STA 314: Statistical Methods for Machine Learning I

Lecture 1 - Introduction and Nearest Neighbours

Chris J. Maddison[1]

University of Toronto

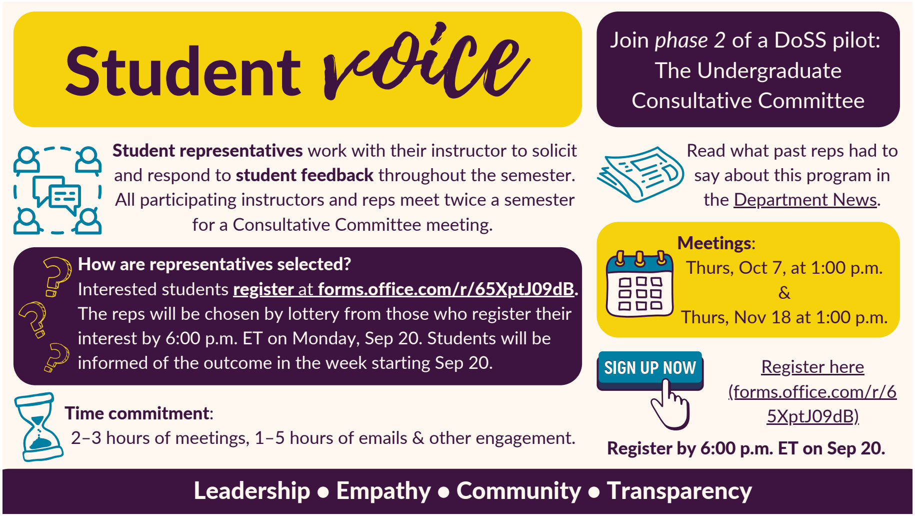
[1]Slides adapted from CSC 311.

Are you happy with fully online tutorials?

I sent out a survey to measure preferences for in-person tutorials. Please take a moment to fill it out:

`https://forms.office.com/r/mzZ06k9Dfa`

You need to sign into your UofT account to fill it out.



Student *voice*

Join *phase 2* of a DoSS pilot: The Undergraduate Consultative Committee

Student representatives work with their instructor to solicit and respond to **student feedback** throughout the semester. All participating instructors and reps meet twice a semester for a Consultative Committee meeting.

Read what past reps had to say about this program in the Department News.

How are representatives selected?
Interested students **register at forms.office.com/r/65XptJ09dB.** The reps will be chosen by lottery from those who register their interest by 6:00 p.m. ET on Monday, Sep 20. Students will be informed of the outcome in the week starting Sep 20.

Meetings:
Thurs, Oct 7, at 1:00 p.m.
&
Thurs, Nov 18 at 1:00 p.m.

Time commitment:
2–3 hours of meetings, 1–5 hours of emails & other engagement.

SIGN UP NOW

Register here (forms.office.com/r/65XptJ09dB)

Register by 6:00 p.m. ET on Sep 20.

Leadership ● Empathy ● Community ● Transparency

This course

- This course is a broad introduction to machine learning.
- We cover two major types:
    - Supervised learning
        - nearest neighbours, decision trees, ensembles, linear regression, logistic regression, SVMs
    - Unsupervised learning
        - PCA, K-means, probabilistic models
- We cover a variety of important methods:
    - maximum likelihood, optimization with gradient descent, Bayesian inference
- 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.

Do I have the appropriate background?

- We are using the Python programming language in this course.
- How much do you need to know?
    - The emphasis will be on the use of the numerical computing package NumPy (tutorial 2) and on the implementation of the key subroutines of ML methods.
    - You will **not need to write an entire Python package**. We will provide you with very complete starter code.
    - You will **need to confidently modify / complete the body of a Python function** to make the code perform the algorithm. correctly.
- Why not R?
    - I don't know R.
    - The machine learning community mostly uses Python.
    - Follow-up courses like STA414 typically use Python.

Course Information

Most information: website is main source of information; check regularly!

`https://www.cs.toronto.edu/~cmaddis/courses/sta314_f21/`

Announcements, grades, & links: Quercus.

- Did you receive the announcement?

Discussions: Piazza.

- Sign up: `https://piazza.com/utoronto.ca/fall2021/sta314`
- Your grade does not depend on your participation on Piazza. It's a good way for asking questions, discussing with the course community. Only discuss course materials/assignments/exams, but **do not give homework hints**.

Course Information

Daily class schedule: All sections (LEC0101 and LEC0201) have the same basic schedule.

Class Component	Delivery Method
2hr Lecture	Zoom
1hr Office Hour	Zoom
1hr Tutorial	Zoom / in-person

More detailed delivery instructions on the next slide.

Course Information

Delivery instructions:

- All lectures and office hours are synchronous via Zoom.
- Tutorials are synchronous via Zoom and **potentially** in-person.
  - **First 2 weeks, all tutorials are synchronous via Zoom.**
  - Afterwards, we will determine the demand and appropriateness for in-person tutorials.
  - Whether your tutorial is in-person or online depends on your enrollment (see website for locations).
  - If you are enrolled in an in-person tutorial, but prefer online, you can join the online tutorial for your section.
  - **Do not attend an in-person tutorial unless you are enrolled in one.**
- Office hours are **not mandatory**.
- Week 5 will be fully asynchronous, because of Thanksgiving.

Course Information

- Lectures will be delivered synchronously via Zoom, and recorded for asynchronous viewing by enrolled students. All information about attending virtual lectures, tutorials, and office hours will be sent to enrolled students through Quercus.
- 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`. There is a volunteer note-taker in the course, if you need this service. Check Quercus for details.

Course Information

All that information is in the STA314 syllabus and on the website. If you remember just one thing:

Check Quercus and website regularly.

Suggested Readings

Suggested readings will be given for each lecture. These are **completely optional**, but useful. The following will be useful throughout the course:

- Hastie, Tibshirani, and Friedman. *The Elements of Statistical Learning*.
- Christopher Bishop. *Pattern Recognition and Machine Learning*.
- Kevin Murphy. *Machine Learning: a Probabilistic Perspective*.
- David MacKay. *Information Theory, Inference, and Learning Algorithms*.
- Shai Shalev-Shwartz & Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*.
- David Barber. *Bayesian Reasoning and Machine Learning*.

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

Marking

- (60%) 4 assignments
  - Combination of pen & paper derivations and light-weight programming exercises.
  - Weighted equally.
  - Hand-in on Quercus.
- (40%) Two 1-hour tests held during normal class time
  - See website for times and dates.
  - Weighted equally.
  - Taken on Quercus.

More on Assignments

Students may work on the assignments alone or in pairs. The marking scheme will not be adapted depending on the size of the team. If you choose to work in a pair, then you:

- Must register the pair with the instructur before the due date.
- Must include a contributions statement in your submission that describes the contribution of each team member.

You must not share proofs, pseudocode, code, or simulation results with students that are not your teammates. Violation of this policy is an academic offence and will be investigated and reported as such.

Assignments should be handed in by deadline; a late penalty of 10% of the total credit of the assignment 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 most one week after the due date.

What is the difference between this course and CSC311?

- In the past, STA314 and CSC311 have been taught differently, despite being exclusions.
    - This year, I am bringing them in line with each other.
- Here are major differences this year:
    - We do not cover neural networks nor reinforcement learning.
    - We cover the probabilistic interpretation in a bit more detail.
    - The emphasis in the homeworks is more on proofs and less on coding.
- In other words, **STA314 takes a more statistical perspective than CSC311** while covering the same core of material.

Advanced Courses

This course will help prepare you for the following courses.

- **STA414** (Statistical Methods for Machine Learning II)
  - This course is the follow-up course, which delves deeper into the probabilistic interpretation of machine learning that we cover in the last few weeks.
- **CSC413** (Neural Networks and Deep Learning)
  - This course covers deep learning and automatic differentiation.
- **CSC412** (Probabilistic Learning and Reasoning)
  - The CSC analogue of STA414.

Questions?

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 rigorous conclusions about data
    - ML is more concerned with making predictions that are very accurate
- and the communities are somewhat different...
    - 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.

Relations to AI

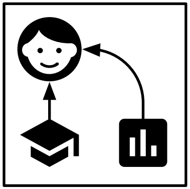
- Nowadays, “machine learning” is often brought up with “artificial intelligence” (AI)
- AI does not always imply a learning based system
    - Symbolic reasoning
    - Rule based system
    - Tree search
    - etc.
- Learning based system $\rightarrow$ learned based on the data $\rightarrow$ more flexibility,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

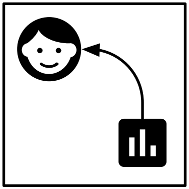
Types of machine learning

Supervised Learning



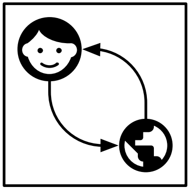
Machine is given data and examples of what to predict.

Unsupervised Learning



Machine is given data, but not what to predict.

Reinforcement Learning



Machine gets data by interacting with an environment and tries to minimize a cost.

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.

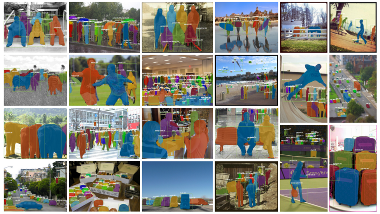
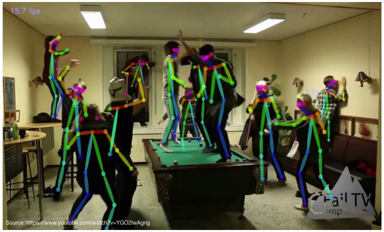
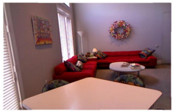


Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).



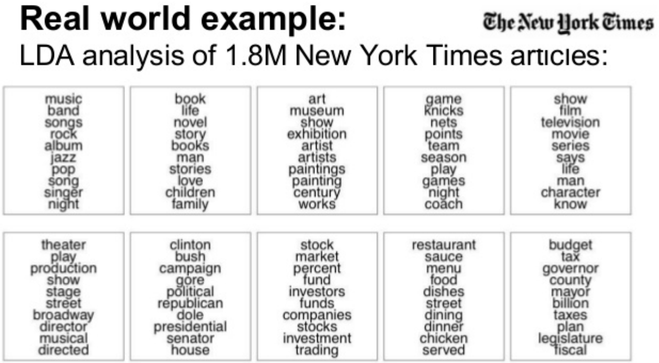

DAQUAR 1553
What is there in front of the sofa?
Ground truth: table
IMG+BOW: table (0.74)
2-VIS+BLSTM: table (0.88)
LSTM: chair (0.47)

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.



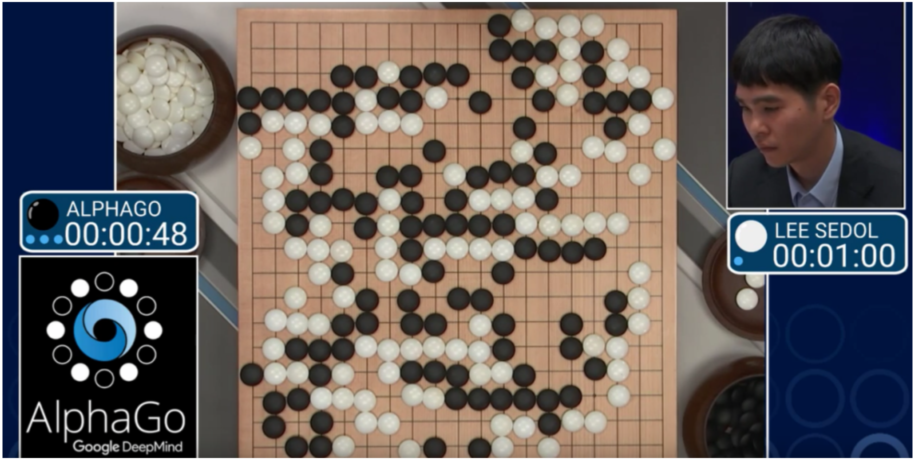
Real world example: The New York Times

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

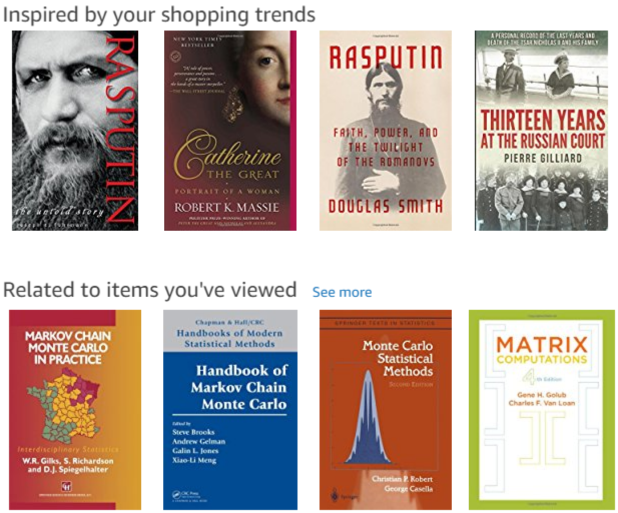
theater	clinton	stock	restaurant	budget
play	bush	market	sauce	tax
production	campaign	percent	menu	governor
show	gore	fund	food	county
stage	political	investors	dishes	mayor
street	republican	funds	street	billion
broadway	dole	companies	dining	taxes
director	presidential	stocks	dinner	plan
musical	senator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

Playing Games



DOTA2 - ▸ Link

Recommender Systems : Amazon, Netflix, ...



Inspired by your shopping trends

RASPUTIN
the untold story
Catherine the Great
Portrait of a Woman
Robert K. Massie
Rasputin
Faith, Power, and the Twilight of the Romanovs
Douglas Smith
Thirteen Years at the Russian Court
Pierre Gilliard

Related to items you've viewed See more

Markov Chain Monte Carlo in Practice
Interdisciplinary Statistics
W.R. Gilks, S. Richardson and D.J. Spiegelhalter
Chapman & Hall/CRC Handbooks of Modern Statistical Methods
Handbook of Markov Chain Monte Carlo
Steve Brooks
Andrew Gelman
Galin L. Jones
Xiao-Li Meng
Springer Texts in Statistics
Monte Carlo Statistical Methods
Second Edition
Christian P. Robert
George Casella
Matrix Computations
4th Edition
Gene H. Golub
Charles F. Van Loan

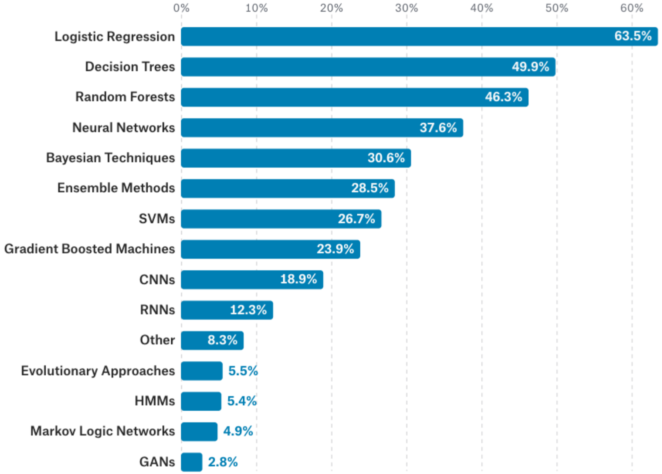
Why this class?

“I’ve heard that neural networks solve everything, can we just learn those?”

- 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 (STA414).

Why this class?

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



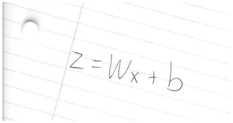
ML Workflow

ML workflow sketch:

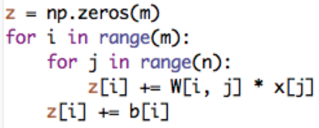
1. Should I use ML on this problem?
   - Is there a pattern to detect?
   - Can I solve it analytically?
   - Do I have data?
2. Gather and organize data.
   - Preprocessing, cleaning, visualizing.
3. Establishing a baseline, i.e., implement the simplest, default ML method.
4. Choose and tailor an ML method to your problem.
5. Analyze performance & mistakes, and iterate.

Implementing machine learning systems

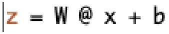
- A key part of this workflow involves modifying existing methods to make them fit your setting.
- This will often involve deriving an algorithm (with pencil and paper), and then translating the math into code.
- One of the key ideas in machine learning is array processing or vectorized computations, i.e., express the algorithm in terms of matrix/vector operations to exploit hardware efficiency (more in next week's tutorial on NumPy).



$z = Wx + b$



```
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

- There are now very convenient and powerful 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.
- For many tasks of interest we are given some data and are interested in predicting something about that data.

Task	Input data	We wish to predict
object recognition	image	object category
image captioning	image	caption
document classification	text	document category
speech-to-text	audio waveform	text
⋮	⋮	⋮

- Supervised learning is applicable when we have many examples of good predictions, i.e., we can supervise the learner by telling it exactly what to predict.

Introduction

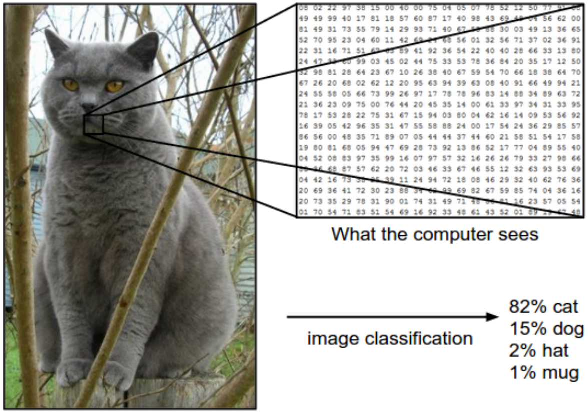
More precisely, in supervised learning, we are given

- training set consisting of
- inputs x and corresponding
- labels t.

Our goal is to predict the label or learn the mapping from $x \rightarrow t$. Let's unpack this carefully.

Input Vectors

Let's consider image input data. What an image looks like to the computer:



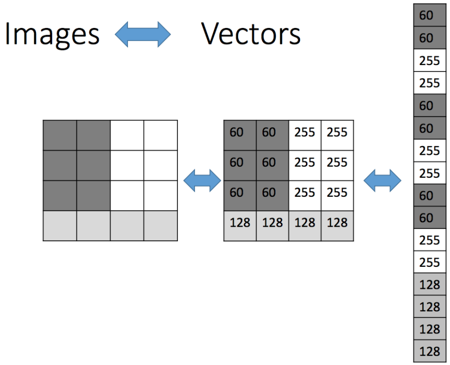
[Image credit: Andrej Karpathy]

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.

Labels

- You can think of labels as answers to questions about the data.
  - e.g., what is the ambient temperature in the picture?
  - e.g., what is the primary object in the image?
- We can use numbers to represent labels as well.
- Traditionally, we use different names depending on the type of label t.
  - Regression: t is a real number, e.g., the ambient temperature
  - Classification: $t \in \{1, \ldots, C\}$ is an element of a discrete set, e.g., let 1 mean "dog", 2 mean "cat", etc.
  - Structured prediction: these days, t is often a highly structured object (e.g., image can also be a label)

Training sets

- To summarize, mathematically, our training set consists of a collection of pairs of an input $\boldsymbol{x} \in \mathbb{R}^d$ and its corresponding label t.
- Denote the training set $\{(\boldsymbol{x}^{(1)}, t^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, t^{(N)})\}$
  - Note: these superscripts have nothing to do with exponentiation!
- Our goal is to learn a mapping from $\boldsymbol{x}^{(i)} \rightarrow t^{(i)}$ that performs well (will be more precise about this later).

Nearest Neighbors

- Suppose we're given a novel input vector $\boldsymbol{x}$ we'd like to classify.
- The idea: find the nearest input vector to $\boldsymbol{x}$ in the training set and copy its label.
- Can formalize "nearest" in terms of Euclidean distance

$$||\boldsymbol{x}^{(a)} - \boldsymbol{x}^{(b)}||_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

Algorithm:

1. Find example $(\boldsymbol{x}^*, t^*)$ (from the stored training set) closest to $\boldsymbol{x}$. That is:

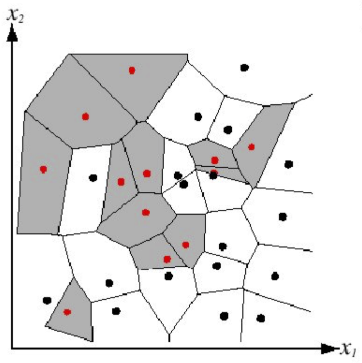
$$\boldsymbol{x}^* = \arg \min_{\boldsymbol{x}^{(i)} \in \text{train. set}} \text{distance}(\boldsymbol{x}^{(i)}, \boldsymbol{x})$$

2. Output $y = t^*$

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

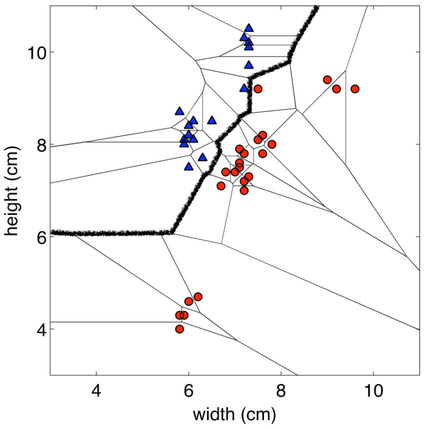
Nearest Neighbors: Decision Boundaries

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

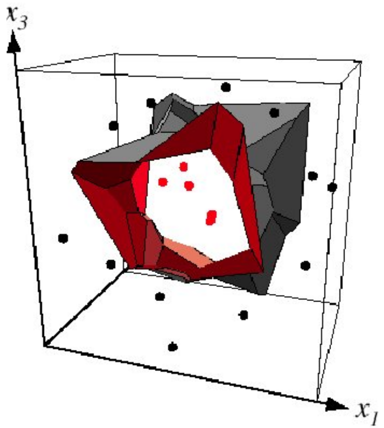


Nearest Neighbors: Decision Boundaries

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

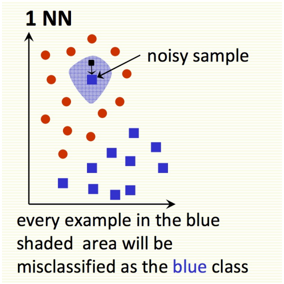


Nearest Neighbors: Decision Boundaries



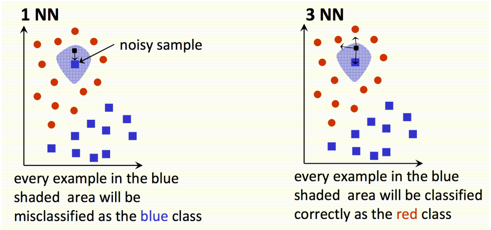
Example: 2D decision boundary

Nearest Neighbors


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? Smooth by having k nearest neighbors vote

k-Nearest Neighbors



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

Algorithm (kNN):

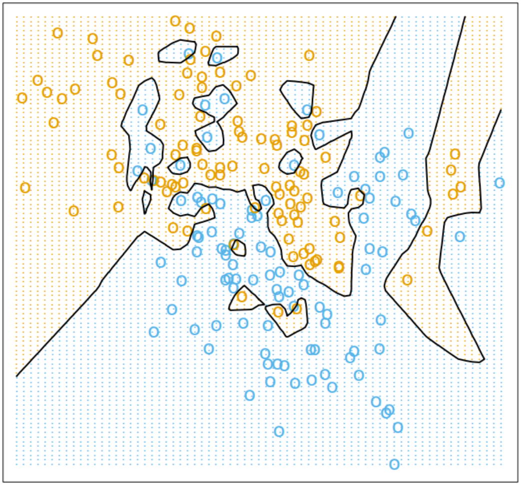
1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

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

$\mathbb{I}\{\text{statement}\} = 1$ when the statement is true, and 0 otherwise.

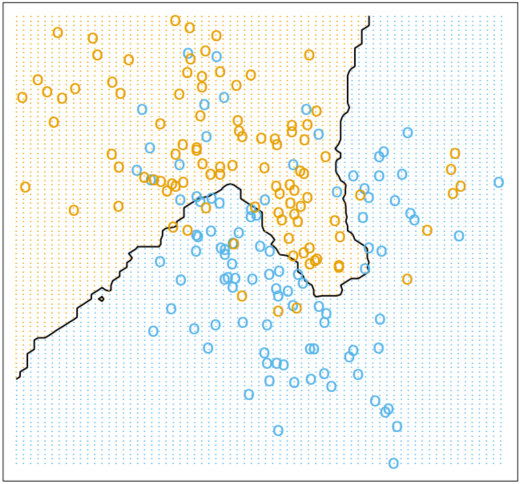
k-NN

k=1



k-NN

k=15



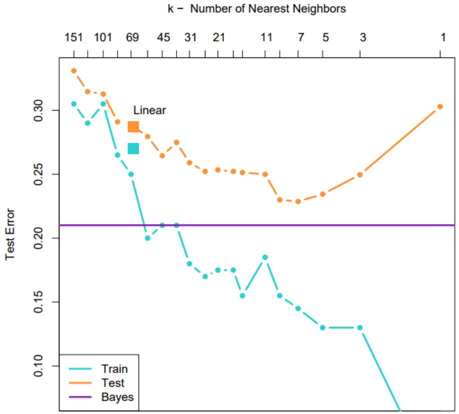
k-NN

Tradeoffs in choosing k?

- Small k
    - Good at capturing fine-grained patterns
    - May be sensitive to random idiosyncrasies in the training data, we call this overfitting.
- Large k
    - Makes stable predictions by averaging over lots of examples
    - May fail to capture important regularities, we call this underfitting.
- Balancing k
    - Optimal choice of k depends on number of data points n.
    - Nice theoretical properties if $k \to \infty$ and $\frac{k}{n} \to 0$ (ESL 2.4).
    - Rule of thumb: choose $k < \sqrt{n}$.

Generalization error for *k*-NN

- We would like our algorithm to generalize to data it hasn't seen before.
- How can we measure the generalization error (error rate on new examples)?
- Use a new, unseen set of input-label pairs called a test set.

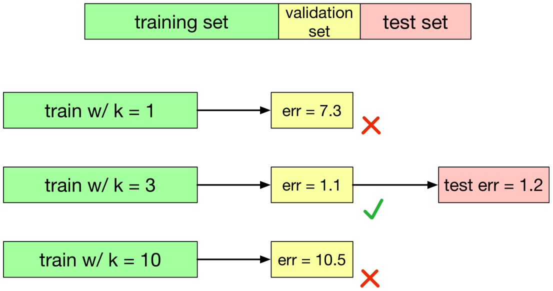


Hyperparameters

- Maybe we can use the test set to pick k? However, once we use the test set of pick k, it is no longer an unbiased way of measuring of how well we will do on *unseen* data.
- k is an example of a hyperparameter, something we can't fit as part of the learning algorithm itself.

Validation sets

- We can tune hyperparameters using a validation set, which is a third, unseen set of input-label pairs.



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

Pitfalls of k-NN: The Curse of Dimensionality

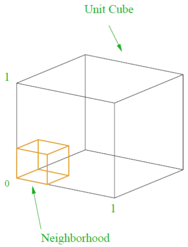
Low-dimensional visualizations are misleading!

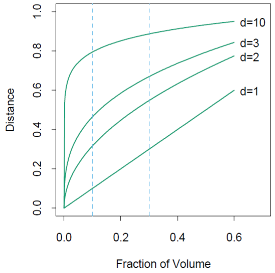
In high dimensions, “most” points are far apart.

Pitfalls of k-NN: The Curse of Dimensionality

Suppose we want we want the nearest neighbor of *every query* $x \in [0,1]^d$ to be closer than ϵ, how many points do we need in our training set to guarantee it?

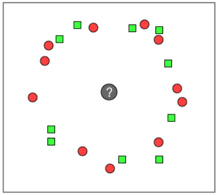
- The volume of a single ball of radius ϵ around each point is $\mathcal{O}(\epsilon^d)$
- The total volume of $[0,1]^d$ is 1.
- $\mathcal{O}\left(\left(\frac{1}{\epsilon}\right)^d\right)$ points are needed to cover the volume, i.e., increasing exponentially in d.





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.
- 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.

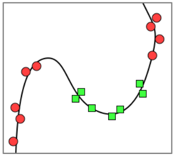
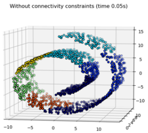
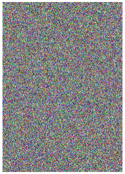



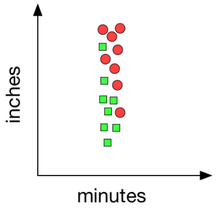
Image credit: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html`

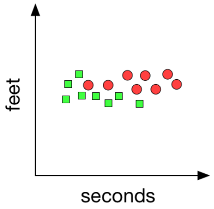
- The neighborhood structure (and hence the Curse of Dimensionality) depends on the intrinsic dimension.
- The space of megapixel images is 3 million-dimensional. The true number of degrees of freedom is much smaller.



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

$$\tilde{x}_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

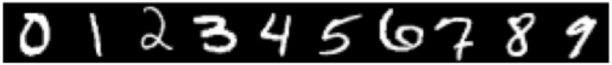
- Caution: depending on the problem, the scale might be important!

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

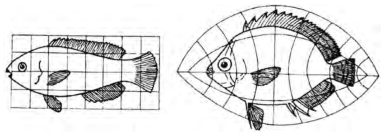


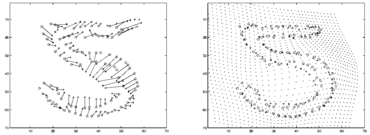
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	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

- Changing the similarity measure can really improve k-NN.
- 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 the state of the art at the time, but required careful engineering.





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

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?