CSC 2515: Machine Learning

Lecture 1 - Introduction and Nearest Neighbours

David Duvenaud

Based on Materials from Roger Grosse, University of Toronto

This course

- Broad introduction to machine learning
  - First half: algorithms and principles for supervised learning
    - nearest neighbors, decision trees, ensembles, linear regression, logistic regression, SVMs
    - neural nets!
  - Unsupervised learning: PCA, K-means, mixture models
  - Basics of reinforcement learning

This course

- Broad introduction to machine learning
  - First half: algorithms and principles for supervised learning
    - nearest neighbors, decision trees, ensembles, linear regression, logistic regression, SVMs
    - neural nets!
  - Unsupervised learning: PCA, K-means, mixture models
  - Basics of reinforcement learning
- This course is taught as a stand-alone grad course.
  - But the structure and difficulty will be similar to past years, when it was cross-listed as an undergrad course.
  - The majority of students are from outside Computer Science.

Course Information

Course Website: `https://subercui.github.io/csc2515/`

Slides will be posted to web page in advance of lecture, but I'll continue to make edits up to Thursday night. So please re-download!

1-hour Tutorials on Tuesdays at 10am and 10pm.

We will use Discourse for **discussions**.

- Sign up: `https://bb-2021-01.teach.cs.toronto.edu/c/csc2515h`
- Your grade **does not depend on your participation on the forum**. It's just a good way for asking questions, discussing with your instructor, TAs and your peers.

Course Information

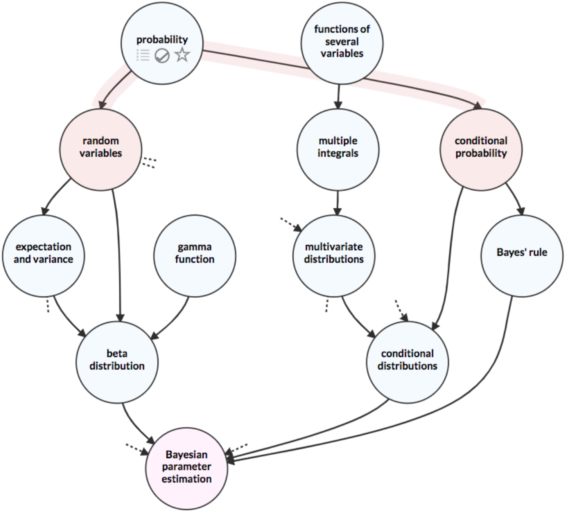
Recommended readings will be given for each lecture. But the following will be useful throughout the course:

- Hastie, Tibshirani, and Friedman: "The Elements of Statistical Learning"
- Christopher Bishop: "Pattern Recognition and Machine Learning", 2006.
- Kevin Murphy: "Machine Learning: a Probabilistic Perspective", 2012.
- David Mackay: "Information Theory, Inference, and Learning Algorithms", 2003.
- Shai Shalev-Shwartz & Shai Ben-David: "Understanding Machine Learning: From Theory to Algorithms", 2014.

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

Course Information

See Metacademy (`https://metacademy.org`) for additional background, and to help review prerequisites.



Requirements and Marking

- 4 assignments
  - Combination of pen & paper derivations and short programming exercises
  - Weights to be decided (expect 11% each), for a total of 44%
- Midterm
  - Date TBD, 12 hour window
  - Worth roughly 20% of course mark
- Final Project
  - Roughly last 3 weeks of course
  - Kaggle competition + extension
  - Worth roughly 36% of course mark

  You are free to explore something related to your research, but the project should be a distinct and isolated component – you must state which parts of the project were completed during the project timeframe. You may work in teams of up to 3 people. You will be required to complete a project report on your work in the form of an academic paper, as well as a final presentation.

More on Assignments

Collaboration After attempting the problems on an individual basis, you may discuss and work together on thehomework assignments with up to two classmates. However, you must write your own code and write up your own solutions individually and explicitly name all collaborators at the top of the homework.

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

Assignments should be handed in by deadline; a late penalty of 10% per day will be assessed thereafter (up to 4 days, then submission is blocked).

Related Courses

- More advanced ML courses such as **csc413** (neural nets) and **csc412** (probabilistic graphical models) both build upon the material in this course.
- If you've already taken an applied statistics course, there will be some overlap. Sorry.
- Might also consider grad topics courses after one of these advanced courses.

What is learning?

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

Merriam Webster dictionary

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

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

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?

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)

What is machine learning?

- 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

What is machine learning?

- 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
- But it's 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. "deep learning cowboys"
  - Stats puts more emphasis on the art of reasoning about particular datasets, ML more about generic automatic methods.

What is machine learning?

- Types of machine learning
  - **Supervised learning:** have labeled examples of the correct behavior
  - **Reinforcement learning:** learning system receives a reward signal, tries to learn to maximize the 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)

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

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

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

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

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

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

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

Instance segmentation - Link

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

COCOQA 5078
How many leftover donuts is the red bicycle holding?
Ground truth: three
IMG+BOW: two (0.51)
2-VIS+BLSTM: three (0.27)
BOW: one (0.29)

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

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

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

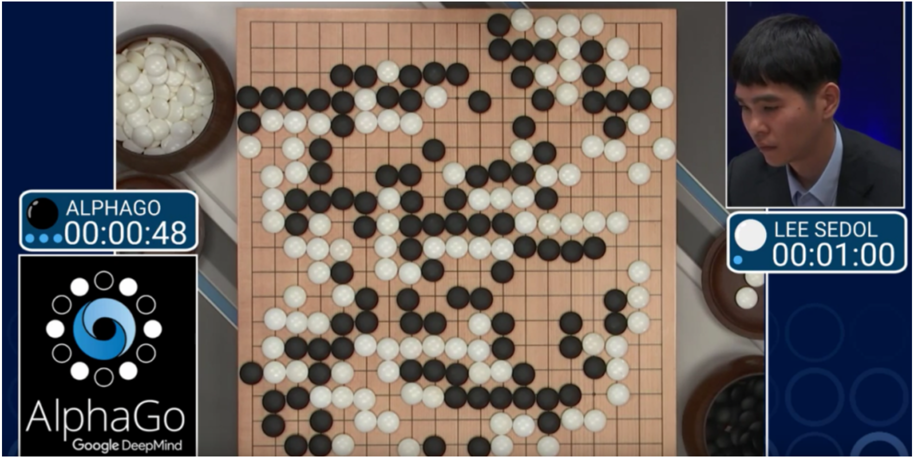
Real world example: The New York Times

LDA analysis of 1.8M New York Times articles:

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

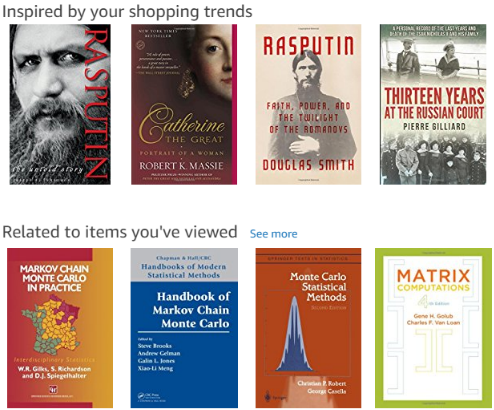
theater	clinton	stock	restaurant	budget
play	bush	market	sauce	tax
production	campaign	percent	menu	governor
show	gore	fund	food	county
stage	political	investors	dishes	mayor
street	republican	funds	street	billion
broadway	dole	companies	dining	taxes
director	presidential	stocks	dinner	plan
musical	senator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

Playing Games



DOTA2 - Link

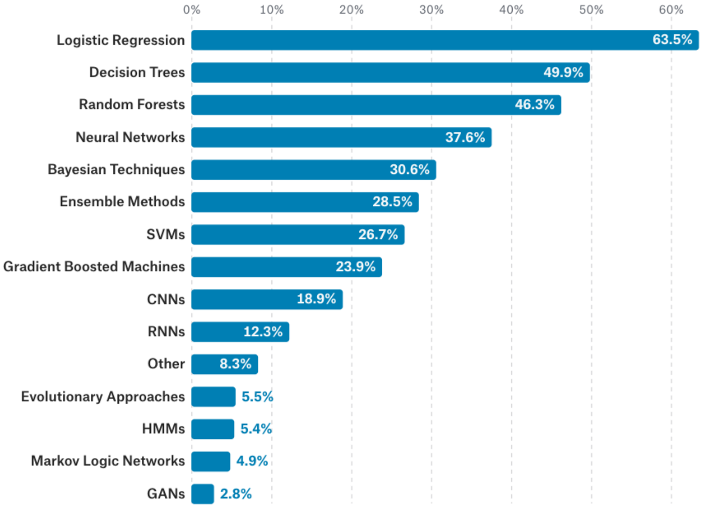
E-commerce & Recommender Systems : Amazon, netflix, ...



Next week (Tuesday at 3pm) you're invited to a guest lecture in JSC370 from the Amazon recommender systems team.

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?

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.

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.
3. Preprocessing, cleaning, visualizing.

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.
3. Preprocessing, cleaning, visualizing.
4. Establishing a baseline.

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.
3. Preprocessing, cleaning, visualizing.
4. Establishing a baseline.
5. Choosing a model, loss, regularization, ...

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.
3. Preprocessing, cleaning, visualizing.
4. Establishing a baseline.
5. Choosing a model, loss, regularization, ...
6. Optimization (could be simple, could be a Phd...).

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.
3. Preprocessing, cleaning, visualizing.
4. Establishing a baseline.
5. Choosing a model, loss, regularization, ...
6. Optimization (could be simple, could be a Phd...).
7. Hyperparameter search.

ML Workflow

ML workflow sketch:

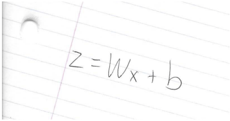
1. Should I use ML on this problem?
   - Is there a pattern to detect?
   - Can I solve it analytically?
   - Do I have data?
2. Gather and organize data.
3. Preprocessing, cleaning, visualizing.
4. Establishing a baseline.
5. Choosing a model, loss, regularization, ...
6. Optimization (could be simple,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

Implementing machine learning systems

- You will often need to derive an algorithm (with pencil and paper), and then translate the math into code.

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 shorter!




```python
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]
```

```python
z = np.dot(W, x) + b
```

Experimental new languages: Julia, Dex:
`https://github.com/google-research/dex-lang/`

Implementing machine learning systems

- Neural net frameworks: PyTorch, TensorFlow, etc.
  - automatic differentiation
  - compiling computation graphs
  - libraries of algorithms and network primitives
  - support for graphics processing units (GPUs)

Implementing machine learning systems

- Neural net frameworks: PyTorch, TensorFlow, 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!
  - Also so you can build custom models (more in CSC412)

Questions?

?

Nearest Neighbours

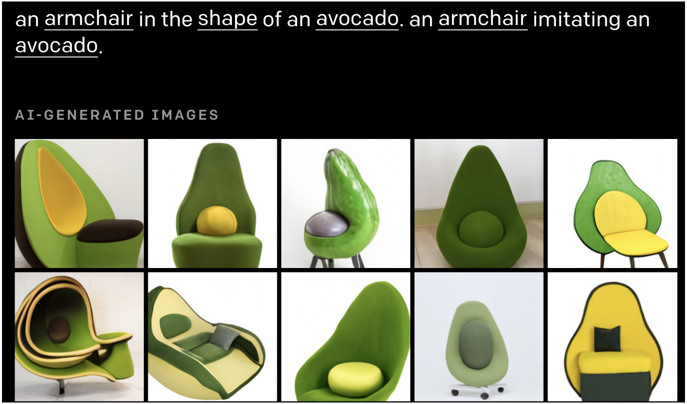
Introduction

- Today (and for the next 6 weeks) we're focused on supervised learning.
- This means we're given a training set consisting of inputs and corresponding labels, e.g.

Task	Inputs	Labels
object recognition	image	object category
image captioning	image	caption
document classification	text	document category
speech-to-text	audio waveform	text
text-to-image	description	image
$\vdots$	$\vdots$	$\vdots$

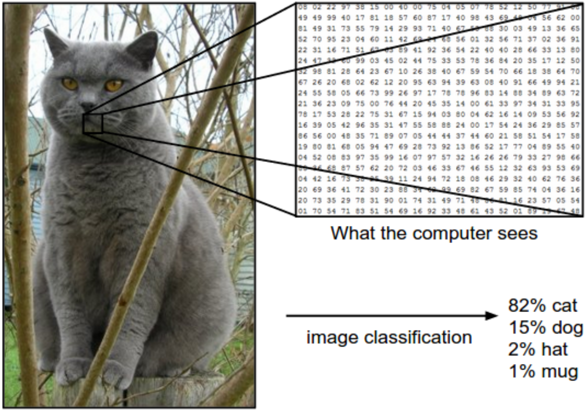
Text to Image

- Can now do plausible text-to-image in some settings. (Actually just models $p(\text{text}, \text{image})$ with a Markov model and then samples from $p(\text{image}|\text{text})$



Input Vectors

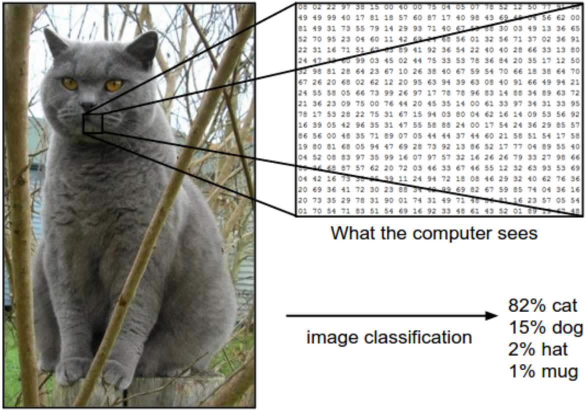
What an image looks like to the computer:



[Image credit: Andrej Karpathy]

Input Vectors

What an image looks like to the computer:



[Image credit: Andrej Karpathy]

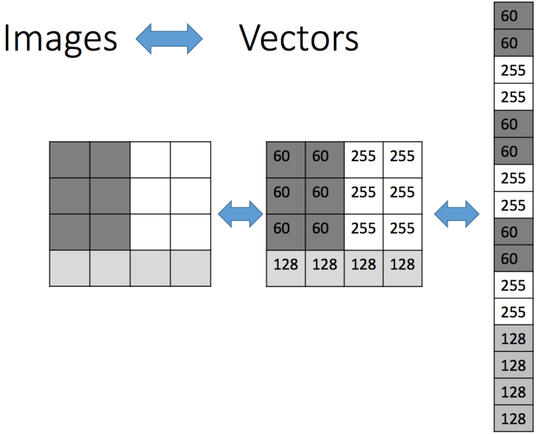
Input Vectors

- 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!

VECTOR INSTITUTE

Input Vectors

Can use raw pixels:



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

Input Vectors

- Mathematically, our training set consists of a collection of pairs of an input vector $\mathbf{x} \in \mathbb{R}^d$ and its corresponding target, or label, t
  - Regression: t is a real number (e.g. stock price)
  - Classification: t is an element of a discrete set $\{1, \ldots, C\}$
  - These days, t is often a highly structured object (e.g. image)
- Denote the training set $\{(\mathbf{x}^{(1)}, t^{(1)}), \ldots, (\mathbf{x}^{(N)}, t^{(N)})\}$
  - Note: these superscripts have nothing to do with exponentiation!

Nearest Neighbors

- Suppose we're given a novel input vector $\mathbf{x}$ we'd like to classify.
- The idea: find the nearest input vector to $\mathbf{x}$ in the training set and copy its label.

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

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

Nearest Neighbors

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

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

Algorithm:

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

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

2. Output $y = t^*$

Nearest Neighbors

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

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

Algorithm:

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

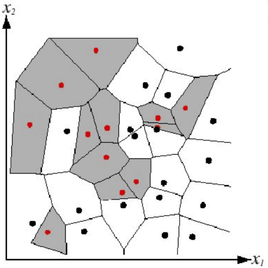
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\operatorname{argmin}} \ \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

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

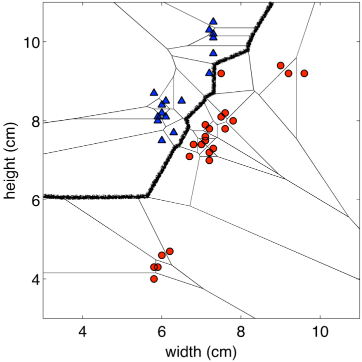
Nearest Neighbors: Decision Boundaries

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

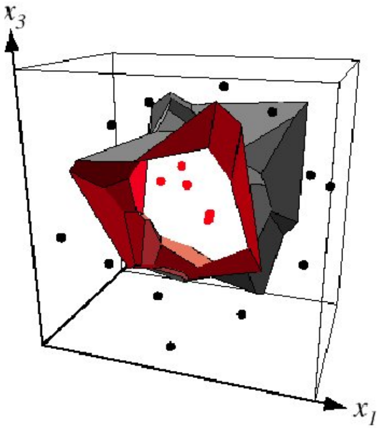


Nearest Neighbors: Decision Boundaries

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



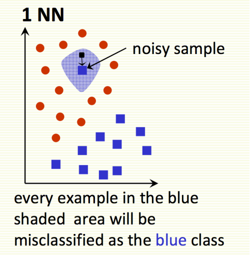
Nearest Neighbors: Decision Boundaries



Example: 3D decision boundary

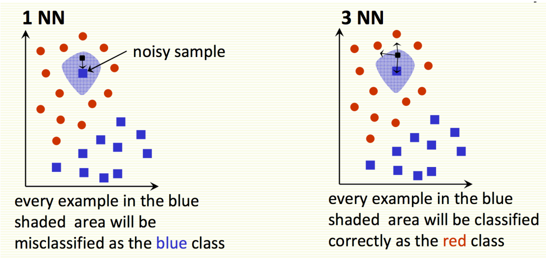
Nearest Neighbors

[Pic by Olga Veksler]



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

k-Nearest Neighbors

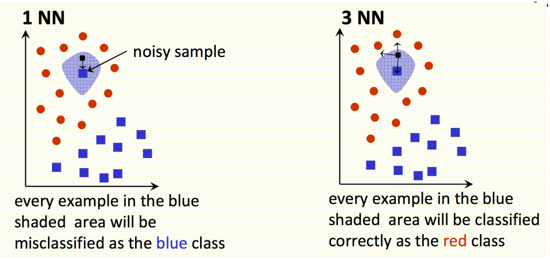


[Pic by Olga Veksler]

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

k-Nearest Neighbors

ic by Olga Veksler]



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

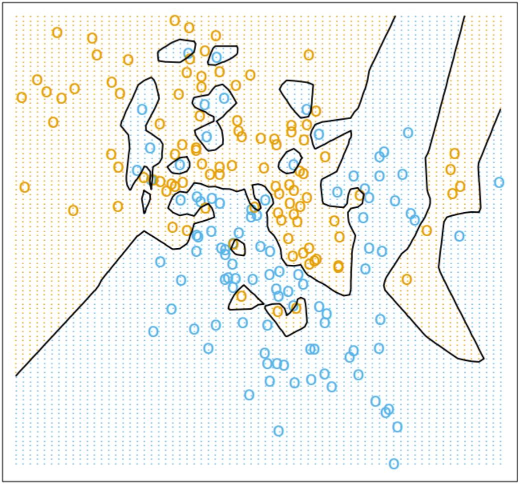
Algorithm (kNN):

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

$$y = arg \max_{t^{(z)}} \sum_{r=1}^{k} \delta(t^{(z)}, t^{(r)})$$

K-Nearest neighbors

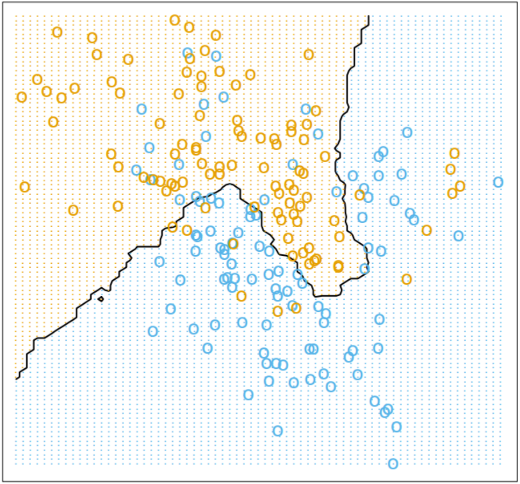
k=1



[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

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

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

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

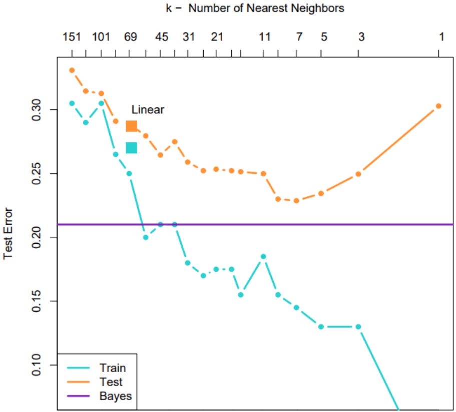
k-Nearest Neighbors

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
- Rule of thumb: $k < sqrt(n)$, where n is the number of training examples

K-Nearest neighbors

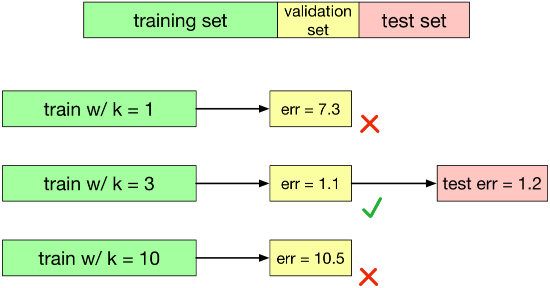
- We would like our algorithm to generalize to data it hasn't before.
- We can measure the generalization error (error rate on new examples) using a test set.



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

Validation and Test Sets

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



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

Consistency

- Is KNN consistent? I.e., given enough data, will it give the "right" answer?
- To analyze this, suppose the inputs $\mathbf{x}$ and targets t are random variables drawn independently and identically distributed (i.i.d.) from a data generating distribution with density $p(\mathbf{x}, t)$.

Consistency

- Is KNN consistent? I.e., given enough data, will it give the "right" answer?
- To analyze this, suppose the inputs $\mathbf{x}$ and targets t are random variables drawn independently and identically distributed (i.i.d.) from a data generating distribution with density $p(\mathbf{x}, t)$.
- The Bayes optimal classifier is the function $f(\mathbf{x})$ which minimizes the misclassification rate, i.e.

$$f(\mathbf{x}) = y_* = \arg\min_y \Pr(y \neq t \mid \mathbf{x}) = \arg\max_y \Pr(y = t \mid \mathbf{x}).$$

  Its error rate is called the Bayes error.
- Question: how close does KNN get to the Bayes error in the limit of infinite data?

Consistency

- Assume $p(\mathbf{x}, t)$ is smooth as a function of $\mathbf{x}$.
- Main idea: suppose N (the number of training examples) is very large, and consider a query point $\mathbf{x}_q$ which we'd like to classify.
  - By smoothness, $p(t \,|\, \mathbf{x})$ is approximately constant for nearby $\mathbf{x}$.
  - Hence, the labels of the neighbors can be seen as independent random variables with PMF $p(t \,|\, \mathbf{x}_q)$.

Consistency

- First consider $k = 1, N \to \infty$.

Consistency

- First consider $k = 1, N \to \infty$.
  - y (the nearest neighbour prediction) and t (the true label at $\mathbf{x}_q$) are (approximately) independent random variables with PMF $p(t \,|\, \mathbf{x}_q)$.
  - Apply the Union Bound:

$$\Pr(t \neq y \,|\, \mathbf{x}_q) \leq \Pr(t \neq y_* \,|\, \mathbf{x}_q) + \Pr(y_* \neq y \,|\, \mathbf{x}_q) = 2\Pr(t \neq y_* \,|\, \mathbf{x}_q).$$

  - I.e., the asymptotic error of 1-NN is at most twice the Bayes error.

Consistency

- First consider $k = 1, N \to \infty$.
  - y (the nearest neighbour prediction) and t (the true label at $\mathbf{x}_q$) are (approximately) independent random variables with PMF $p(t \,|\, \mathbf{x}_q)$.
  - Apply the Union Bound:

$$\Pr(t \neq y \,|\, \mathbf{x}_q) \leq \Pr(t \neq y_* \,|\, \mathbf{x}_q) + \Pr(y_* \neq y \,|\, \mathbf{x}_q) = 2\Pr(t \neq y_* \,|\, \mathbf{x}_q).$$

  - I.e., the asymptotic error of 1-NN is at most twice the Bayes error.
- Now consider $k, N \to \infty$ and $k/N \to 0$.

Consistency

- First consider $k = 1, N \to \infty$.
  - y (the nearest neighbour prediction) and t (the true label at $\mathbf{x}_q$) are (approximately) independent random variables with PMF $p(t \,|\, \mathbf{x}_q)$.
  - Apply the Union Bound:

$$\Pr(t \neq y \,|\, \mathbf{x}_q) \leq \Pr(t \neq y_* \,|\, \mathbf{x}_q) + \Pr(y_* \neq y \,|\, \mathbf{x}_q) = 2\Pr(t \neq y_* \,|\, \mathbf{x}_q).$$

  - I.e., the asymptotic error of 1-NN is at most twice the Bayes error.
- Now consider $k, N \to \infty$ and $k/N \to 0$.
  - The counts of neighbors' labels (approximately) follow a multinomial distribution with k trials.
  - For large k, the argmax will agree with the Bayes classifier with high probability. (E.g., apply the Central Limit Theorem.)
  - Hence, the KNN approaches the Bayes error, i.e. KNN is Bayes consistent.

Consistency

- First consider $k = 1, N \to \infty$.
  - y (the nearest neighbour prediction) and t (the true label at $\mathbf{x}_q$) are (approximately) independent random variables with PMF $p(t \,|\, \mathbf{x}_q)$.
  - Apply the Union Bound:

$$\Pr(t \neq y \,|\, \mathbf{x}_q) \leq \Pr(t \neq y_* \,|\, \mathbf{x}_q) + \Pr(y_* \neq y \,|\, \mathbf{x}_q) = 2\Pr(t \neq y_* \,|\, \mathbf{x}_q).$$

  - I.e., the asymptotic error of 1-NN is at most twice the Bayes error.
- Now consider $k, N \to \infty$ and $k/N \to 0$.
  - The counts of neighbors' labels (approximately) follow a multinomial distribution with k trials.
  - For large k, the argmax will agree with the Bayes classifier with high probability. (E.g., apply the Central Limit Theorem.)
  - Hence, the KNN approaches the Bayes error, i.e. KNN is Bayes consistent.
- Bayes consistency is a very special property, and holds for hardly any of the algorithms covered in this course.

Pitfalls: The Curse of Dimensionality

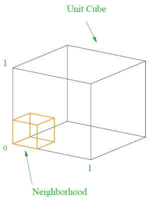
- Consistency is great, but it might take a very large amount of data to get close to the Bayes error.
  - Especially in high dimensions! KNN suffers from the Curse of Dimensionality.

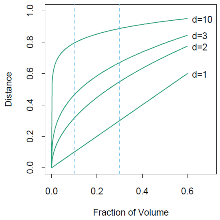
Pitfalls: The Curse of Dimensionality

- Consistency is great, but it might take a very large amount of data to get close to the Bayes error.
  - Especially in high dimensions! KNN suffers from the Curse of Dimensionality.
- How large does N need to be to guarantee we have an ϵ-neighbour?

Pitfalls: The Curse of Dimensionality

- Consistency is great, but it might take a very large amount of data to get close to the Bayes error.
  - Especially in high dimensions! KNN suffers from the Curse of Dimensionality.
- How large does N need to be to guarantee we have an ϵ-neighbour?
- The volume of a single ball of radius ϵ is $\mathcal{O}(\epsilon^d)$
- The total volume of $[0, 1]^d$ is 1.
- Therefore $\mathcal{O}\left((\frac{1}{\epsilon})^d\right)$ balls are needed to cover the volume.

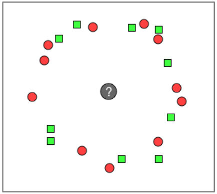




[Image credit: "The Elements of Statistical Learning"]

Pitfalls: The Curse of Dimensionality

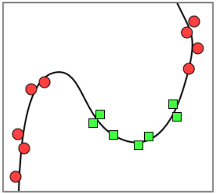
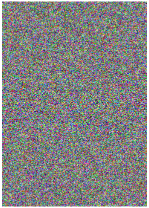
- Another perspective on the Curse of Dimensionality: in high dimensions, “most” points are approximately the same distance. (Homework question coming up...)



- This is just one example of how 2-D visualizations of high-dimensional spaces can be extremely misleading!

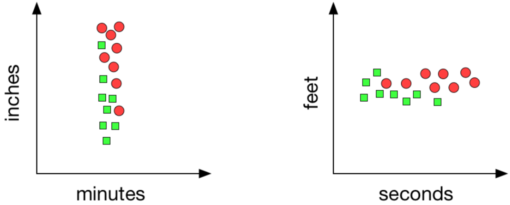
Pitfalls: The Curse of Dimensionality

- Saving grace: some datasets may have low intrinsic dimension, i.e. lie on or near a low-dimensional manifold.
- E.g., natural images have a lot fewer degrees of freedom than the number of pixels in the image.
- The distance to the neighbors depends on the intrinsic dimension, not the dimension of the input space. Hence, KNN can still work in high dimensions, as long as the data are intrinsically low-dimensional.

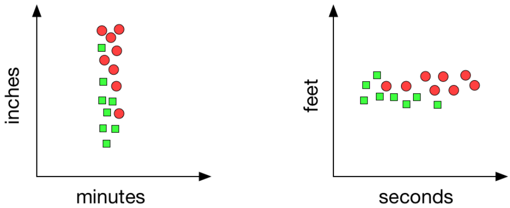

Pitfalls: Normalization

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



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! (Can you think of an example?)

Pitfalls: Computational Cost

- Number of computations at training time: 0

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!

Pitfalls: Computational Cost

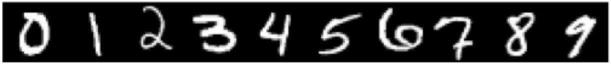
- Number of computations at training time: 0
- Number of computations at test time, per query (naïve algorithm)
  - Calculate D-dimensional Euclidean distances with N data points: $\mathcal{O}(ND)$
  - Sort the distances: $\mathcal{O}(N \log N)$
- This must be done for *each* query, which is very expensive by the standards of a learning algorithm!
- Need to store the entire dataset in memory!

Pitfalls: Computational Cost

- Number of computations at training time: 0
- Number of computations at test time, per query (naïve algorithm)
  - Calculate D-dimensional Euclidean distances with N data points: $\mathcal{O}(ND)$
  - Sort the distances: $\mathcal{O}(N \log N)$
- This must be done for *each* query, which is very expensive by the standards of a learning algorithm!
- Need to store the entire dataset in memory!
- Tons of work has gone into algorithms and data structures for efficient nearest neighbors with high dimensions and/or large datasets.

Example: Digit Classification

- Decent performance when lots of data

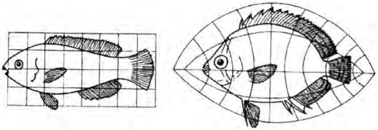


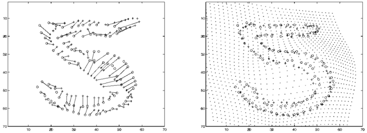
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Example: Digit Classification

- KNN can perform a lot better with a good similarity measure.
- Example: shape contexts for object recognition. In order to achieve invariance to image transformations, they tried to warp one image to match the other image.
  - Distance measure: average distance between corresponding points on *warped* images
- Achieved 0.63% error on MNIST, compared with 3% for Euclidean KNN.
- Competitive with conv nets at the time, but required careful engineering.

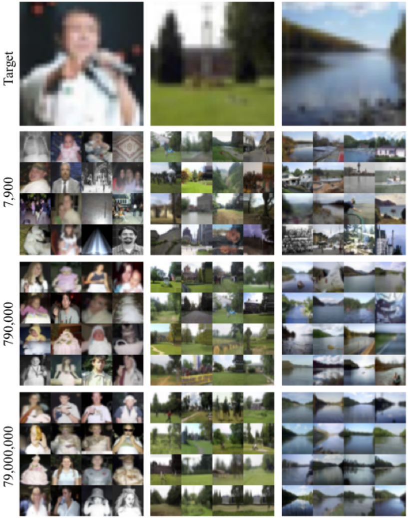




[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]

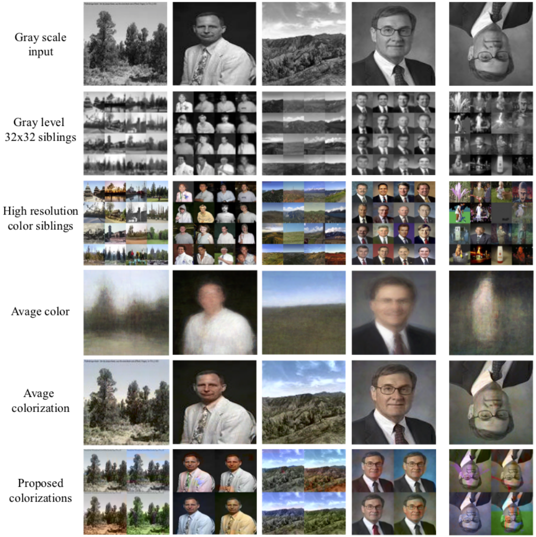
Example: 80 Million Tiny Images

- 80 Million Tiny Images was the first extremely large image dataset. It consisted of color images scaled down to 32×32.
- With a large dataset, you can find much better semantic matches, and KNN can do some surprising things.
- Note: this required a carefully chosen similarity metric.



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

Example: 80 Million Tiny Images



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

Questions?

?