# CSC321 Lecture 1: Introduction

Roger Grosse

## 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
- Some reasons you might 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

# 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

# Course information

- Course about machine learning, with a focus on neural networks
  - Independent of CSC411, and CSC412, with about 25% overlap in topics
  - First 2/3: supervised learning
  - Last 1/3: unsupervised learning and reinforcement learning
- Two sections
  - Equivalent content, same assignments and exams
  - Both sections are full, so please attend your own.

# Course information

- Formal prerequisites:
  - **Calculus:** (MAT136H1 with a minimum mark of 77)/(MAT137Y1 with a minimum mark of 73)/(MAT157Y1 with a minimum mark of 67)/MAT235Y1/MAT237Y1/MAT257Y1
  - **Linear Algebra:** MAT221H1/MAT223H1/MAT240H1
  - **Probability:** STA247H1/STA255H1/STA257H1
  - **Multivariable calculus (recommended):** MAT235Y1/MAT237Y1/MAT257Y1
  - **Programming experience (recommended)**

# Course information

- Expectations and marking
  - Written homeworks (20% of total mark)
    - Due Wednesday nights at 11:59pm, starting 1/17
    - 2-3 short conceptual questions
    - Use material covered up through Tuesday of the preceding week
  - 4 programming assignments (30% of total mark)
    - Python, PyTorch
    - 10-15 lines of code
    - may also involve some mathematical derivations
    - give you a chance to experiment with the algorithms
  - Exams
    - midterm (15%)
    - final (35%)
- See Course Information handout for detailed policies

# Course information

- Textbooks
  - None, but we link to lots of free online resources. (see syllabus)
    - Professor Geoffrey Hinton's Coursera lectures
    - the Deep Learning textbook by Goodfellow et al.
    - Metacademy
  - I will *try* to post detailed lecture notes, but I will not have time to cover every lecture.
- Tutorials
  - Roughly every week
  - Programming background; worked-through examples

# Course information

Course web page:

http://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/

Includes detailed course information handout

# Supervised learning examples

**Supervised learning:** have labeled examples of the correct behavior

e.g. Handwritten digit classification with the MNIST dataset

- **Task:** given an image of a handwritten digit, predict the digit class
  - **Input:** the image
  - **Target:** the digit class

# Supervised learning examples

**Supervised learning:** have labeled examples of the correct behavior

e.g. Handwritten digit classification with the MNIST dataset

- **Task:** given an image of a handwritten digit, predict the digit class
  - **Input:** the image
  - **Target:** the digit class
- **Data:** 70,000 images of handwritten digits labeled by humans
  - **Training set:** first 60,000 images, used to train the network
  - **Test set:** last 10,000 images, not available during training, used to evaluate performance

# Supervised learning examples

**Supervised learning:** have labeled examples of the correct behavior

e.g. Handwritten digit classification with the MNIST dataset

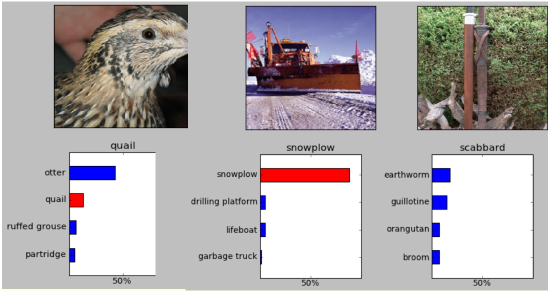
- **Task:** given an image of a handwritten digit, predict the digit class
  - **Input:** the image
  - **Target:** the digit class
- **Data:** 70,000 images of handwritten digits labeled by humans
  - **Training set:** first 60,000 images, used to train the network
  - **Test set:** last 10,000 images, not available during training, used to evaluate performance
- This dataset is the "fruit fly" of neural net research
- Neural nets already achieved $> 99\%$ accuracy in the 1990s, but we still continue to learn a lot from it

# Supervised learning examples

What makes a “2”?

# Supervised learning examples

Object recognition




(Krizhevsky and Hinton, 2012)

ImageNet dataset: thousands of categories, millions of labeled images

Lots of variability in viewpoint, lighting, etc.

Error rate dropped from 25.7% to 5.7% over the course of a few years!

# Supervised learning examples

Caption generation

**TAGS:**

frisbees frisbee pushups golfers kickball

**Nearest Neighbor Sentence:**

- several people that are playing in a frisbee game .

**Top-5 Generated:**

- a group of girls are playing a game of frisbee .
- a group of girls are playing a soccer game .
- a group of girls playing on a soccer game .
- a group of people playing a game of frisbee .
- the young people are playing a game of frisbee .

Given: dataset of Flickr images with captions

More examples at `http://deeplearning.cs.toronto.edu/i2t`

# Unsupervised learning examples

- In **generative modeling**, we want to learn a distribution over some dataset, such as natural images.
- We can evaluate a generative model by sampling from the model and seeing if it looks like the data.
- These results were considered impressive in 2014:


Denton et al., 2014, Deep generative image models using a Laplacian pyramid of adversarial networks

# Unsupervised learning examples

- New state-of-the-art:

# Unsupervised learning examples

- Recent exciting result: a model called the CycleGAN takes lots of images of one category (e.g. horses) and lots of images of another category (e.g. zebras) and learns to translate between them.


https://github.com/junyanz/CycleGAN

- You will implement this model for Programming Assignment 4.

# Reinforcement learning

- An **agent** interacts with an **environment** (e.g. game of Breakout)
- In each time step,
  - the agent receives **observations** (e.g. pixels) which give it information about the **state** (e.g. positions of the ball and paddle)
  - the agent picks an **action** (e.g. keystrokes) which affects the state
- The agent periodically receives a **reward** (e.g. points)
- The agent wants to learn a **policy**, or mapping from observations to actions, which maximizes its average reward over time

# Reinforcement learning

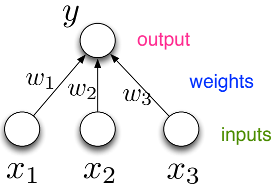
DeepMind trained neural networks to play many different Atari games

- given the raw screen as input, plus the score as a reward
- single network architecture shared between all the games
- in many cases, the networks learned to play better than humans (in terms of points in the first minute)

https://www.youtube.com/watch?v=V1eYniJ0Rnk

# What are neural networks?

- Most of the biological details aren't essential, so we use vastly simplified models of neurons.
- While neural nets originally drew inspiration from the brain, nowadays we mostly think about math, statistics, etc.




output  bias  i'th weight

$$y = g\left(b + \sum_i x_i w_i\right)$$

nonlinearity  i'th input

- Neural networks are collections of thousands (or millions) of these simple processing units that together perform useful computations.

# What are neural networks?

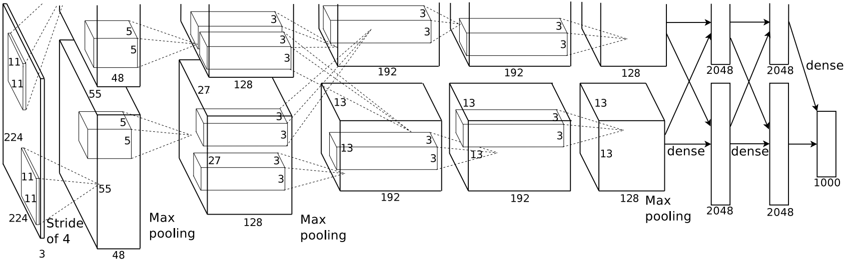
Why neural nets?

- inspiration from the brain
  - proof of concept that a neural architecture can see and hear!
- very effective across a range of applications (vision, text, speech, medicine, robotics, etc.)
- widely used in both academia and the tech industry
- powerful software frameworks (Torch, PyTorch, TensorFlow, Theano) let us quickly implement sophisticated algorithms

# "Deep learning"

Deep learning: many layers (stages) of processing

E.g. this network which recognizes objects in images:

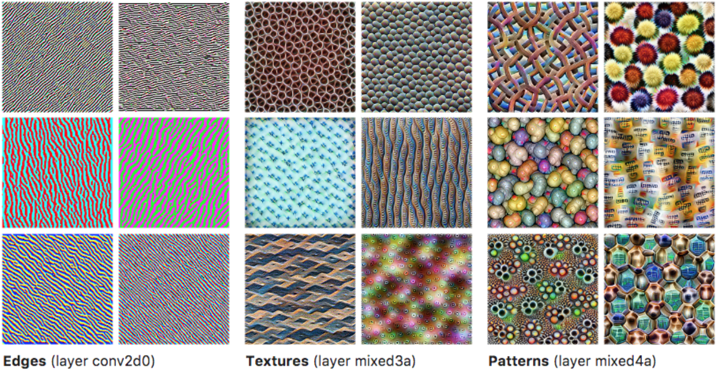



(Krizhevsky et al., 2012)

Each of the boxes consists of many neuron-like units similar to the one on the previous slide!

# "Deep learning"

- You can visualize what a learned feature is responding to by finding an image that excites it. (We'll see how to do this.)
- Higher layers in the network often learn higher-level, more interpretable representations




https://distill.pub/2017/feature-visualization/

# "Deep learning"

- You can visualize what a learned feature is responding to by finding an image that excites it.
- Higher layers in the network often learn higher-level, more interpretable representations

**Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

https://distill.pub/2017/feature-visualization/

# Software frameworks

- Array processing (NumPy)
  - **vectorize** computations (express them in terms of matrix/vector operations) to exploit hardware efficiency
- Neural net frameworks: Torch, PyTorch, TensorFlow, Theano
  - automatic differentiation
  - compiling computation graphs
  - libraries of algorithms and network primitives
  - support for graphics processing units (GPUs)
- For this course:
  - Python, NumPy
  - **Autograd**, a lightweight automatic differentiation package written by Professor David Duvenaud and colleagues
  - **PyTorch**, a widely used neural net framework

# Software frameworks

Why take this class, if PyTorch does so much for you?

So you know what do to 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!

# Next time

Next lecture: linear regression