Machine Learning Basics: Learning Algorithms

Sargur N. Srihari
srihari@cedar.buffalo.edu

Overview of discussion

- Deep learning is a specific kind of Machine Learning
  - To understand deep learning well, one needs a solid understanding of basic principles of ML
- This chapter is a brief course on most important general principles of ML
- Such as discussed in Intro to ML (Bishop 2006)

Plan of discussion

- Definition of a learning algorithm
  - Example of linear regression algorithm
- How fitting data differs from generalizing to new data
- ML algos have hyperparameters determined outside of learning algorithm– how to set using additional data
- Statistical approaches: frequentist and Bayesian
- Supervised versus unsupervised learning
- Optimization using stochastic gradient descent
- Combining components into an ML algorithm
  - Optimization, cost function, model, and dataset
- Factors that have limited traditional ML to generalize
  - Motivating deep learning to overcome these obstacles

Topics in Machine Learning Basics

1. Learning Algorithms
2. Capacity, Overfitting and Underfitting
3. Hyperparameters and Validation Sets
4. Estimators, Bias and Variance
5. Maximum Likelihood Estimation
6. Bayesian Statistics
7. Supervised Learning Algorithms
8. Unsupervised Learning Algorithms
9. Stochastic Gradient Descent
10. Building a Machine Learning Algorithm
11. Challenges Motivating Deep Learning

1. Learning Algorithms

- An ML algorithm is an algorithm that is able to learn from data
  - But what do we mean by learning?
  - Definition (well-posed learning problem):
    - 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 task T, as measured by P, improves with experience E
- One can imagine a wide variety of experiences E, tasks T, and performance measures P
  - We intuitively describe what these are

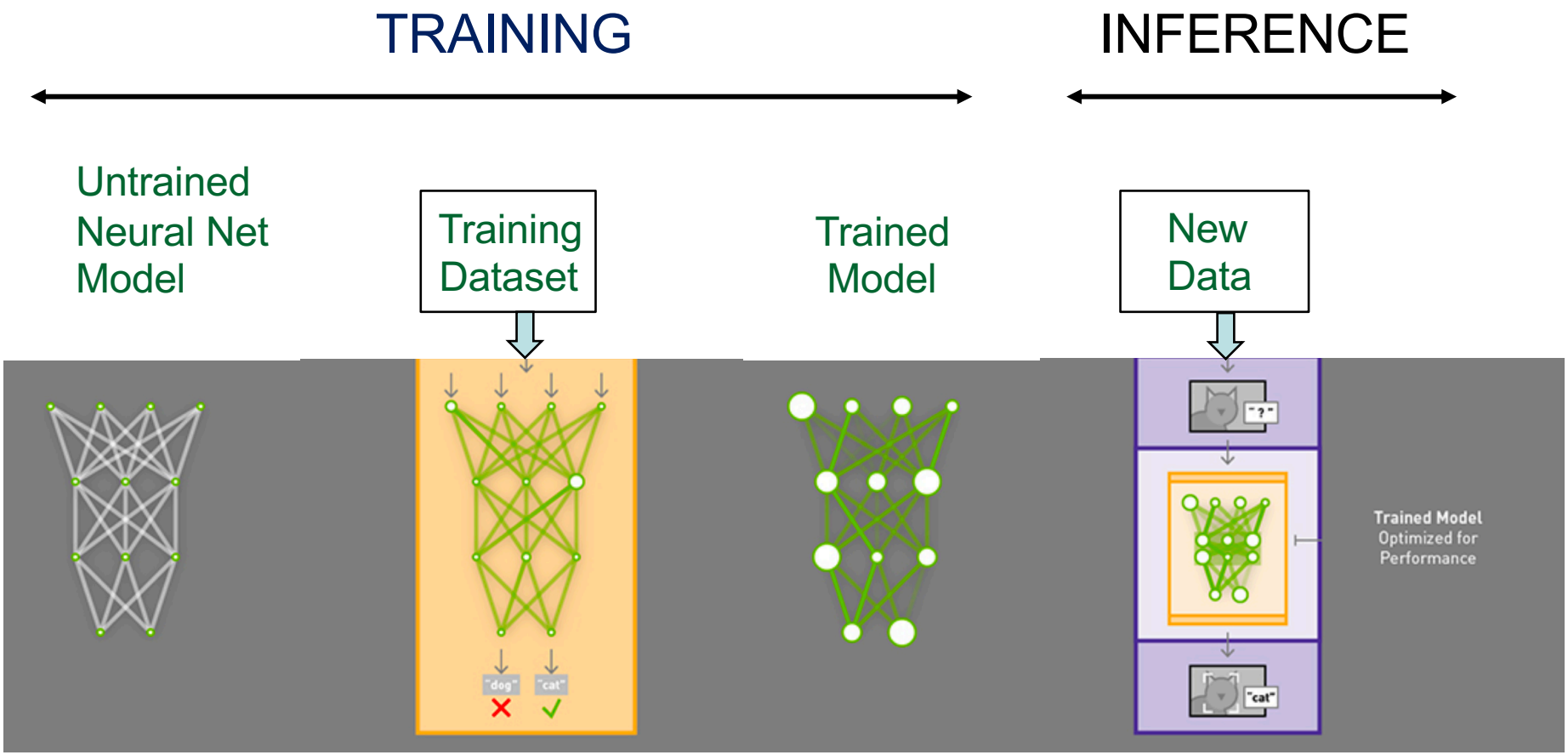
The Task, T

- ML enables tackling tasks too difficult to solve with fixed programs, written and designed by human beings
  - Scientific and philosophical perspective:
    - ML is interesting because developing ML entails understanding principles that underlie intelligence
- Process of learning itself is not the task
  - Learning is our means of attaining ability to perform the task
    - E.g., if we want a robot to be able to walk, then walking is the task
    - We could either program the robot to learn to walk or we could directly write a program manually for walking

Machine Learning Task Description

- Usually described in terms of how the machine learning system should process an example
- An example is a collection of features that have been quantitatively measured for some object/event that we want the ML system to process
- Typically represent an example as a vector $\boldsymbol{x} \in \mathbb{R}^n$ where each entry x_i of the vector is another feature

Inference in Machine Learning



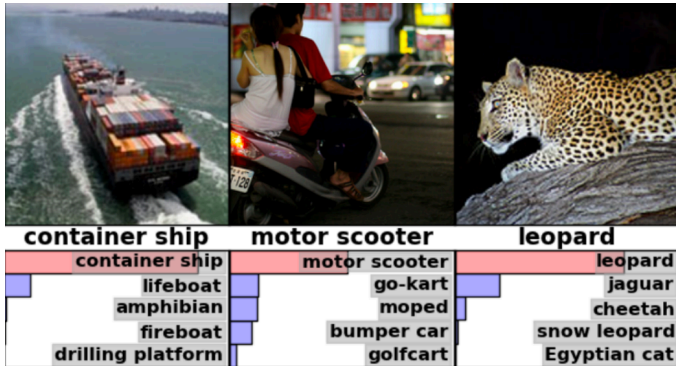
https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai/

Kinds of Tasks solved using ML

1. Classification
2. Classification with missing inputs
3. Regression
4. Transcription
5. Machine Translation
6. Structured Output
7. Anomaly Detection
8. Synthesis and Sampling
9. Imputation of Missing Values
10. Denoising
11. Density Estimation

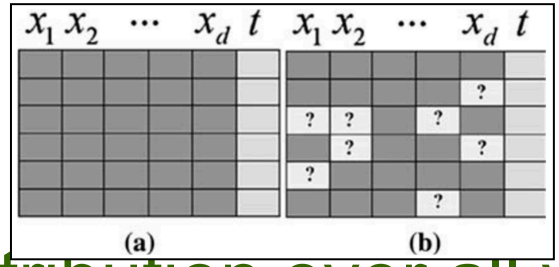
1. Classification

- Computer program asked to specify which of k categories some input belongs to
  - Learning algorithm is asked to produce a function f: $\mathrm{R}^n \rightarrow \{1,..,k\}$, where n =no of input variables
    - When $y = f(\boldsymbol{x})$ model assigns input vector $\boldsymbol{x}$ to a category identified by a numeric code y
- Other variants of classification task:
  - f outputs a probability distribution over classes



2. Classification with missing inputs

- Classification more challenging when not every measurement in its input vector is present
  - Simple classification requires a single function mapping from a vector input to a category output
    - When inputs missing, must learn a set of functions
    - Each corresponds to classifying $\boldsymbol{x}$ with a different subset of its inputs missing
      - Ex: medical diagnosis when medical tests are expensive/invasive



  - Solution: learn distribution over all variables, solve by marginalizing over missing variables

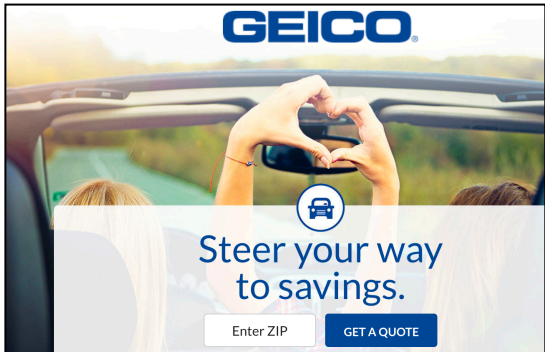
3. Regression

- Computer program required to predict a numerical value given some input
- Algorithm to output $f: \mathrm{R}^n \rightarrow \mathrm{R}$
  - Task similar to classification except that format of output is different
  - Ex: expected claim amount an insured person will make (used to set insurance premiums) or prediction of future prices of securities
    - Also used for algorithmic trading

Policyholder age	Mean	Std. Error
60+	186.08	6.084
50-59	186.20	5.551
40-49	184.38	5.155
35-39	171.70	5.516
30-34	203.40	6.930
25-29	208.15	7.482
21-24	219.52	10.824
17-20	224.51	20.690

Vehicle group	Mean	Std. Error
D	239.87	9.513
C	200.98	6.168
B	177.71	4.663
A	178.91	5.871

Vehicle age	Mean	Std. Error
10+	129.85	7.404
8-9	192.35	8.227
4-7	255.50	5.976
0-3	281.89	6.620



4. Transcription

- System observes unstructured representation of some kind of data and transcribe the information into discrete textual form
- OCR: input is an image of text
  - asked to return this text in the form of a sequence of characters in ASCII/Unicode
    - Google Streetview uses deep learning for address nos.
- Speech recognition: an audio waveform is converted to word ID codes

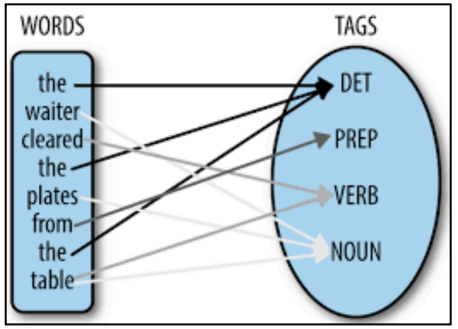
Google cloud speech API
Recognizes 110 language variants

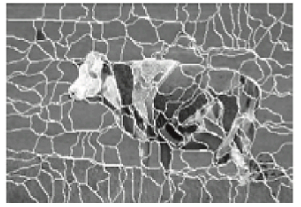
5. Machine Translation

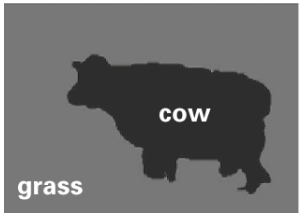
- Input already consists of a sequence of symbols in some language
- Computer program must convert it into a sequence of symbols in another language
- Commonly applied to natural languages, Ex:English to French
- Deep learning is having an important impact

6. Structured Output

- Output data structure has relationships between elements.
  - *Parsing*: map sentence into a tree
    - Nodes tagged as verbs, adverbs,..
  - *Pixel-wise segmentation*: every superpixel in image is assigned to specific category
  - *Image captioning*: observe an image and output a sentence describing the image
    - Words produced by captioning program must form a valid sentence

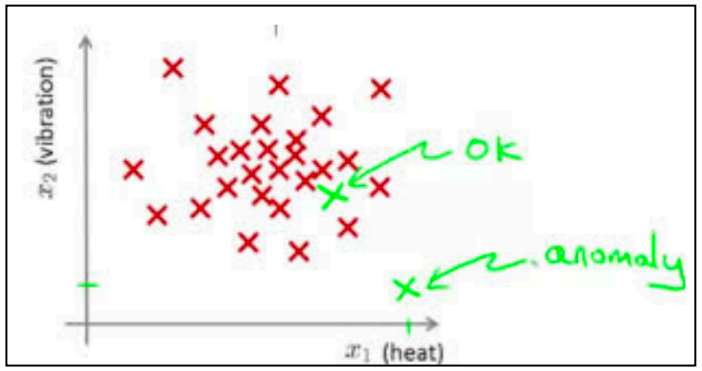






7. Anomaly Detection

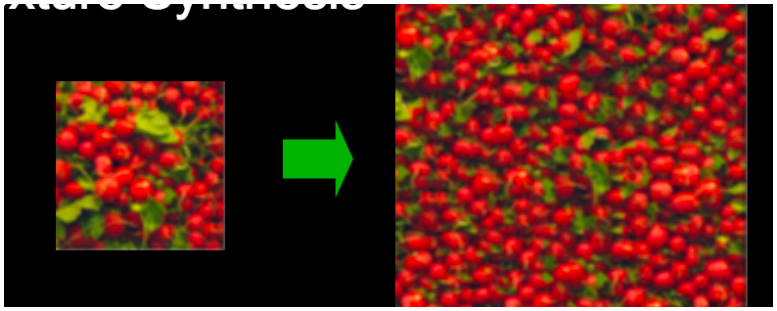
- Sift through a set of events or objects and flags some of them as being unusual or atypical



- Ex: credit card fraud detection
  - Model purchase habits to detect misuse of cards
    - Thief's purchases come from different probability distributions than your own

8. Synthesis and Sampling

- Generate new samples similar to those in training data
- Useful when generating large content by hand is expensive/boring/require too much time. Exs:
  - Videogames generate textures for large objects not requiring artist to manually label pixels



  - Speech synthesis: provide input sentence and ask to generate audio waveform containing a spoken version, but with large variation to seem natural

9. Imputation of Missing Values

- ML algorithm is given a new example $\boldsymbol{x} \in \mathrm{R}^n$
  - But with some entries x_i of $\boldsymbol{x}$ missing
  - Algorithm must predict values of missing entries

- In R, missing values are indicated by NA's. To see data in Social Indicators Survey (picking rows 91–95), we type R code

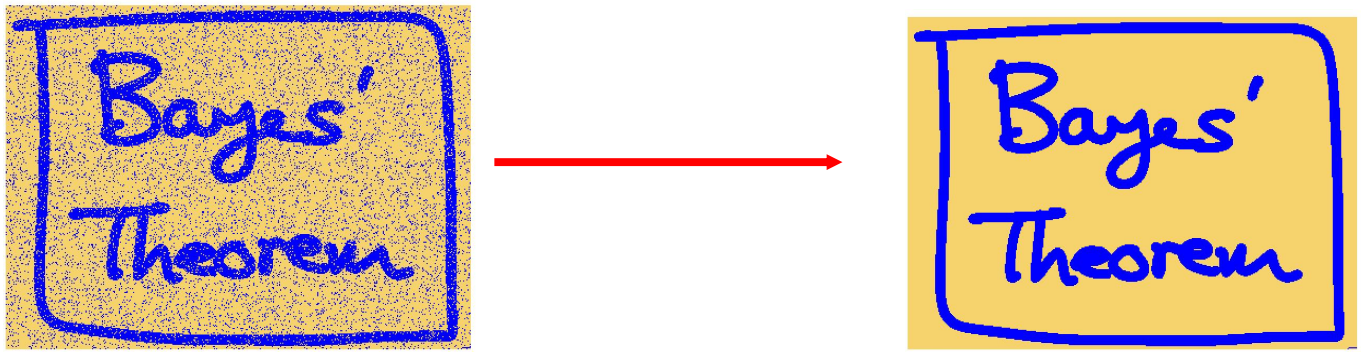
  cbind (sex, race, educ_r, r_age, earnings, police)[91:95,] and get R output

```
      sex race educ_r r_age earnings police
 [91,] 1 3 3 31 NA 0
 [92,] 2 1 2 37 135.00 1
 [93,] 2 3 2 40 NA 1
 [94,] 1 1 3 42 3.00 1
 [95,] 1 3 1 24 0.00 NA
```

- In regression R excludes cases in which inputs are missing;
- This limits information available in analysis, especially if there are many inputs with missingness.

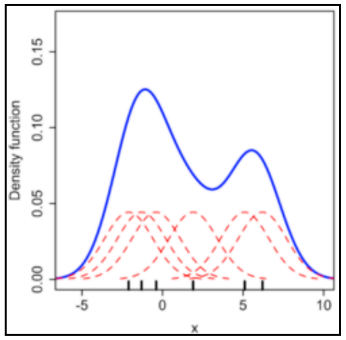
10. Denoising

- The ML algorithm is given an input a corrupted examples $\tilde{\boldsymbol{x}} \varepsilon \mathrm{R}^n$ obtained in an unknown corruption process
- Learner must predict the clean example $\boldsymbol{x}$ from its corrupted version $\tilde{\boldsymbol{x}}$
- Or more generally the conditional probability distribution $p(\boldsymbol{x}|\tilde{\boldsymbol{x}})$



11. Density Estimation

- Learn a function $p_{\text{model}}: \mathrm{R}^n \rightarrow \mathrm{R}$ where $p_{\text{model}}(\boldsymbol{x})$ is a pdf if $\boldsymbol{x}$ is continuous and pmf if discrete
  - It must know where examples cluster tightly and where they are unlikely



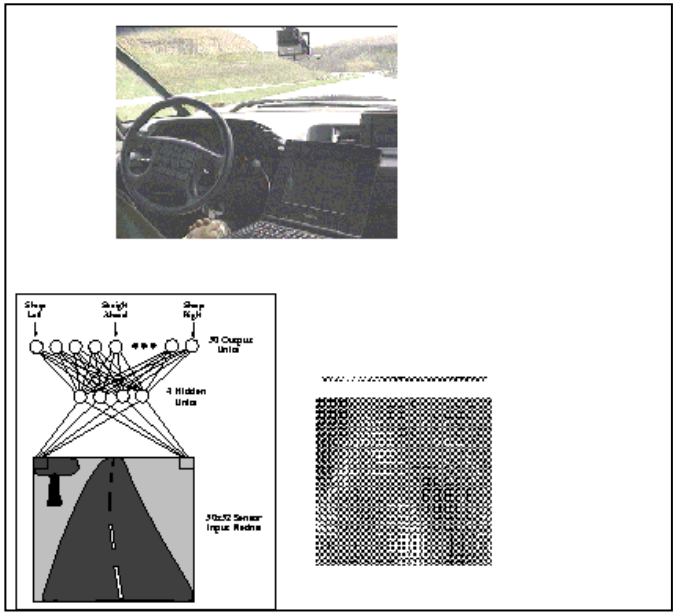
- If we explicitly capture distribution we can solve other tasks as well
  - If we can capture $p(\boldsymbol{x})$ we can solve missing value imputation task. If x_i is missing and values of $\boldsymbol{x}_{-i}$ are given then we get $p(x|\boldsymbol{x}_{-i})$

The Performance Measure, P

- Need measure of performance specific to task T
  - For classification, classification with missing inputs and transcription:
    - *Accuracy*: Proportion of samples for which correct output is produced
      - Equivalent to *error rate*, proportion of examples for which model produces incorrect output
        - » Called 0-1 loss, 0 loss if correct, 1 loss if incorrect
  - For density estimation
    - Average log probability assigned to some examples
- Usually on data not seen before, a test set
- Often difficult to choose a good measure

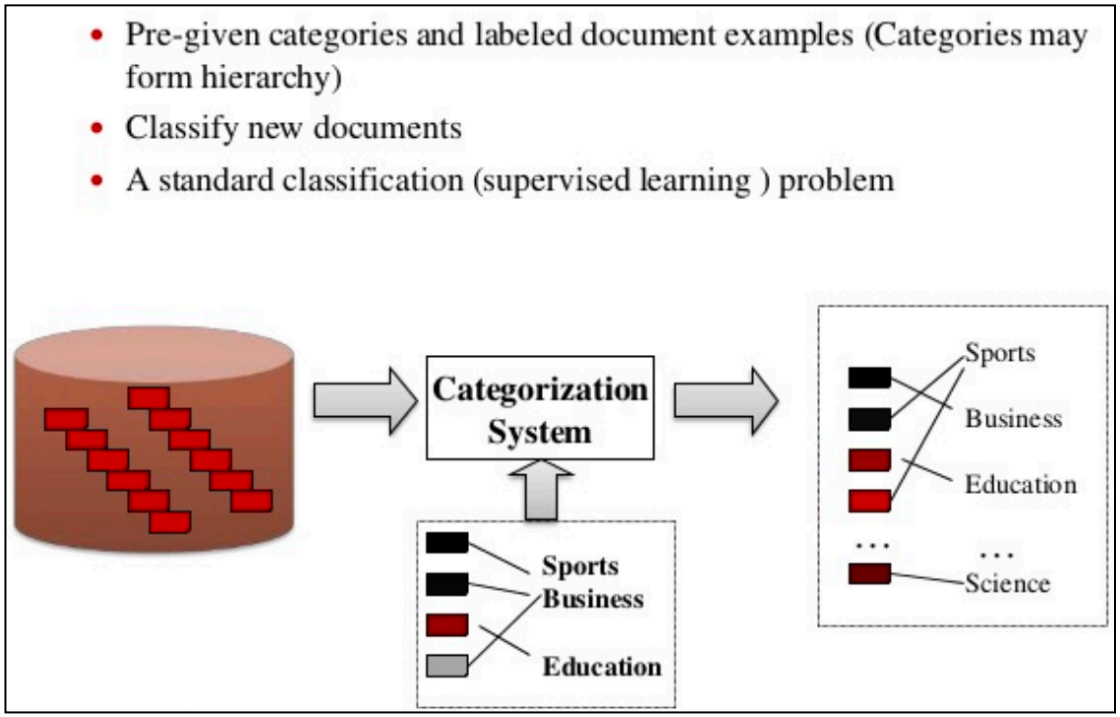
Example of Performance Measures

- Autonomous Vehicle Navigation
  - Task T : driving on public highway using vision sensors
  - Performance measure P : average distance traveled before an error (as judged by human overseer)
  - Training experience E : sequence of images and steering commands recorded observing a human driver



Example of Performance Measures

- Text Categorization Problem
  - Task T : assign a document to its content category
  - Performance measure P : Precision and Recall
  - Training experience E : Example pre-classified documents



The Experience, E

- Broadly categorized as unsupervised or supervised
  - Based on what kind of experience they are allowed to have during learning process
- Most algorithms experience a dataset
  - Also called data points

Example of a data set

- Anderson's Iris data (oldest set in stat/ML)
  - Measurements of 150 iris flowers
    - sepal length, sepal width, petal length, petal width
      - in 3 species: Setosa, versicolor, virginica



Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.3	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*

7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.5	2.3	4.0	1.3	*I. versicolor*
6.5	2.8	4.6	1.5	*I. versicolor*
5.7	2.8	4.5	1.3	*I. versicolor*
6.3	3.3	4.7	1.6	*I. versicolor*

6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*
6.3	2.9	5.6	1.8	*I. virginica*
6.5	3.0	5.8	2.2	*I. virginica*
7.6	3.0	6.6	2.1	*I. virginica*
4.9	2.5	4.5	1.7	*I. virginica*
7.3	2.9	6.3	1.8	*I. virginica*

Unsupervised learning experience

- Data set contains many useful features
- Learn structure of this data set
- Learn the entire probability distribution that generated this data set
  - Explicitly for density estimation
  - Implicitly for synthesis or denoising
- Perform a role such as clustering
  - i.e., dividing data set into clusters of similar examples

Supervised Learning experience

- Data set contains features, but each example is associated with a label or target
- Iris dataset is annotated with the species of each iris plant
- A supervised learning algorithm can learn to classify iris plants into three different species

Here is a test case

Blur between supervised/unsupervised

- Unsupervised: observe several examples of $\mathbf{x}$ and learn $p(\mathbf{x})$
- Supervised: observe examples of $\mathbf{x}$ and y and learn to estimate $p(y|\mathbf{x})$
- Many ML methods can be used for both tasks
  - E.g., Using chain rule of probability, for $\mathbf{x} \varepsilon R^n$

$$p(\mathrm{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, .., x_{i-1})$$

  - We can model $p(\mathbf{x})$ by splitting it into n supervised learning problems
  - Alternatively we can solve supervised problem of learning $p(y|\mathbf{x})$ by using unsupervised learning to determine $p(\mathbf{x},y)$ and then inferring $p(y|\mathbf{x})$

Semi-supervised Learning

- Some examples include a supervision target but others do not
- In *multi-instance learning*, an entire collection of samples is labeled as containing or not containing an example of a class, but individual members of the collection are not labeled

Reinforcement Learning

- Algorithms interact with environment, no fixed data set
  - A feedback loop between system and its experience
- Dog is given a reward/punishment for an action
  - Policies: what actions to take in a particular situation
  - Utility estimation: how good is state (→used by policy)
- No supervised output but delayed reward
  - Credit assignment
    - what was responsible for outcome
- Applications:
  - Game playing, Robot in a maze, Multiple agents, partial observability, …

Reinforcement learning (RL) connected to a deep neural network proves to be an effective solution for learning to navigate in complex environments without any prior knowledge. RL agent can infer from complex environments by punishment-reward system. It can model decision making process. Applications: AlphaGo Zero beat the world's professionals (December 2017). OpenAI presented a bot, that won in Dota2 world championship (Aug 2018). RL has also many applications in robotics.	Paper: "Playing Atari with Deep Reinforcement Learning" by Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin Riedmiller, NIPS 2013 (Cited by 1490) Dataset: Q-learning generates data exclusively from experience, without incorporation of the prior knowledge. If we put all our history data into a table with *state, action, reward, next state* and then sample from it, it should be possible to train our agent that way, without the dataset. Backend: Python3, Keras, Tensorflow Core libraries: OpenAI Gym, Keras - RL Code: https://github.com/nathanmargaglio/DQN
	Strategy: (1) estimate the discounted sum of rewards of taking action *a* in state *s* - ***Q(s, a)*** function, (2) choose the action with the maximum Q-value in any given state. $Q_{i+1}(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s',a') \mid s,a]$ where s - state; a - action to take; r - reward; y - discounting factor. An agent learns by getting positive or negative rewards. **Loss:** Huber Loss (modified MSE/MAE)

Reinforcement Learning: ATARI

Environment: BreakoutDeterministic-v4

Backend: Keras, Python3

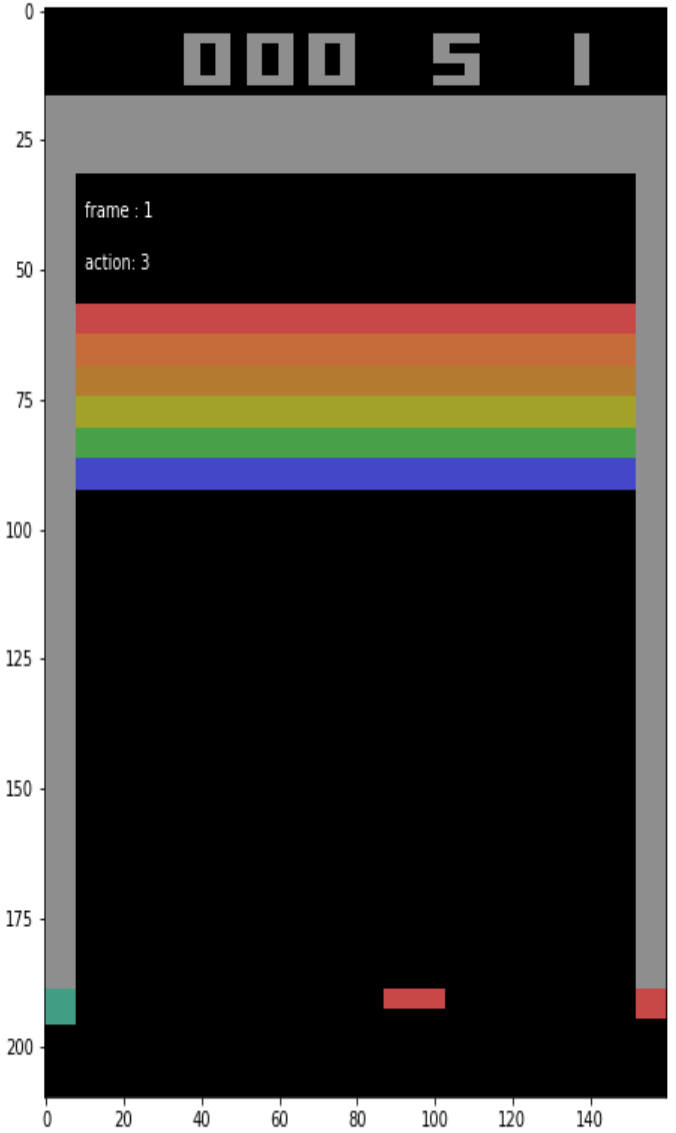
Libraries: OpenAI Gym, Keras-RL

Reward: max score - 208 (the benchmark in the paper 225)

Preprocessing: original image was downsampled from 210×160 pixel images to 105×80 and converted from RGB to gray-scale to decrease the computation

Training time: 15 hours including simulation time on a GTX 650 with 1 GB of RAM

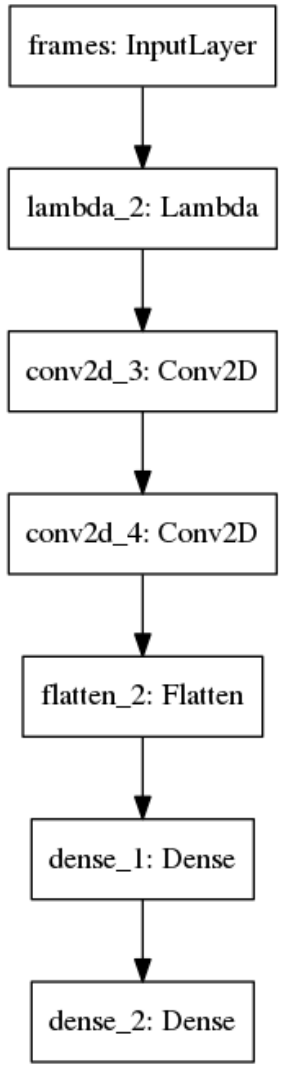
Notations:
Frame - a snapshot of the environment state at every point
Action (a) - a set of actions, that agent can take {0, 1, 2, 3}
Upper left corner - score (our evaluation metric)
Upper middle - number of "lives" for each game (initially 5)
Upper right corner - might be version



Network structure & parameters

Layer (type)	Output Shape	Param #
frames (InputLayer)	(None, 4, 105, 80)	0
lambda_2 (Lambda)	(None, 4, 105, 80)	0
conv2d_3 (Conv2D)	(None, 16, 25, 19)	4112
conv2d_4 (Conv2D)	(None, 7, 11, 32)	9760
flatten_2 (Flatten)	(None, 2464)	0
dense_1 (Dense)	(None, 256)	631040
dense_2 (Dense)	(None, 4)	1028

Total params: 645,940
Trainable params: 645,940
Non-trainable params: 0



Describing a data set

- Common way of describing a data set is with a *design matrix*
- Different examples in each row
- Each column corresponds to a different feature
- Iris dataset contains 150 examples with four features for each example
  - Data set is a design matrix $\mathrm{X} \in \mathrm{R}^{150 \times 4}$
  - $\mathrm{X}_{i,1}$ is sepal length of plant i, $\mathrm{X}_{i,2}$ is sepal width of plant i

Dealing with varying sizes

- For a design matrix, each example is a vector of same size
  - But photos of varying size contain different nos. of pixels
- Rather than describing the matrix with *m* rows we describe it as a set of *m* elements $\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},..,\boldsymbol{x}^{(m)}\}$. It does not imply that vectors $\boldsymbol{x}^{\{i\}}$ and $\boldsymbol{x}^{\{j\}}$ have the same size
  - Instead of multiplying by a weight matrix of fixed size, convolution with a kernel is applied different no. of times depending on size of input