A to Z of AI/ML: A Quick Introduction to Artificial Intelligence and Machine Learning Capabilities and Tools

EngCon 2017

Mark Crowley
Assistant Professor
Electrical and Computer Engineering
University of Waterloo
mcrowley@uwaterloo.ca

Sep 23, 2017

Outline

Introduction

What is AI?

Neural Networks

Convolutional Neural Networks

Do you need AI/ML?

My Background

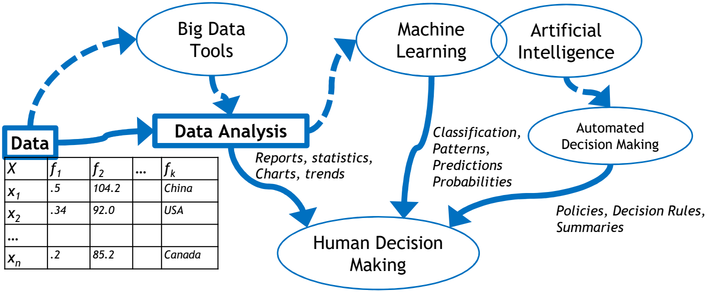
- Waterloo : Assistant Professor, ECE Department since 2015
- PhD at UBC in Computer Science with Prof. David Poole
- Postdoc at Oregon State University
- UW ECE ML Lab: `https://uwaterloo.ca/scholar/mcrowley/lab`
- Waterloo Institute for Complexity and Innovation (WICI)
- Research Fellow at Element^AI
- Pattern Analysis and Machine Intelligence (PAMI)
- `http:\waterloo.ai`
    - List of faculty
    - Research projects (co-op/internships)
    - List of spinoff companies from UWaterloo (good place for project ideas)

What do you think of when you hear?

Artificial Intelligence

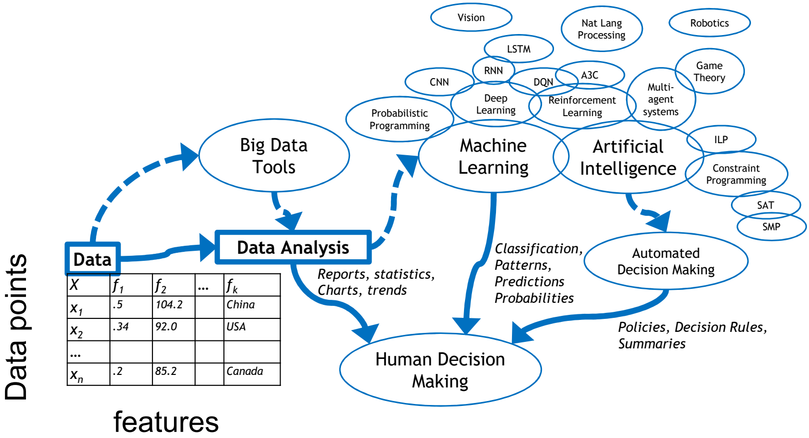
Machine Learning

Data, Big Data, Machine Learning, AI, etc, etc,



X	f_1	f_2	...	f_k
x_1	.5	104.2		China
x_2	.34	92.0		USA
...				
x_n	.2	85.2		Canada

Data, Big Data, Machine Learning, AI, etc, etc,



X	f_1	f_2	...	f_k
x_1	.5	104.2		China
x_2	.34	92.0		USA
...				
x_n	.2	85.2		Canada

Major Types/Areas of AI

Artificial Intellgience: some algorithm to enable computers to perform actions we define as requireing intelligence.

Major Types/Areas of AI

Artificial Intellgience: some algorithm to enable computers to perform actions we define as requireing intelligence. **This is a moving target.**

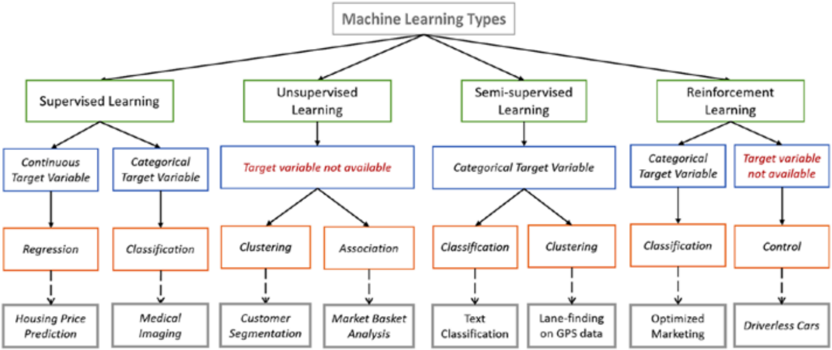
Major Types/Areas of AI

Artificial Intellgience: some algorithm to enable computers to perform actions we define as requireing intelligence. **This is a moving target.**

- Search Based Heuristic Optimization (A*)
- Evolutionary computation (genetic algorithms)
- Logic Programming (inductive logic programming, fuzzy logic)
- Probabilistic Reasoning Under Uncertainty (bayesian networks)
- Computer Vision
- Natural Language Processing
- Robotics
- **Machine Learning**

Types of Machines Learning

Machine Learning: *"Detect patterns in data, use the uncovered patterns to predict future data or other outcomes of interest"* – Kevin Murphy, Google Research.



Deep Learning

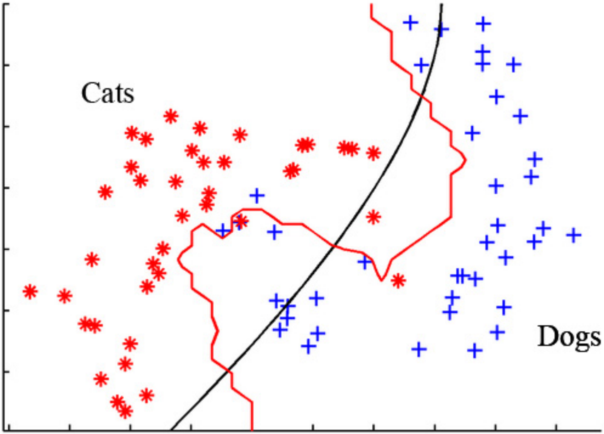
Deep Learning: methods which perform machine learning through the use of multilayer neural networks of some kind. Deep Learning can be applied in any of the three main types of ML:

- **Supervised Learning** : very common, enourmous improvement in recent years
- **Unsupervised Learning** : just beginning, lots of potential
- **Reinforement Learning** : recent (past 3 years) this has exploded, exspecially for video games

Increasing Complexity of Supervised ML Methods

1. mean, mode, max, min - basic statistics and patterns
2. prediction/regression - least squares, ridge regression
3. linear classification - use distances and separation of data points. (logistic regression, SVM, KNN)
4. Kernel Based Classification - define a mapping from original data to a new space, allow nonlinear divisions to be found
5. Decision trees - learn rules that divide data arbitrarily (C4.5, Random Forests, AdaBoost)
6. Neural Networks - learn function using 'neurons'
7. Deep Neural Networks - same, but deep :)
8. Recurrent Neural Networks - adding links to past timesteps, learning with memory of the past
9. Convolutional Neural Networks - adding convolutional filters, good for images
10. Inception Resnets, Long-Term Short-Term Networks, Voxception Networks, ... oh it keeps going...

One Example of ML: Classification



Clustering vs. Classification

Clustering

- Unsupervised
- Uses unlabeled data
- Organize patterns w.r.t. an optimization criteria
- Requires a definition of similarity
- Hard to evaluate
- Examples: K-means, Fuzzy C-means, Hierarchical Clustering, DBScan

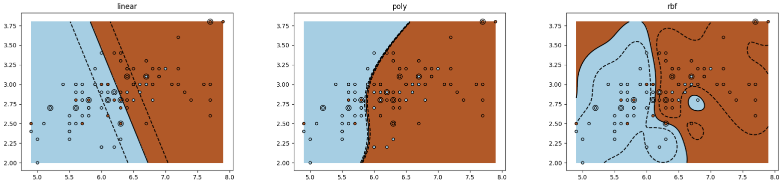
Classification

- Supervised
- Uses labeled data
- Requires training phase
- Domain sensitive
- Easy to evaluate (you know the correct answer)
- Examples: Naive Bayes, KNN, SVM, Decision Trees, Random Forests

Classification Performance Depends on the Algorithm

A good example of this choices is Support Vector Machines (SVMs).

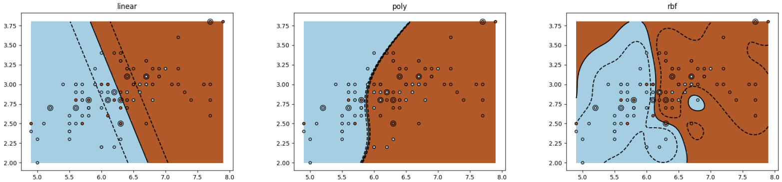
- popular until dawn of deep learning in past few years
- core idea: find a dividing hyperplane
- many variations: plane can be linear, polynomial, gaussian, high-dimensional



Classification Performance Depends on the Algorithm

A good example of this choices is Support Vector Machines (SVMs).

- popular until dawn of deep learning in past few years
- core idea: find a dividing hyperplane
- many variations: plane can be linear, polynomial, gaussian, high-dimensional

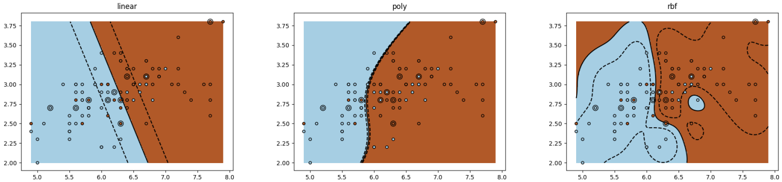


So what is the "right" approach?

Classification Performance Depends on the Algorithm

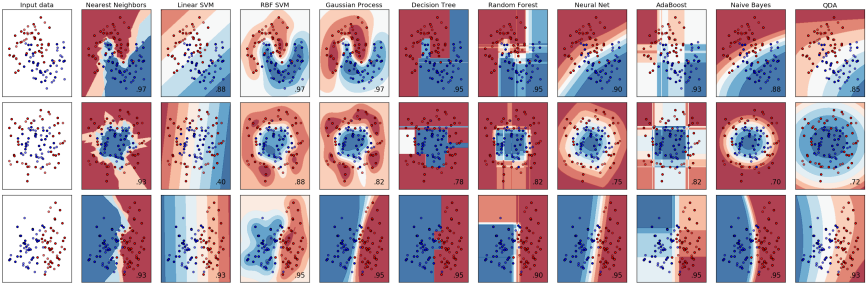
A good example of this choices is Support Vector Machines (SVMs).

- popular until dawn of deep learning in past few years
- core idea: find a dividing hyperplane
- many variations: plane can be linear, polynomial, gaussian, high-dimensional



So what is the “right” approach? Experimentation!

Classification Performance Depends on the Algorithm



So choose carefully...

See `http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html`

Outline

Introduction

What is AI?

Neural Networks

- Building Upon Classic Machine Learning
- History Of Neural Networks
- Improving Performance

Convolutional Neural Networks

Do you need AI/ML?

Linear Regression vs. Logistic Regression

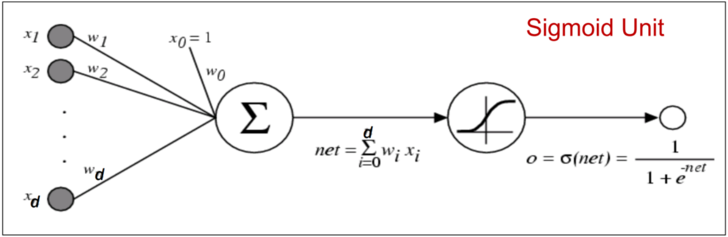
- A simple type of **Generalized Linear Model**
- Linear regression learns a function to predict a continuous variable output of continous or discrete input variables

$$Y = b_0 + \sum(b_i X_i) + \epsilon$$

- Logistic regression predicts the probability of an outcome, the appropriate class for an input vector or the **odds** of one outcome being more likely than another.

Logistic Regression as a Graphical Model

$$o(\mathbf{x}) = \sigma(w^T x_i) = \sigma(w_0 + \sum_i w_i x_i) = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i x_i))}$$



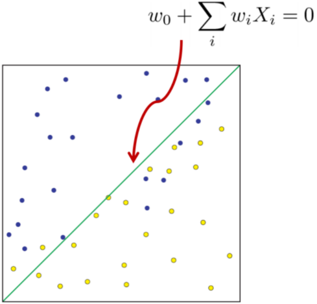
Logistic Regression Used as a Classifier

Logistic Regression can be used as a simple linear classifier.

- Compare probabilities of each class $P(Y=0|X)$ and $P(Y=1|X)$.
- Treat the halfway point on the sigmoid as the decision boundary.

$P(Y=1|X) > 0.5$ classify X in class 1

$$w_0 + \sum_i w_i x_i = 0$$



Training Logistic Regression Model via Gradient Descent

- Can't easily perform Maximum Likelihood Estimation
- The negative log-likehood of the logistic function is given by NLL and it's gradient by g

$$NLL(w) = \sum_{i=1}^{N} \log \left(1 + \exp(-(w_0 + \sum_i w_i x_i)) \right)$$

$$g = \frac{\partial}{\partial w} = \sum_i (\sigma(w^T x_i) - y_i) x_i$$

Then we can update the parameters iteratively

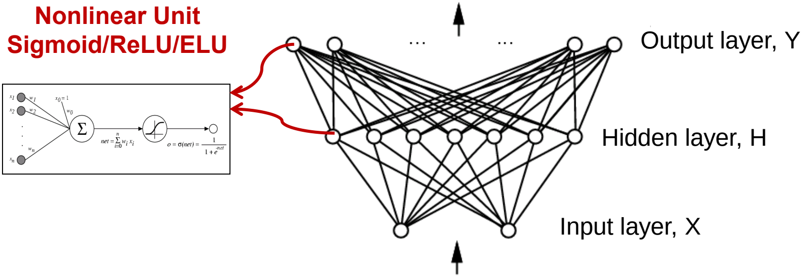
$$\theta_{k+1} = \theta_k - \eta_k g_k$$

where η_k is the learning rate or step size.

Neural Networks to learn $f : X \rightarrow Y$

- f can be a non-linear function
- **X** (vector of) continuous and/or discrete variables
- **Y** (vector of) continuous and/or discrete variables

Feedforward Neural networks - Represent f by network of non-linear (logistic/sigmoid/ReLU) units:



Basic Three Layer Neural Network

Input Layer

- vector data, each input collects **one feature**/dimension of the data and passes it on to the (first) hidden layer.

Hidden Layer

- Each hidden unit computes a weighted sum of all the units from the input layer (or any previous layer) and passes it through a **nonlinear activation function**.

Output Layer

- Each output unit computes a weighted sum of all the hidden units and passes it through a (possibly nonlinear) **threshold function**.

Properties of Neural Networks

- Universality: Given a large enough layer of hidden units (or multiple layers) a NN can represent **any function**.
- Representation Learning: classic statistical machine learning is about learning functions to map input data to output. But Neural Networks, and especially Deep Learning, are more about learning **a representation** in order to perform classification or some other task.

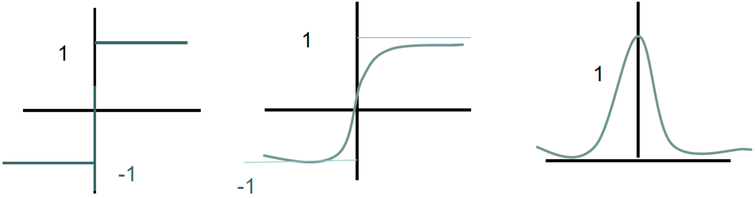
Hidden Layer: Adding Nonlinearity

- Each hidden unit emits an output that is a nonlinear **activation function** of its net activation.

$$y_j = f(net_j)$$

- This is essential to neural networks power, if it's linear then it all becomes just linear regression.
- The output is thus thresholded through this nonlinear activation function.

Activation Functions

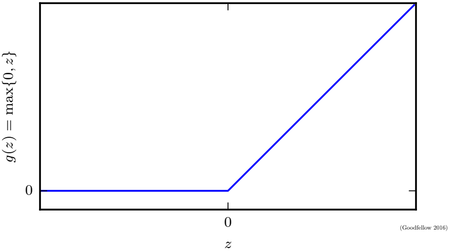


a) Threshold b) Sigmoid c) Gaussian

$$F_T(x) = \begin{cases} 1 & \text{if } x > \tau \\ -1 & \text{otherwise} \end{cases} \qquad F_S(x) = \frac{1}{\left(1 + e^{-cx}\right)} \qquad F_G(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

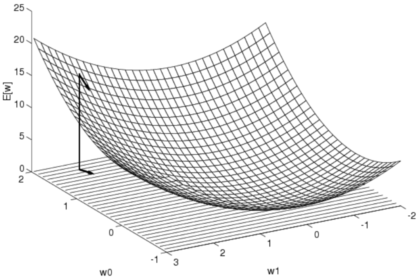
- `tanh` was another common function.
- `sigmoid` is now discourage except for final layer to obtain probabilities. Can **over-saturate** easily.
- ReLU is the new standard activation function to use.

Rectified Linear Activation



- **Rectified Linear Units (ReLU)** have become standard $\max(0, net_j)$
  - strong signals are alwasy easy to distinguish
  - most values are zero, deritive is mostly zero
  - they do not saturate as easily as sigmoid
- new Exponential linear units - evidence that they perform better than ReLU in some situations.

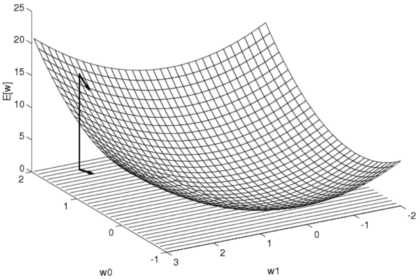
Gradient Descent



Error Function: Mean Squared Error, cross-entropy loss, etc.

(Slides from Tom Mitchell ML Course, CMU, 2010)

Gradient Descent



Error Function: Mean Squared Error, cross-entropy loss, etc.

Gradient: $\nabla E[\mathbf{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_d}\right]$

Training Update Rule: $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$ where η is the training rate.

Note: For regression, others, this gradient is convex. In ANNs it is not. So we must solve iteratively

(Slides from Tom Mitchell ML Course, CMU, 2010)

Incremental Gradient Descent

Let error function be : $E_l[\mathbf{w}] = \frac{1}{2}(y^l - o^l)^2$

Do until satisfied:

- For each training example l in D
  1. Compute the gradient $\bigtriangledown E[\mathbf{w}]$
  2. update weights : $\mathbf{w} = \mathbf{w} - \eta \bigtriangledown E[\mathbf{w}]$

Note: can also use batch gradient descent on many points at once.

Backpropagation Algorithm

We need an iterative algorithm for getting the gradient efficiently.
For each training example:

1. Forward propagation: Input the training example to the network and compute outputs
2. Compute output units errors:

$$\delta_k^l = o_k^l(1 - o_k^l)(y_k^l - o_k^l)$$

3. Compute hidden units errors:

$$\delta_h^l = o_h^l(1 - o_h^l)\sum_k w_{h,k}\delta_k^l$$

4. Update network weights:

$$w_{i,j} = w_{i,j} + \Delta w_{i,j}^l = w_{i,j} + \eta\delta_j^l o_i^l$$

A Short History

- 40's Early work in NN goes back to the 40s with a simple model of the neuron by McCulloh and Pitt as a summing and thresholding devices.
- 1958 Rosenblatt in 1958 introduced the **Perceptron**,a two layer network (one input layer and one output node with a bias in addition to the input features.
- 1969 Marvin Minsky: 1969. Perceptrons are 'just' linear, AI goes logical, beginning of "AI Winter"
- 1980s Neural Network resurgence: Backpropagation (updating weights by gradient descent)
- 1990s SVMs! Kernals can do **anything!** (no, they can't)

A Short History

- **1993** `LeNet 1` for digit recognition
- **2003** Deep Learning (Convolutional Nets Dropout/RBMs, Deep Belief Networks)
- **1986, 2006** Restricted Boltzman Machines
- **2006** Neural Network outperform RBF SVM on MNIST handwriting dataset (Hinton et al.)
- **2012** `AlexNet` for **ImageNet** challenge - this algorithm beat competition by error rate of 16% vs 26% for next best
  - ImageNet : contains 15 million annotated images in over 22,000 categories.
  - ZFNet paper (2013) extends this and has good description of network structure
- **2012-present** Google Cat Youtube, speech recognition, self driving cars, computer defeats regional Go champion, ...
- **2014** `GoogLeNet` added many layers and introduced inception modules (allows parallel computation rather than serially

A Short History

- 2014 Generative Adversarial Networks (GANs) introduced.
- 2015 Microsoft algorithm beats human performance at ImageNet challenge.
- 2016 AlphaGo defeats one of best world players of Go Lee Sedol using Deep Reinforcement Learning.
- 2016 Deep Mind introduces A3C Deep RL algorithm that can learn to play Atari games from images by playing with no instructions.

Outline

Introduction

What is AI?

Neural Networks

- Building Upon Classic Machine Learning
- History Of Neural Networks
- Improving Performance

Convolutional Neural Networks

Do you need AI/ML?

Problems with ANNs

- Overfitting
- Very inneficient for images, timeseries, large numbers of inputs-outputs
- Slow to train
- Hard to interpret the resulting model
- Overfitting

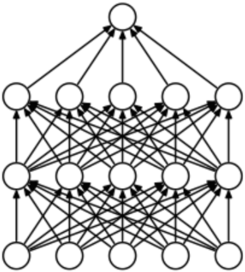
Heuristics for Improving Backpropagation

There are a number of useful heuristics for training Neural Networks that are useful in practice (maybe we'll learn more today):

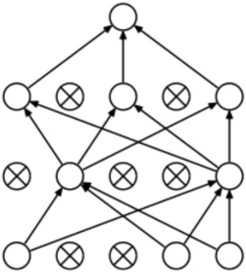
- Less hidden nodes, just enough complexity to work, not too much to overfit.
- Train multiple networks with different sizes and search for the best design.
- Validation set: train on training set until error on validation set starts to rise, then evaluate on evaluation set.
- Try different activiation functions: tanh, ReLU, ELU, ...?
- Dropout (Hinton 2014) - randomly ignore certain units during training, don't update them via gradient descent, leads to hidden units that specialize
- Modify learning rate over time (cooling schedule)

Dropout

- Dropout (Hinton 2014) - randomly ignore certain units during training, don't update them via gradient descent, leads to hidden units that specialize.
- With probability p don't include a weight in the gradient updates.
- Reduces overfitting by encouraging robustness of weights in the network.

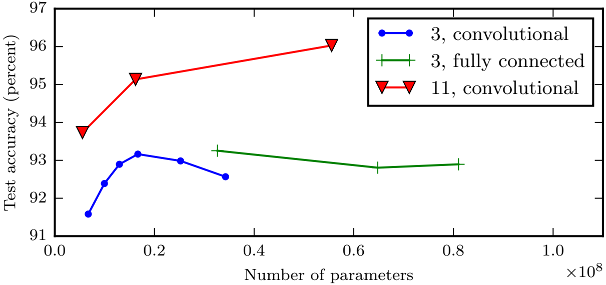


Standard Neural Net



After applying dropout.

Large, Shallow Models Overfit More



(Goodfellow 2016)

Outline

Introduction

What is AI?

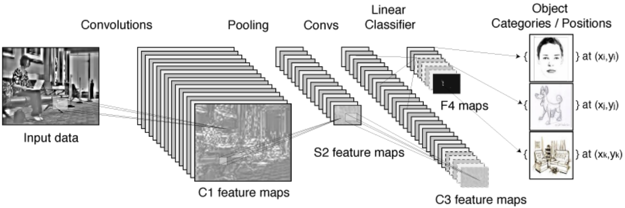
Neural Networks

Convolutional Neural Networks
- Motivation
- Other Types of Deep Neural Networks

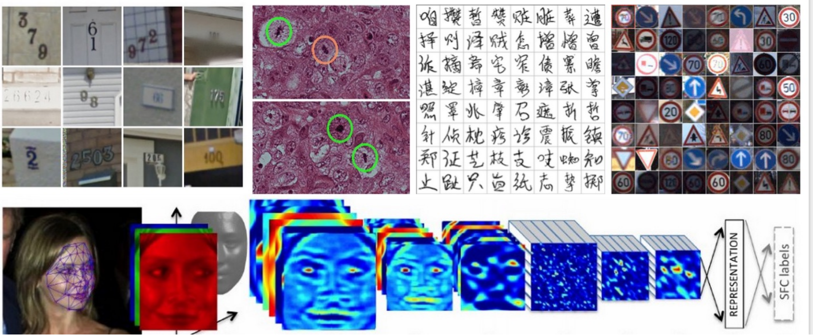
Do you need AI/ML?

Convolutional Network Structure

- input data: image (eg. 256x256 pixels x3 channels RGB)
- output : categorical label



Example Applications of CNNs

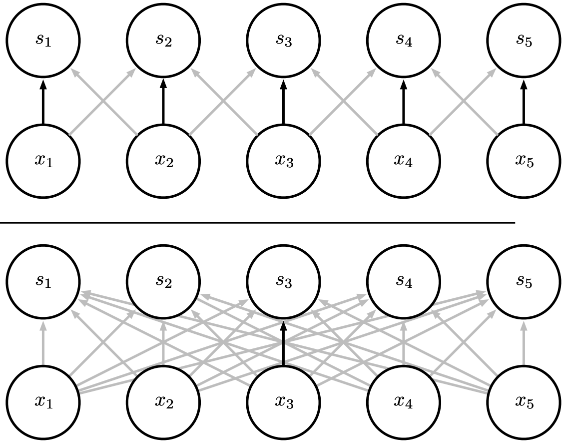


(Karpathy Blog, Oct, 25, 2015 - http:karpathy.github.io20151025selfie)

Parameter sharing

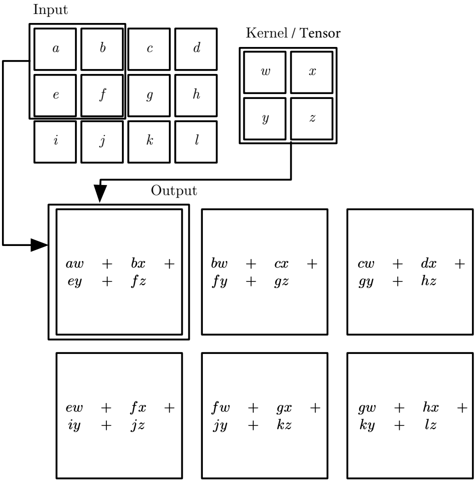
Convolution shares the same parameters across all spatial locations

Traditional matrix multiplication does not share any parameters



(Goodfellow 2016)

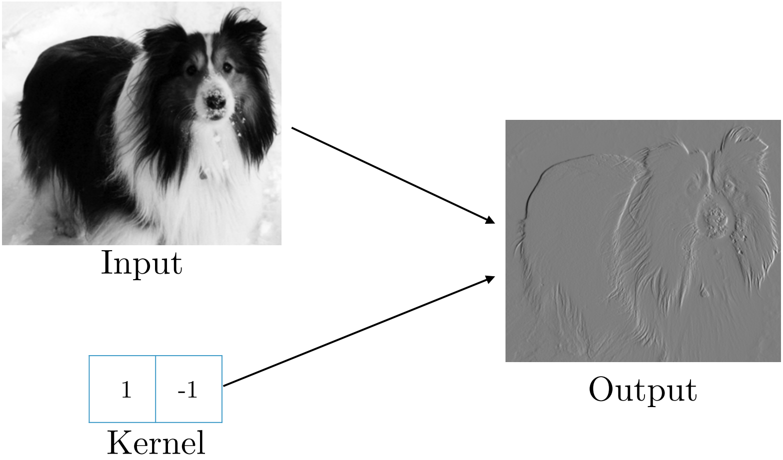
2D Convolution



(Goodfellow 2016)

A simple example

Edge detection by convolution with a kernal that subtracts the value from the neighbouring pixel on the left for every pixel.



(Goodfellow 2016)

Other CNN Modification

Pooling: Nearby pixels tend to represent the same thing/class/object. So, pool responses from nearby nodes. (eg. mean, median, min, **max**)

Strides: number of pixels overlap between adjacent filters

Zero padding: removing edge pixels from filter scan, can reduce size of network and deal with edge effects

Connectivity: Alternate local connectivity options, partial connectivity

Other Types of Deep Neural Networks

- **RBM:** Restricted Boltzman Machines (RBM) - older directed deep model.
- **RNN:** Recurrent Neural Networks (RNN) - allow links from outputs back to inputs, over time, good for time series learning
- **LSTM:** Long-Term Short-Term networks - more complex form of RNN
- **DeepRL:** Deep Reinforcement Learning
- **GAN:** General Adversarial Networks - train two networks at once

Other Types of Deep Neural Networks

RBM: Restricted Boltzman Machines (RBM) - older directed deep model.

RNN: Recurrent Neural Networks (RNN) - allow links from outputs back to inputs, over time, good for time series learning

LSTM: Long-Term Short-Term networks - more complex form of RNN

- integrate strategically remembered particular information from the past
- formalizes a process for *forgetting* information over time.
- useful if you need to learn patterns over time and your data feautres

DeepRL: Deep Reinforcement Learning

GAN: General Adversarial Networks - train two networks at once

Other Types of Deep Neural Networks

RBM: Restricted Boltzman Machines (RBM) - older directed deep model.

RNN: Recurrent Neural Networks (RNN) - allow links from outputs back to inputs, over time, good for time series learning

LSTM: Long-Term Short-Term networks - more complex form of RNN

DeepRL: Deep Reinforcement Learning

- CNNs + Fully Connected Deep Network for learning a representation of a policy
- Reinforcement Learning for updating the policy through experience to make improved decision decisions
- Requires a value/reward function

GAN: General Adversarial Networks - train two networks at once

Other Types of Deep Neural Networks

RBM: Restricted Boltzman Machines (RBM) - older directed deep model.

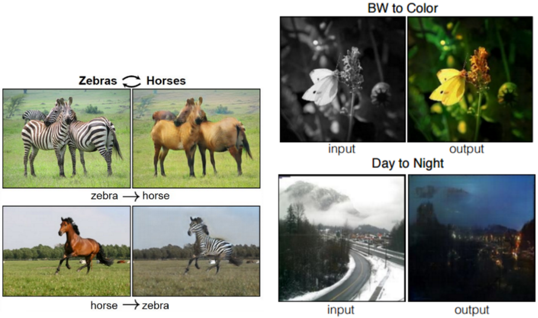
RNN: Recurrent Neural Networks (RNN) - allow links from outputs back to inputs, over time, good for time series learning

LSTM: Long-Term Short-Term networks - more complex form of RNN

DeepRL: Deep Reinforcement Learning

GAN: General Adversarial Networks - train two networks at once

General Adversarial Networks



- One network produces/hallucinates new answers (generative)
- Second network distinguishes between the real and the generated answers (adversary/critic)
- Blog withCode: "GANS in 50 lines of code PyTorch code." easy way to get started

Outline

Introduction

What is AI?

Neural Networks

Convolutional Neural Networks

Do you need AI/ML?

- Defining Your Questions
- Designing Your AI/ML System
- Languages and Libraries
- Deep Learning Frameworks
- Compute Resources

Defining Your Questions

- Is it a decision to be made?
- Is there a pattern to detect?
- Do you have data?
- What kinds of questions do you have about the data?

  Yes/No questions - Did X happen? Are A and B correlated?

  Timing - When did X happen?

  Anomaly detection - Is X strange/abnormal/unexpected?

  Classification - What kind of Y is X?

  Prediction - Weve seen lots of (X,Y) now we want to know (X',?)
- Do you have labels?
  - Can you give the right answer for some portion of the data?
  - Collecting labels: Automatic? Manual? Crowd-sourced? (eg. Amazon Mechanical Turk) Y
  - Yes → Supervised Learning - Lots of options
  - No → Unsupervised Learning - Some options (getting better all the time)

Answers and Constraints

What kind of answer do you need? (increasing difficulty)

- Find patterns which are present in the data and view them
- Most likely explanation for a pattern
- Probability of (fact about X,A,B...) being true
- A policy for actions to take in the future to maximize benefit
- Guarantees that X will (or will not) happen (very hard)

How big is your data?

- Is it static?
- MB, GB, TB?
- Is it streaming?
- KB/sec, MB/sec
- How many data points/rows/events will there be?

How to Design your AI/ML Question

Define your task:

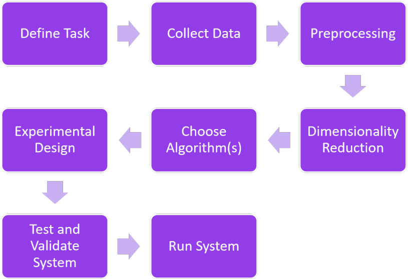
- Prediction, Clustering, Classification, Anomaly Detection?
- Define objectives, error metrics, performance standards

Collect Data:

- Set up data stream (storage, input flow, parallelization, Hadoop)

Preprocessing:

- Noise/Outlier Filtering
- Completing missing data (histograms, interpolation)
- Normalization (scaling data)



How to Design your AI/ML Question

Dimensionality Reduction / Feature Selection:

- Choose features to use/extract from data
- PCA/LDA/LLE/GDA

Choose Algorithm:

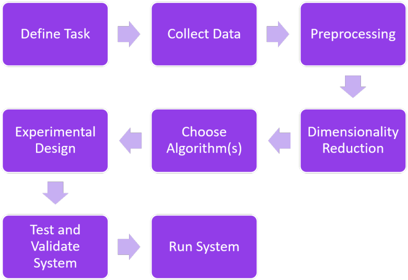
- Consider goals, questions
- Tractability

Experimental Design:

- train/validate/test data sets
- cross-validation

Run it! :

- Deployment



Language Choices

Any language can be used for implementing/using AI/ML algorithms, but some make it much easier

- **C++:** you can do it, may need to implement many things yourself
- **Java:** many of libraries for ML (Weka is a good open source one, Deeplearning4j)
- **Scala:** leaner, functional language that compile to JVM bytecode, good for prototyping, can reuse libraries for Java (Deeplearning4j)
- **R:** focussed on statisical methods, more and more machine learning libraries implemented for this
- **Matlab:** good for all the calculations, if you have the right libraries it's great (not cheap or very portable beyond school)
- **Python:** most commonly used right now for deep learning, we're gonna need another slide ...

Python

- **numpy** - numerical libraries, implementation of matrix and linear algebra datastructures, graphing tools
- **pandas** - table datastructure, statistical analysis tools (implements many useful features from R)
- **scipy** - includes all of the above and more, full installation of scientific libraries, basically turns Python into R+Matlab
- **scikit-learn** - many standard machine learning algorithms implemented as easy-to-use Python APIs
- **jupyter notebooks** - these are powerful web-based interfaces to python for data analysis and machine learning.

Deep Learning Frameworks

- Caffe - older, easy to set up mockups, harder to install?
- Theano - made out of University of Montreal, great theoretical setup, very flexible, python only
- Tensorflow - made by Google, scales to many GPUs, servers, lots of optimization, requires planning of the whole system beforehand, most languages
- PyTorch - easier to mock things up, try different designs, not as optimized for large scale performance as tensorflow
- MXNet - made by Microsoft, supports most languages and OS's
- Deeplearning4j - Java focussed framework
- Keras - open interface to create models in multiple frameworks (tensorflow, theano, MXNet)

Cloud Services

There are several powerful, free services you can access via a student account which you can request directly.

AWS: Amazon Web Service - very large, has accessible APIs to connect to, many options for hardware to run on (but the best ones will cost extra)

Azure: Microsoft - lots of visual tools for composing AI/ML components.

Google Cloud ML Engine: - uses all the latest tools and tensorflow models

None of these provide GPU servers for free, that will cost extra. (It will still work, just be slower for deep learning.)

Summary

Introduction

What is AI?
- Landscape of Big Data/AI/ML
- Classification

Neural Networks
- Building Upon Classic Machine Learning
- History Of Neural Networks
- Improving Performance

Convolutional Neural Networks
- Motivation
- Other Types of Deep Neural Networks

Do you need AI/ML?
- Defining Your Questions
- Designing Your AI/ML System
- Languages and Libraries
- Deep Learning Frameworks
- Compute Resources

Useful Books

A book for of three eras of Machine Learning:

[Goodfellow, 2016]
Goodfellow, Bengio and Courville. *"Deep Learning"*, MIT Press, 2016.

- http://www.deeplearningbook.org/
- Website has free copy of book as pdf's.

[Murphy, 2012]
Kevin Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.

[Duda, Pattern Classification, 2001]
R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification (2nd ed.)*, John Wiley and Sons, 2001.

Useful Papers and Blogs

[lecun2015]
Y. LeCun, Y. Bengio, G. Hinton, L. Y., B. Y., and H. G., "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436444, 2015. Great references at back with comments on seminal papers.

[bengio2009]
Y. Bengio, "Learning Deep Architectures for AI", *Foundations and Trends in Machine Learning*, vol. 2, no. 1. 2009. An earlier general referenceon the fundamentals of Deep Learning.

[krizhevsky2012]
A. Krizhevsky, G. E. Hinton, and I. Sutskever,"ImageNet Classification with Deep Convolutional Neural Networks", *Adv. Neural Inf. Process. Syst.* pp. 19, 2012. The beginning of the current craze.

[Karpathy, 2015]
Andrej Karpathy's Blog - `http://karpathy.github.io` Easy to follow explanations with code.