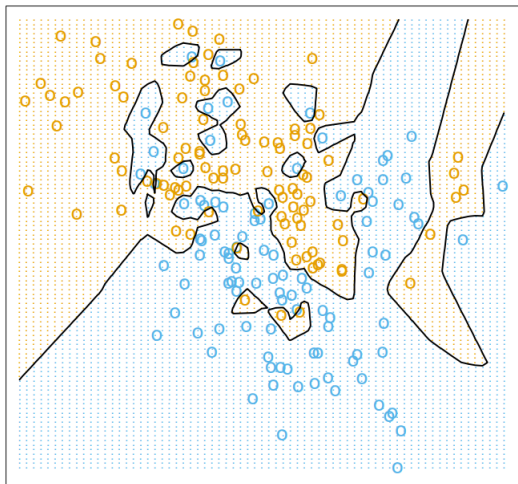
1. Find  $k$  examples  $\{\mathbf{x}^{(i)}, t^{(i)}\}$  closest to the test instance  $\mathbf{x}$
2. Classification output is majority class

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

$\mathbb{I}\{\text{statement}\}$  is the identity function and is equal to one (1) 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 Neighbours

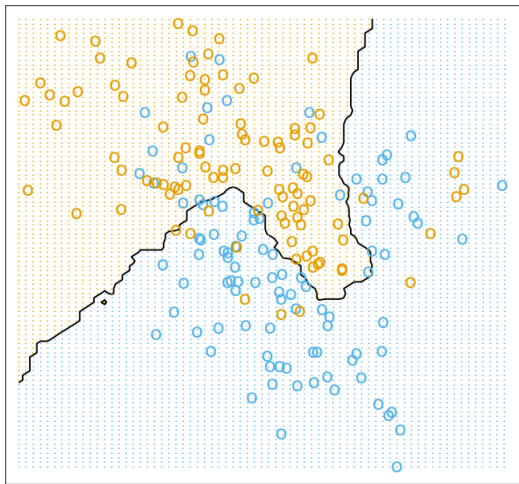
$k = 1$



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

# K-Nearest Neighbours

$k = 15$



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

# K-Nearest Neighbours

## Tradeoffs in choosing $k$ ?

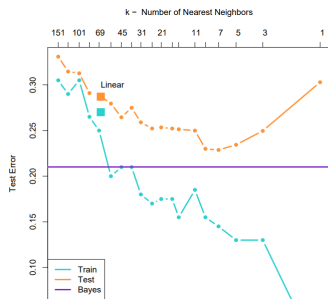
- Small  $k$ 
  - ▶ Good at capturing fine-grained patterns
  - ▶ May **overfit**, i.e. be sensitive to random idiosyncrasies in the training data
- Large  $k$ 
  - ▶ Makes stable predictions by averaging over lots of examples
  - ▶ May **underfit**, i.e. fail to capture important regularities
- Balancing  $k$ :
  - ▶ The optimal choice of  $k$  depends on the number of data points  $n$ .
  - ▶ Nice theoretical properties if  $k \rightarrow \infty$  and  $\frac{k}{n} \rightarrow 0$ .
  - ▶ Rule of thumb: Choose  $k = n^{\frac{2}{2+d}}$ .
  - ▶ We explain an easier way to choose  $k$  using data.

# Generalization

- Learning is most meaningful if the agent can solve new problems that it has not seen before. Memorization alone may not be enough.
- What matters for an ML algorithm is that it **generalizes** well on data that it has not seen before.

# Training and Test Sets

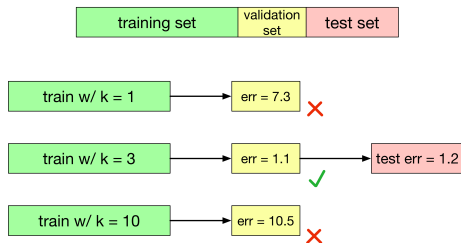
- We use a data set to train an ML method. It is called **training set**.
- We cannot use the same training set to measure how well the method generalizes. The algorithm may merely memorize the training set, but may not be able to generalize to new inputs.
- To measure the **generalization error** (error rate on new examples), we use another independent data set called **test set**.



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

# Training, Validation, and Test Sets

- We often also need another data set to adjust the **hyperparameters** of an algorithm: parameters that do not fit as a part of the learning process itself. Example: the value of  $k$  in the K-NN algorithm.
- We can tune hyperparameters using a **validation set**:



- We use the test set only at the very end to measure the generalization performance of the final configuration. We should not let any information about the test set leak into the solution that we find.



# Why do we need independent training, validation, and test sets?

Let us forget K-NN for a while, and focus on a much simpler problem:

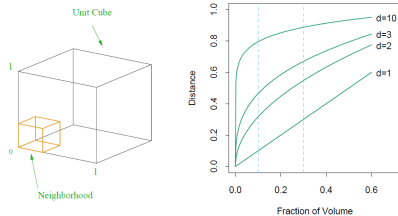
- Suppose we are given real-valued random variables (r.v.)  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}$ , all from the same distribution  $\mu$ .
- We would like to compute the expectation of random variable  $\mathbf{x} \sim \mu$  using the given data, i.e.,  $\mathbb{E}_{\mathbf{x} \sim \mu} [\mathbf{x}]$ .
- **Question:** How to do it?
- **Question:** How to measure the quality of our estimate? (focus on  $N = 1$ ).

# Pitfalls: The Curse of Dimensionality

- Low-dimensional visualizations are misleading! In high dimensions, “most” points are far apart.
- If we want the nearest neighbour to be closer than  $\epsilon$ , how many points do we need to guarantee it?

# Pitfalls: The Curse of Dimensionality

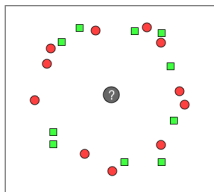
- Low-dimensional visualizations are misleading! In high dimensions, “most” points are far apart.
- If we want the nearest neighbour to be closer than  $\epsilon$ , how many points do we need to guarantee it?
- The volume of a single ball of radius  $\epsilon$  is  $\mathcal{O}(\epsilon^d)$
- The total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



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

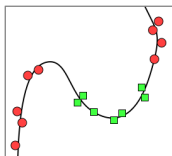
# Pitfalls: The Curse of Dimensionality

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



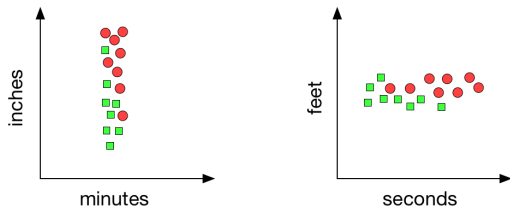
# Pitfalls: The Curse of Dimensionality

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



# Pitfalls: Normalization

- Nearest neighbours 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  $\mu_j$  and standard deviation  $\sigma_j$ , and take

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

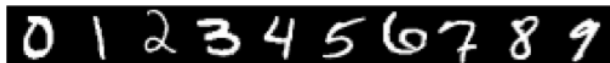
- Caution: depending on the problem, the scale might be important!

# 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 neighbours with high dimensions and/or large datasets.

# Example: Digit Classification

- Decent performance when lots of data



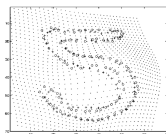
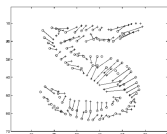
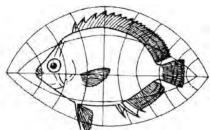
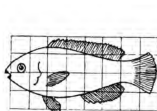
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images:  $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	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

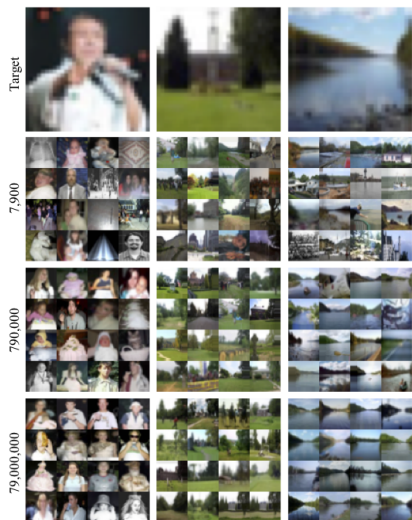
- KNN can perform a lot better with a good similarity measure.
- Example: shape contexts for object recognition. In order to achieve invariance to image transformations, they tried to warp one image to match the other image.
  - ▶ Distance measure: average distance between corresponding points on *warped* 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  $32 \times 32$ .
- 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.



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

# Conclusions

- Simple algorithm that does all its work at test time.
- Can be used for regression too, which we encounter later.
- Can control the complexity by varying  $k$
- Suffers from the Curse of Dimensionality
- Not very explainable.
- Next time: decision trees, another approach to regression and classification

Questions?

?

# Logistics

# This Course

- Broad introduction to machine learning
  - ▶ **Supervised learning:** nearest neighbours, decision trees, ensembles, linear regression, logistic regression, SVM, neural networks
  - ▶ **Unsupervised learning:** PCA, K-means, mixture models
  - ▶ **Reinforcement learning:** Value Iteration and Q-Learning
- Coursework is aimed at advanced undergrads. We will use multivariate calculus, probability, and linear algebra.
- Do I have the appropriate background?
  - ▶ **Linear algebra:** vector/matrix manipulations, properties.
  - ▶ **Calculus:** partial derivatives/gradient.
  - ▶ **Probability:** common distributions; Bayes Rule.
  - ▶ **Statistics:** expectation, variance, covariance, median; maximum likelihood.

- Course Website:  
<https://amfarahmand.github.io/IntroML-Fall2022/>
- Main source of information is the course webpage. Check regularly!
- We will also use Quercus for **announcements**.
- We will use Piazza for **discussions**.
- We will use MarkUs for **assignment submission**.

- We have two sessions per week.
  - ▶ Tuesdays: Lectures (in-person)
  - ▶ Thursday: Tutorial or Lectures (in-person or via Zoom)



# Course Information

- Pandemic is not completely over! Some of your friends and peers might have weak immune system. Long COVID can be serious.
- Highly encourage you to wear mask all the time in the classroom and during in-person office hours.
- Do not sit close to each other, if possible.
- Get vaccinated!
- While cell phones and other electronics are not prohibited in lecture, talking, recording or taking pictures in class is strictly prohibited without the consent of your instructor. Please ask before doing!
- **YouTube videos** from last year are available. Use them for review of material, or if you have to miss a class.

# Course Information

- Please refer to <http://www.illnessverification.utoronto.ca> in case of illness (you need to fill out an absence declaration form on ACORN and contact me).
- If you require additional academic accommodations, please contact UofT Accessibility Services: <https://studentlife.utoronto.ca/department/accessibility-services/>
- I realize that this is an unusually difficult time for all of us. I try to be as accommodating as possible.

# Course Evaluation

This is tentative and may change in the next few days:

- Three (3) assignments (40%)
  - ▶ Combination of mathematical derivations, proofs, and programming exercises.
- Research Project (30%)
  - ▶ Research proposal (5%), written report (20-25%), and (possibly) class presentation (0-5%)
- Take-Home Test (10%)
- Questions & Answers (10%)
- Read some seminal papers. (10%)
  - ▶ Short 1-paragraph summary and two questions on how the method(s) can be used or extended.
- Bonus (5%)
  - ▶ Class participation, finding typos in the slides, evaluating the class, etc

# Collaboration and Assignments

## Collaboration:

- Collaboration on the Homework Assignments **is allowed**, under certain conditions:
  - ▶ You can discuss the assignment with up to two other students (group of three).
  - ▶ You can work on the code together.
  - ▶ Each of you need to write down your homework individually. And mention the name of your collaborators clearly in the report and the source code.
- You need to form a team of 3–4 members to work on your projects (the exact number will be determined after finalizing the number of students enrolled).
- Collaboration on the Take-home test and Questions are **not allowed**.

## Late Submissions (assignments, proposals, reports, etc):

- Submissions 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 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 **CSC413/2516** (Neural Networks and Deep Learning), **CSC412/2506** (Probabilistic Learning and Reasoning), and Introduction to Reinforcement Learning both build upon the material in this course.
- CSC2515 and CSC311 (previously CSC 411) have a long history at the University of Toronto. Each year, its content has been added, removed, or revised by different instructors. We liberally borrow from the previous editions.
- Credit goes to many professors, including (recent past): Roger Grosse, Murat Erdogdu, Richard Zemel, Juan Felipe Carrasquilla, Emad Andrews, etc.
- **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.

# Useful Resources

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.
- Amir-massoud Farahmand, [Lecture Notes on Reinforcement Learning](#), 2021.
- 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.