DAT1

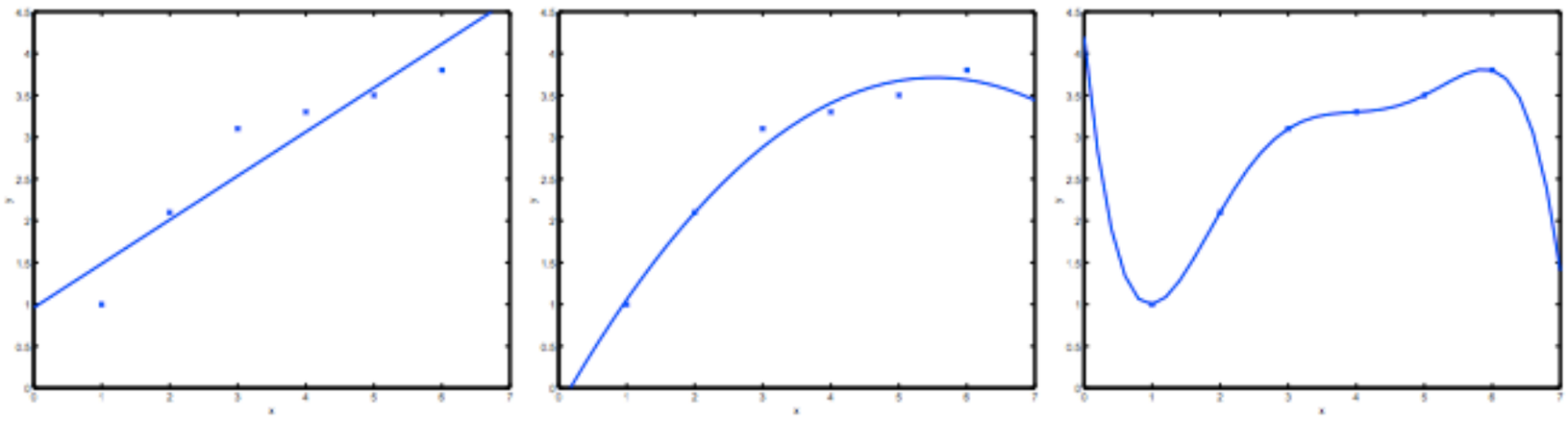
26/3/16

Agenda

- Learning Theory
- Reinforcement Learning Intro
- Spark
- (Recommender Practice)
- Random Topics

Learning Theory

Bias / Variance: parameters, models



How much data needed?

Which models to even consider?

Learning Theory

Lemma. (The union bound). Let $A_1, A_2, \ldots, A_k$ be k different events (that may not be independent). Then

$$P(A_1 \cup \cdots \cup A_k) \leq P(A_1) + \ldots + P(A_k).$$

Lemma. (Hoeffding inequality) Let $Z_1, \ldots, Z_m$ be m independent and identically distributed (iid) random variables drawn from a Bernoulli(ϕ) distribution. I.e., $P(Z_i = 1) = \phi$, and $P(Z_i = 0) = 1 - \phi$. Let $\hat{\phi} = (1/m) \sum_{i=1}^{m} Z_i$ be the mean of these random variables, and let any $\gamma > 0$ be fixed. Then

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2 \exp(-2\gamma^2 m)$$

How many training points (samples) to be within 0.01 of each other with probability 95%?

Learning Theory

Binary classification empirical error

$$\hat{\varepsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\}.$$

(fraction misclassified)

Generalisation error

$$\varepsilon(h) = P_{(x,y)\sim\mathcal{D}}(h(x) \neq y).$$

Learning Theory

PAC framework
“probably approximately correct”

i.e. training and testing from same distribution

Learning Theory

Minimising training error

$$\hat{\theta} = \arg\min_{\theta} \hat{\varepsilon}(h_\theta).$$

$$\mathcal{H} = \{h_\theta : h_\theta(x) = 1\{\theta^T x \geq 0\}, \theta \in \mathbb{R}^{n+1}\}$$

equivalent to $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$

Learning Theory

Finite $\mathcal{H} = \{h_1, \ldots, h_k\}$

We want guarantees on generalisation error when we minimise empirical error

- theoretical framework
- applied cases e.g. too few data points for cross-validation

Learning Theory

- Take fixed h_i in H

- Let $Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}$.

- Training error of h_i

$$\hat{\varepsilon}(h_i) = \frac{1}{m}\sum_{j=1}^{m} Z_j.$$

- Hoeffding

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 m).$$

Learning Theory

Great. But we don't just want this for one h_i, we want it for all of H simultaneously

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 m).$$

$P(A_i)$

$$\begin{aligned} P(\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\ &\leq \sum_{i=1}^{k} P(A_i) \\ &\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\ &= 2k\exp(-2\gamma^2 m) \end{aligned}$$

Learning Theory

$$
\begin{aligned}
P(\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\
&\leq \sum_{i=1}^{k} P(A_i) \\
&\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\
&= 2k\exp(-2\gamma^2 m)
\end{aligned}
$$

If we subtract both sides from 1, we find that

$$
\begin{aligned}
P(\neg\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(\forall h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \\
&\geq 1 - 2k\exp(-2\gamma^2 m)
\end{aligned}
$$

Learning Theory

Example

For instance, we can ask the following question: Given γ and some $\delta > 0$, how large must m be before we can guarantee that with probability at least $1 - \delta$, training error will be within γ of generalization error? By setting $\delta = 2k \exp(-2\gamma^2 m)$ and solving for m, [you should convince yourself this is the right thing to do!], we find that if

$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta},$$

then with probability at least $1 - \delta$, we have that $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in \mathcal{H}$. (Equivalently, this shows that the probability that $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$ for some $h \in \mathcal{H}$ is at most δ.) This bound tells us how many training examples we need in order make a guarantee. The training set size m that a certain method or algorithm requires in order to achieve a certain level of performance is also called the algorithm's **sample complexity**.

Learning Theory

With just a bit more effort, we can get
an explicit guarantee on generalisation error

Theorem. Let $|\mathcal{H}| = k$, and let any m, δ be fixed. Then with probability at least $1 - \delta$, we have that

$$\varepsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \varepsilon(h) \right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

lower bias: increasing H class

variance tradeoff of increasing H class

Learning Theory

In summary, given a hypothesis class of complexity k, we can figure out how many training examples we need to be within a certain distance from minimising generalisation error when carrying out empirical risk minimisation

Analog for continuous case:
Vapnik-Chervonenki (VC) dimension

Reinforcement Learning

Previously, y had a “correct” value

Now instead, what if the best we can provide is a reward function

e.g. robot motion, gaming…trading?

https://www.youtube.com/watch?v=qv6UVOQ0F44

https://www.youtube.com/watch?v=M8YjvHYbZ9w

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

Reinforcement Learning

A Markov decision process is a tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where:

- S is a set of **states**. (For example, in autonomous helicopter flight, S might be the set of all possible positions and orientations of the helicopter.)

- A is a set of **actions**. (For example, the set of all possible directions in which you can push the helicopter's control sticks.)

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s.

- $\gamma \in [0, 1)$ is called the **discount factor**.

- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).

Reinforcement Learning

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \ldots$$

$s_1 \sim P_{s_0 a_0}$.

Upon visiting the sequence of states $s_0, s_1, \ldots$ with actions $a_0, a_1, \ldots$, our total payoff is given by

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots .$$

Or, when we are writing rewards as a function of the states only, this becomes

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots .$$

Reinforcement Learning

We want to maximise expected discounted reward

Economic interpretation of gamma?

$$\mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots\right]$$

Reinforcement Learning

Policy to choose actions

$$\pi : S \mapsto A$$

$$a = \pi(s)$$

Value function based on policy

$$V^{\pi}(s) = \mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, \pi\right].$$

Reinforcement Learning

$$V^{\pi}(s) = \mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, \pi\right].$$

Bellman equation

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s').$$

Can you draw this (perhaps as a decision tree?)

Reinforcement Learning

We want to find the optimal value

$$V^*(s) = \max_\pi V^\pi(s).$$

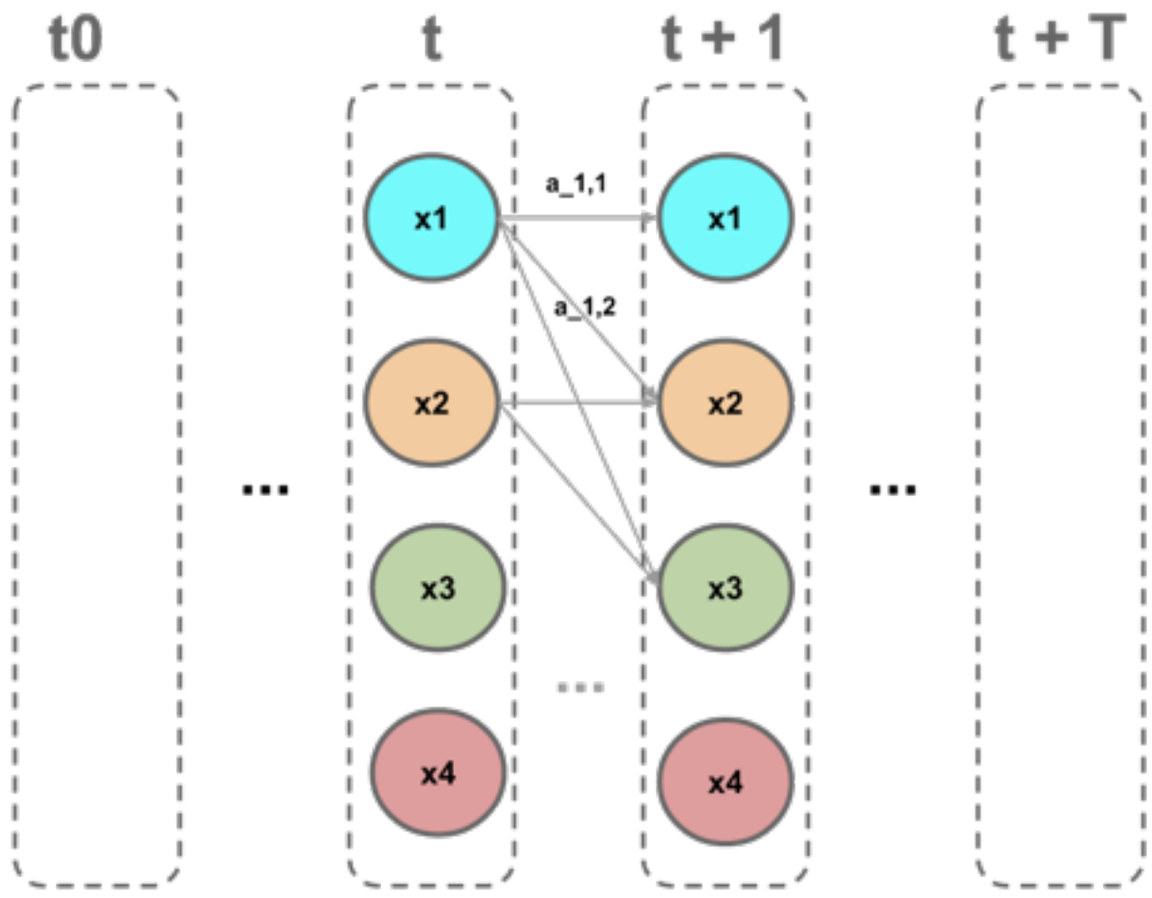
or

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s').$$

max over possible actions

Reinforcement Learning

Exponential paths to explore



Hansel and Grettel!

Reinforcement Learning

Recommended for Dynamic Programming and Value Iteration:

https://en.wikipedia.org/wiki/Memoization

https://www.youtube.com/watch?v=oefOCk3koZo

https://www.youtube.com/watch?v=ip4iSMRW5X4

Reinforcement Learning

Extensions:

Learning P_sa

Continuous space (discretisation)

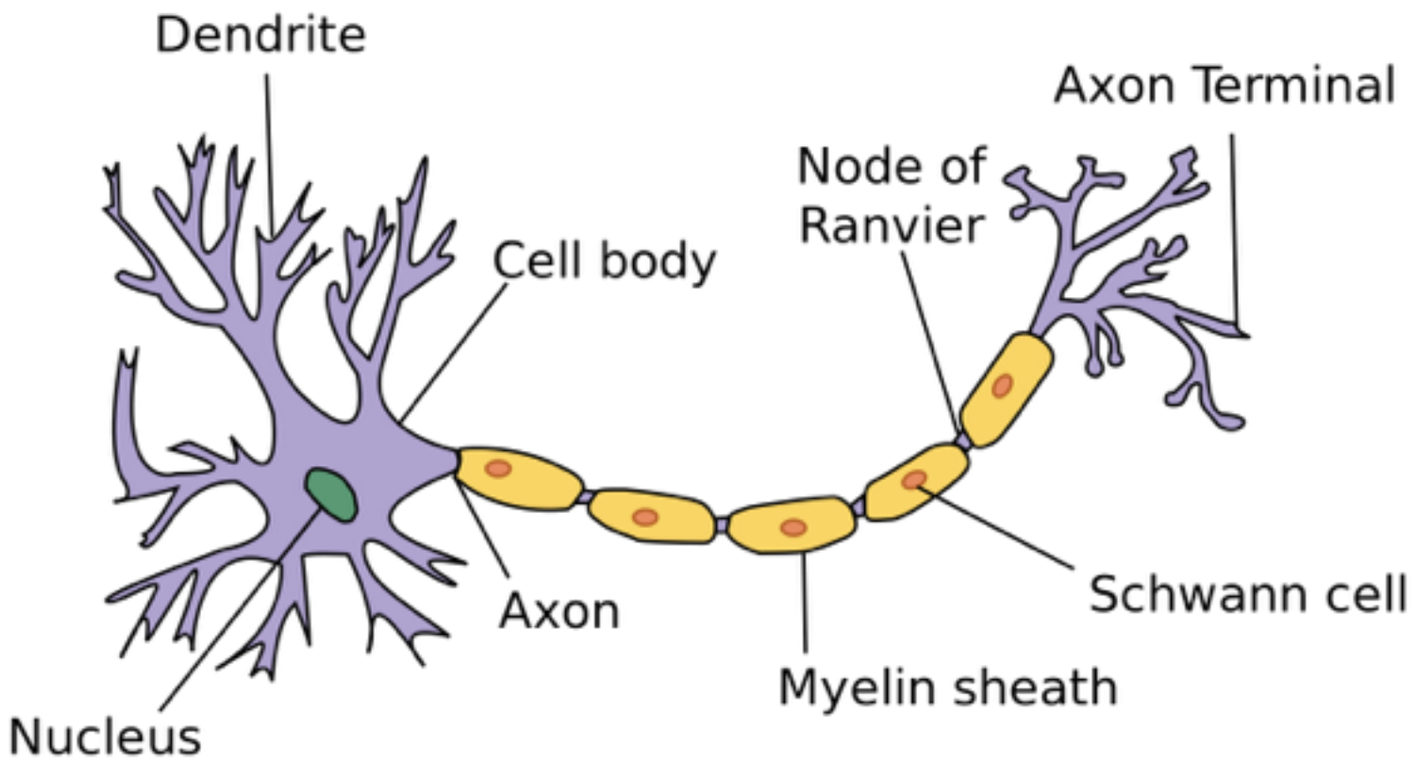
SPARK

Random Topics
Neural Networks

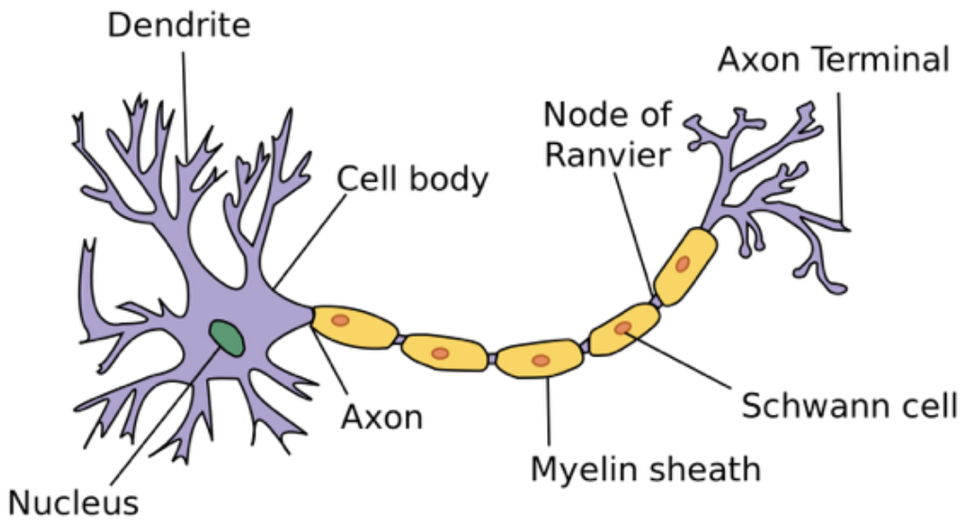
A computational system
comprised of layers and each layer
is built of interconnected perceptrons

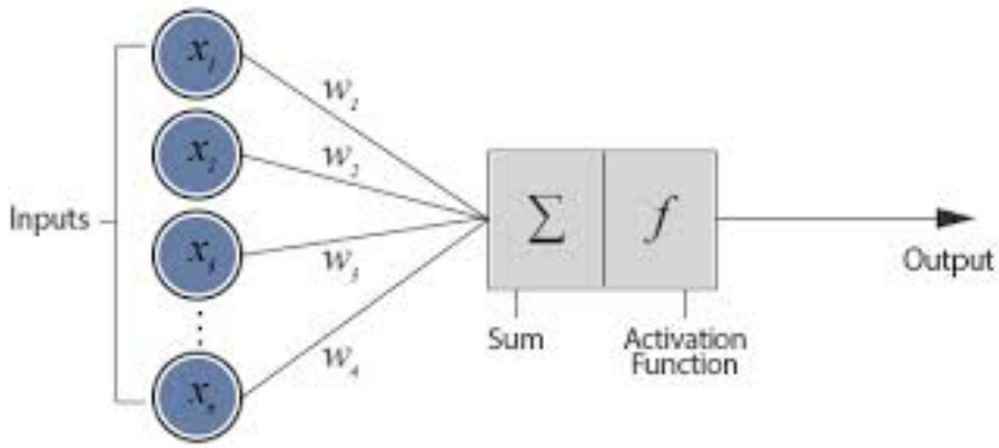
Built to model the animal nervous system?

Random Topics
Neural Networks



Random Topics
Neural Networks



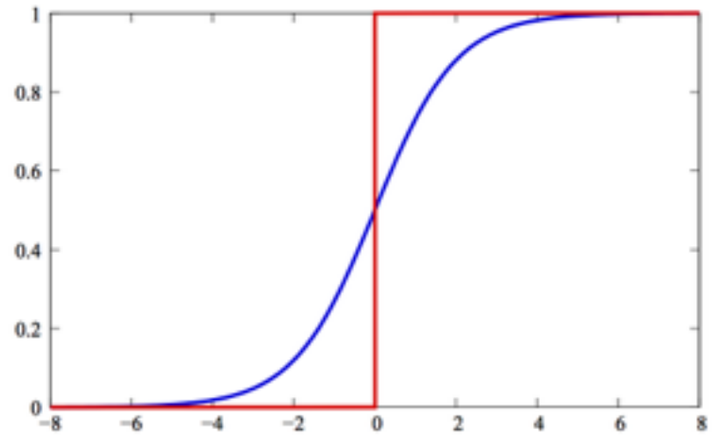


Random Topics
Neural Networks

Single Perceptron

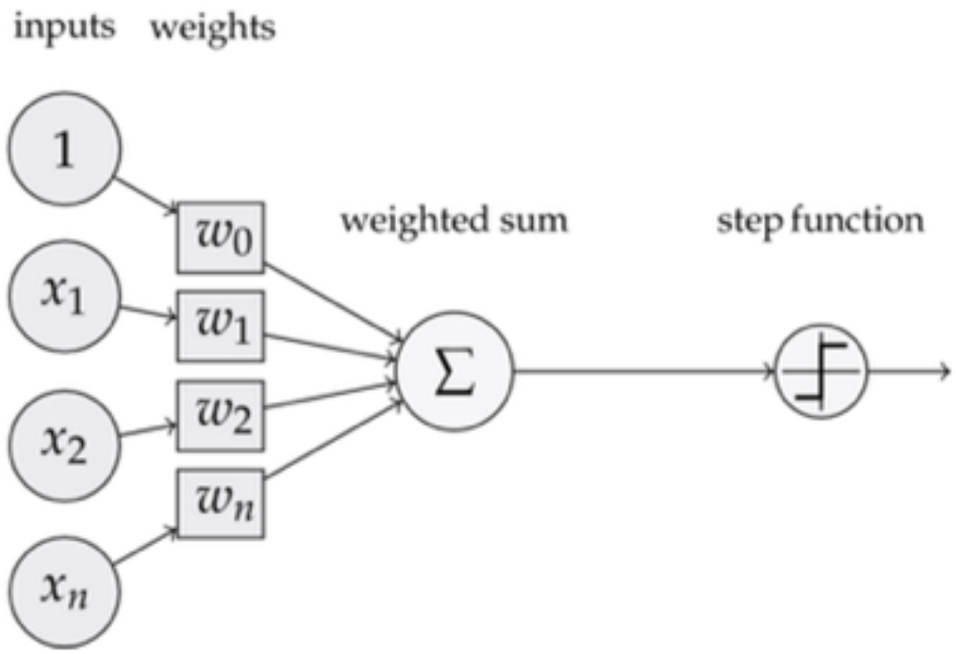
$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

f_{log} is called logistic function



Takes in input and uses an activation function in order to output

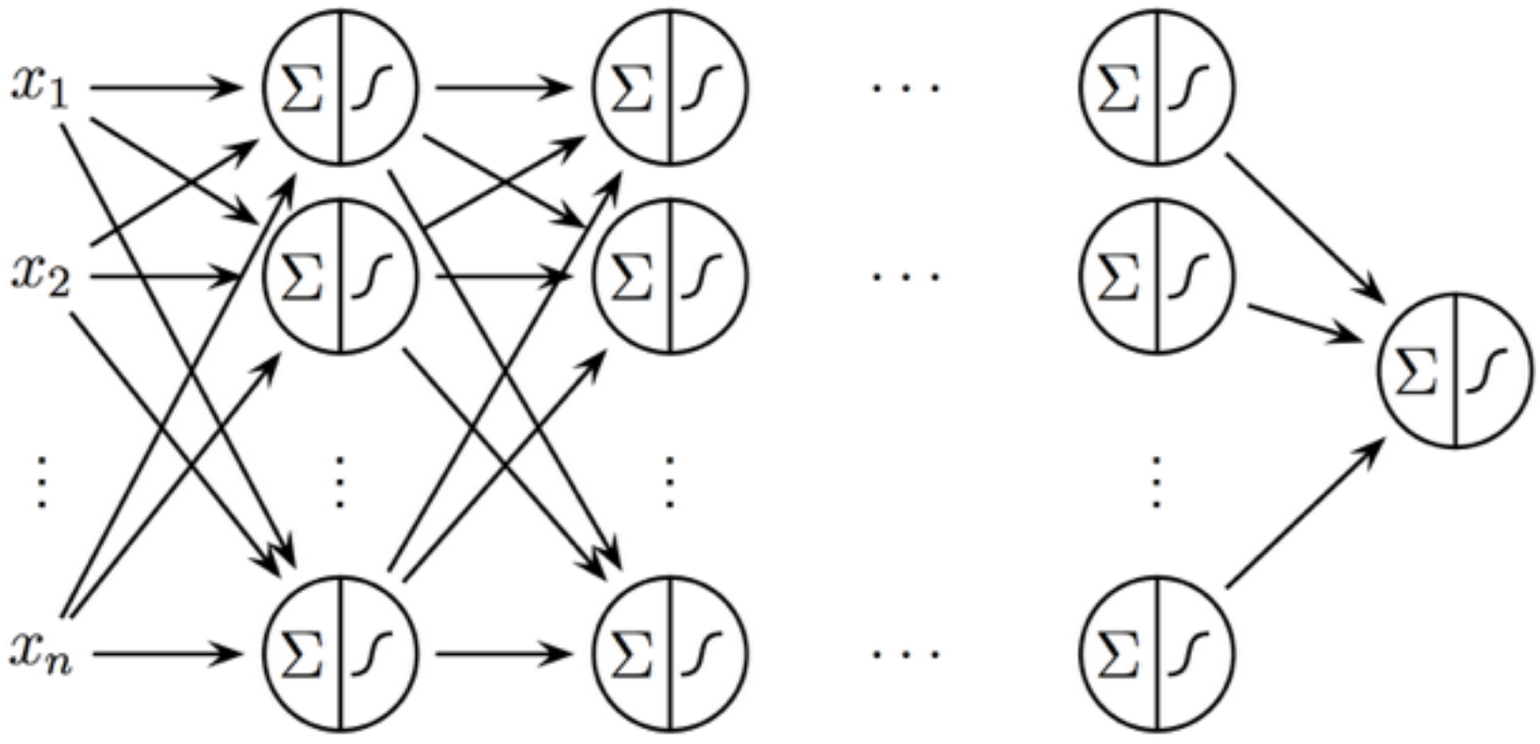
Random Topics
Neural Networks



Random Topics
Neural Networks

Artificial Neural Networks are also known as multi layer perceptrons

A multi layer perceptrons (MLP) is a finite acyclic graph. The nodes are neurons with logistic activation.



Random Topics
Neural Networks

But how does it learn?!

Back-Propagation

https://en.wikipedia.org/wiki/Backpropagation

Random Topics
Neural Networks

Pros

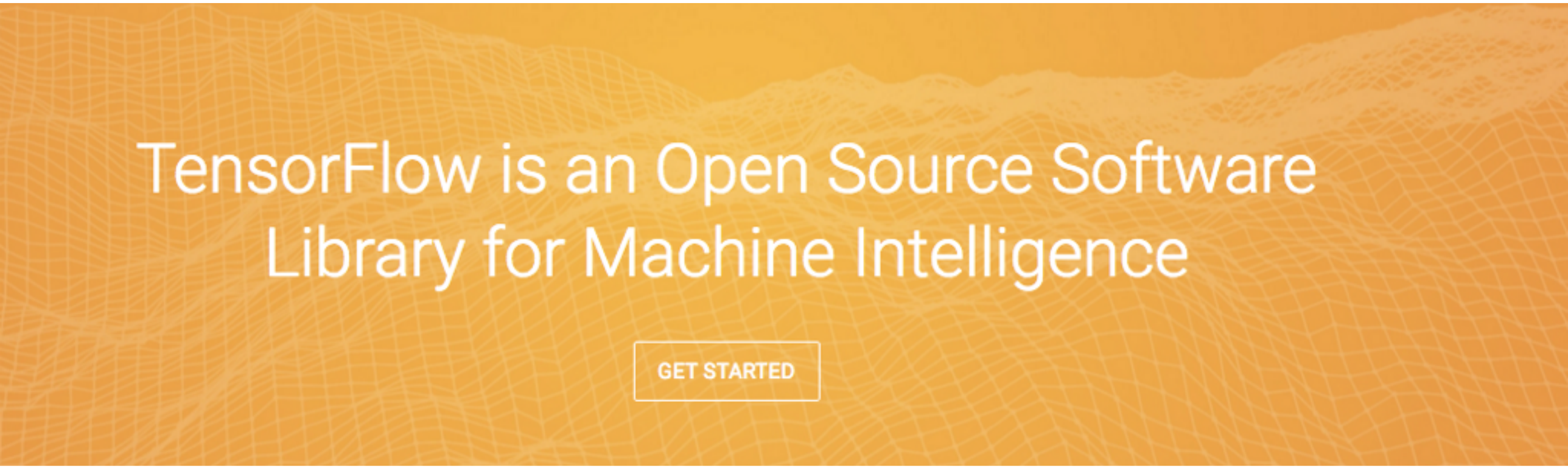
- Online model (updates as you go)
  - Doesn't need to be fit all of the time
- Very fast predictions
- Can approximate almost any type of function
- Can be used in a supervised and unsupervised manner
- Super cool

Cons

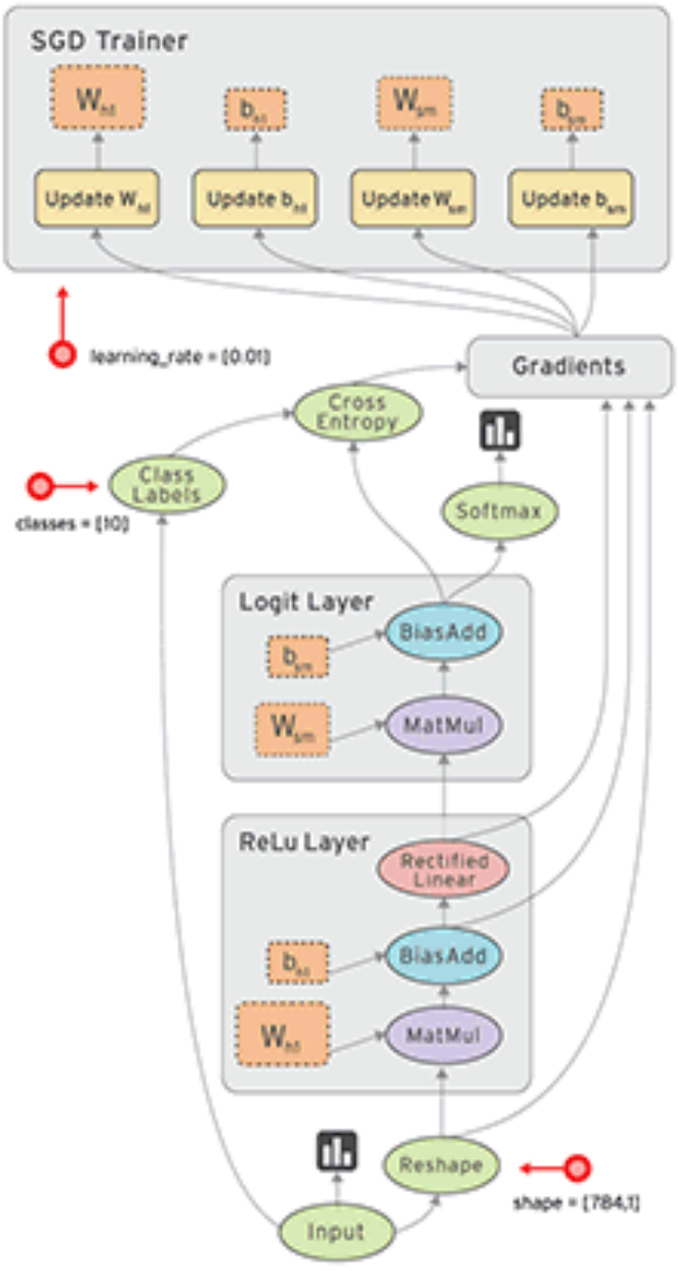
- Requires many training samples to be considered good
- Hard to describe what is happening
- Requires a lot of hardware / computation power
- Slow to train
- Can be difficult to use

Random Topics
Neural Networks

https://www.tensorflow.org/



Random Topics
Neural Networks



Random Topics

Web Scraping

Python Beautiful Soup

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

Exercise: scrape this

list of buyers

http://econpy.pythonanywhere.com/ex/001.html

finance

http://stackoverflow.com/questions/10040954/alternative-to-google-finance-api

Random Topics
Web Scraping

Concepts

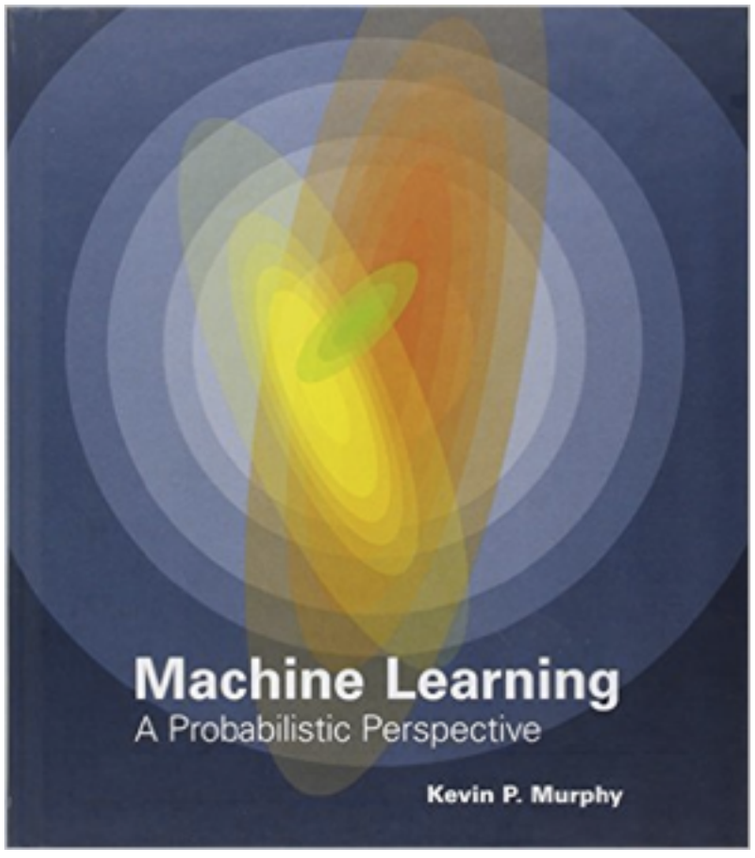
- Robustness: changes in the web page/api
- Scale: massive web scraping fault tolerance
- Integrity: external data sources

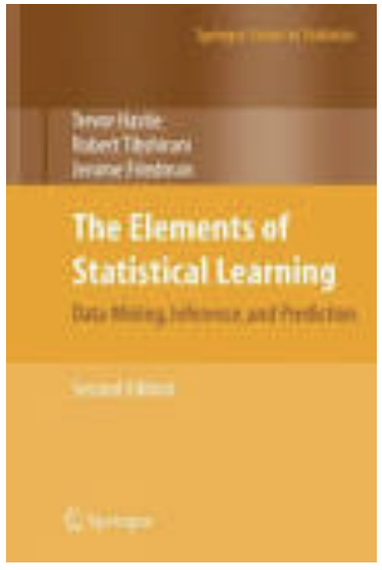
Random Topics
Setting up servers

1. AWS account
2. ec2 instance
3. ssh into it

1. heroku account
2. python “hello” api
3. push to heroku

Random Topics
Resources Going Forward





Random Topics

Resources Going Forward

Michael Nielsen, Christopher Olah
http://colah.github.io/

Geoffrey Hinton

Yann LeCun

Random Topics

Resources Going Forward

Andrej Karpathy

Hugo Larochelle

Random Topics

Resources Going Forward

Linear Algebra

Probability

Parallel Programming

Computer Science

Topology

Information Theory