The Evolution of AI & ML

Applications in different domains

Jaideep Ganguly, ScD (MIT)
Head, Compass India Development Center

Distinguished speaker talk at Indian School of Business (ISB)

January 24, 2020

About Jaideep

1. Head of Compass India Development Center.
2. compass.com is a startup, the India center started in December 2019.
3. *Compass is building the first modern real estate software platform to make search and sell experience intelligent and seamless.*
4. Director of Software Development, Amazon [2010 - 2019].
5. Director of Engineering, Microsoft [2006 - 2010].
6. Doctor of Science, Massachusetts Institute of Technology (MIT).
7. Master of Science, Massachusetts Institute of Technology (MIT).
8. Bachelor of Technology (Honours), Indian Institute of Technology, Kharagpur.

Can a Computer Think?

1. The human brain is a remarkable organ. It has enabled us understand science and advance mankind.
2. The idea of mimicking the human brain or even improving the human cognitive functions is an alluring one and is an objective of Artificial Intelligence research.
3. But we are not even close in-spite of a century of research. However, it continues to have a major hold on our imagination given the potential of the rewards.
4. *The question of whether a computer can think is no more interesting than the question of whether a submarine can swim" - Dijkstra*
5. It is more interesting to understand the evolution of Machine Learning - *How did it start, here are we today and where do we go from here.*

Symbols & Reasoning

1. About 50,000 years ago, palaeontologists believe that some of us (1,000?), were able to deal with symbols - a major step in evolution.
2. Noam Chomsky (MIT) thinks we were then able to create a new concept from 2 existing ideas without limiting the existing concepts.
3. Around 350 BC, Aristotle devised syllogistic logic, the first formal deductive reasoning system to model the way humans reason.
4. 2,000 years later, Bertrand Russel (Cambridge Univ) & his student Alfred Whitehead (Imperial College & Harvard) published Principia Mathematica that laid down the foundations of Mathematics.
5. John McCarthy (Stanford) championed mathematical logic in AI.
6. In 1942, Alan Turing (Cambridge Univ) showed that any form of mathematical reasoning could be processed by a machine.
7. By 1967, Marvin Minsky (MIT) declared that "within a generation, the problem of creating Artificial Intelligence would substantially be solved".
8. Clearly we are not there yet!

Reasoning

1. In 1961, Minsky published Steps towards Artificial Intelligence and talked about search, matching, probability, learning.
2. In 1986, Minsky wrote the book The Society of Mind and talked about heuristic programming.
3. *What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle. — Marvin Minsky, The Society of Mind, p. 308*
4. McCarthy championed mathematical logic for artificial intelligence.
5. Newel (CMU) argued that the mind functions as a single system. He also claims the established cognitive models are vastly underdetermined by experimental data.
6. Simon (CMU) expoused the idea of bounded rationality, that rationality is limited. When individuals make decisions, by the tractability of the decision problem, the cognitive limitations of the mind, and the time available to make the decision.

Heuristic Reasoning & Model Backed Systems

1. In AI, model-based reasoning refers to an inference method used in expert systems based on a model of the physical world. The focus of application development is developing the model.
2. Terry Winograd (MIT) built a model backed system for dialog understanding (SHRDLU).
3. Patrick Winston (MIT) built a model backed system for learning.
4. Gerald Sussman (MIT) built a model backed system for understanding blocks.
5. Roger Schank (Yale), in his Case-based reasoning, believed that understanding stories is the key to modeling human intelligence.
6. David Marr (MIT) treated vision as an information processing system. His hypothesis comprised of a computational level - what does the system, how does it do and a physical level - what neural structures and neuronal activities implement the visual system.

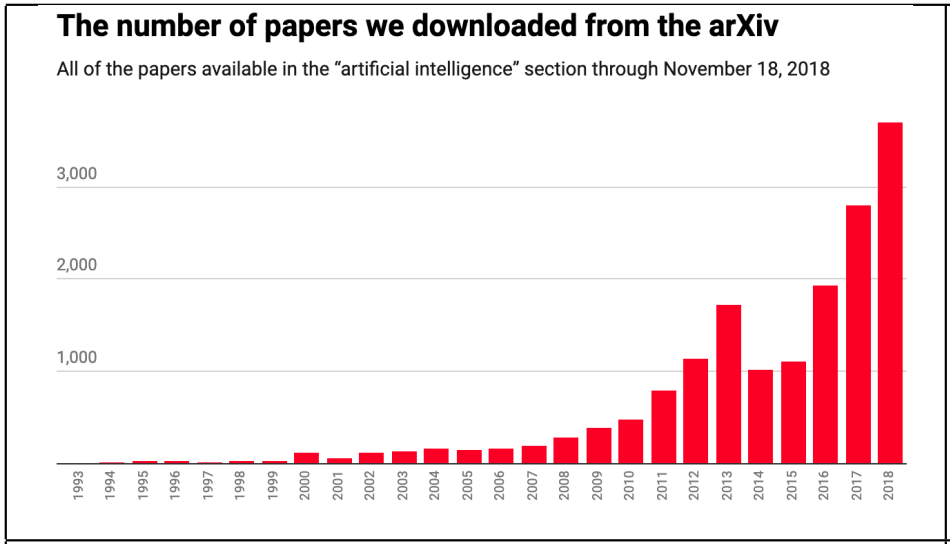
Knowledge Based (KB) Expert Systems

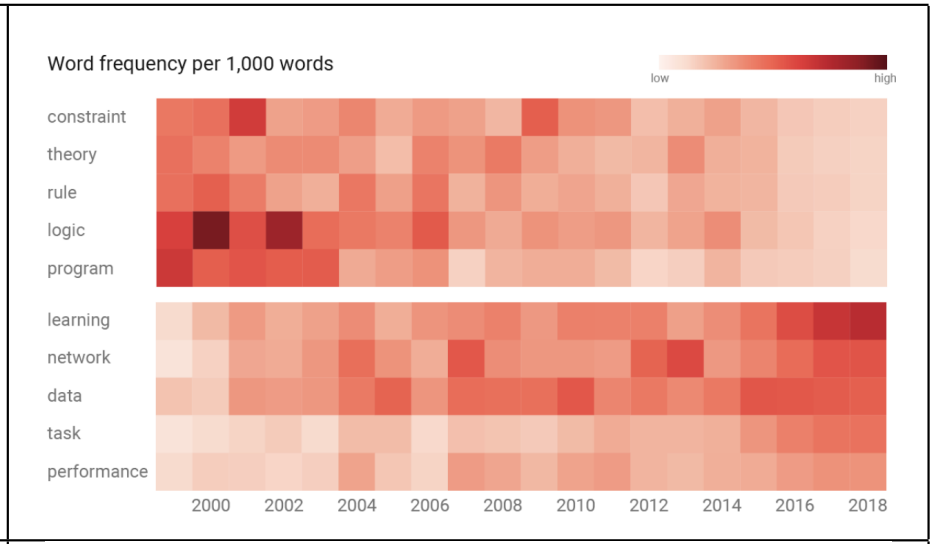
1. The field of AI was defined as computers performing tasks that were specifically thought of as something only humans can do.
2. In the 1980s, the expert systems were of great interest and focused on knowledge and inference mechanisms. They did a good job in their domains but were narrow in specialization and were difficult to scale.
3. Once these systems worked, they were no longer considered to be AI! For example, today the best chess players are routinely defeated by computers but chess playing is no longer really considered as AI! McCarthy referred to as the "AI effect". IBM's Watson is one such program at a level such as that of a human expert.
4. Fifty years ago Jim Slagle's (MIT) symbolic integration program (MACSYMA) was a tremendous achievement.
5. It is very hard to build a program that has "common sense" and not just narrow domains of knowledge.

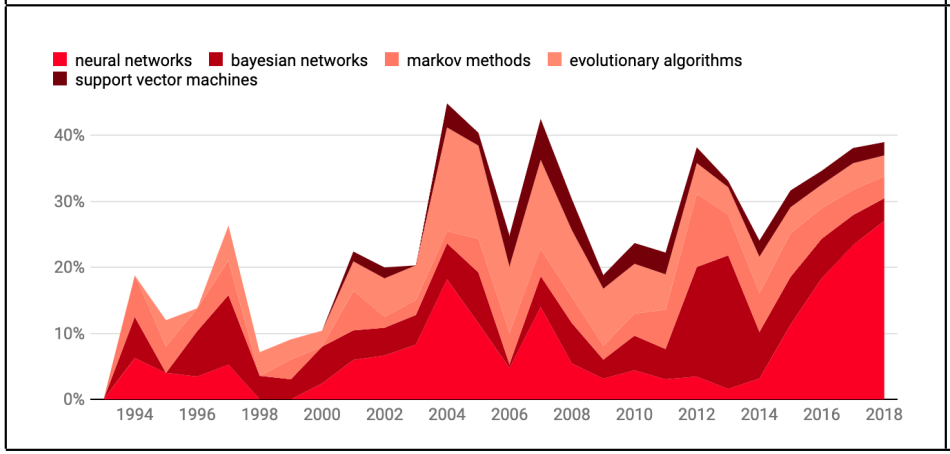
From Perceptrons to Deep Learning

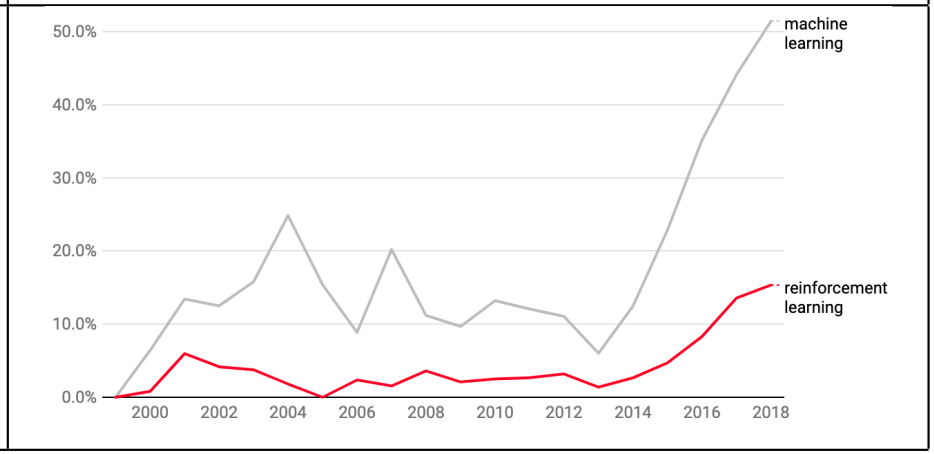
1. Rosenblatt (Cornell) is credited with the concept of Perceptrons, "a machine which responds like the human mind" as early as in 1957.
2. In a critical book written in 1969, Marvin Minsky and Seymour Papert showed that Rosenblatt's original system was blind to simple XOR.
3. That was incorrect and the field of "Neural Networks" disappeared!
4. Geoff Hinton (Toronto Univ) built more complex networks of neurons that allowed networks to learn more complicated functions like XOR.
5. But they learned slowly and couldn't master some of the basic things. By late 1990s, neural networks had again begun to fall out of favor.
6. In 2006, Hinton developed Deep Learning which extends earlier important work by Yann LeCun (New York Univ).
7. Deep learning's important innovation is to have models learn categories incrementally, attempting to nail down lower-level categories (like letters) before attempting to acquire higher-level categories (like words).

Trends - Source MIT Technology Review







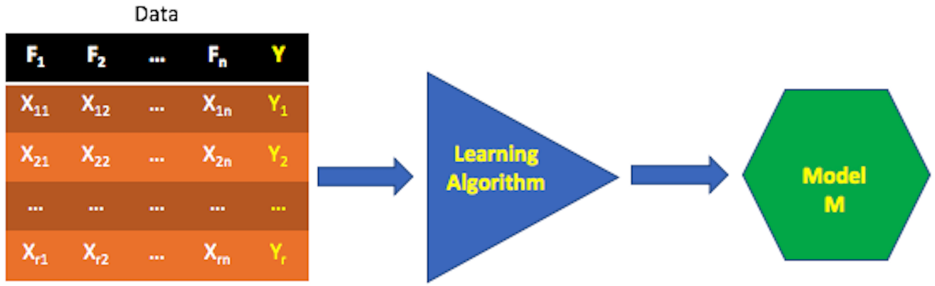


Deep Learning (DL)

1. DL powers Amazon's Alexa, Google's search, Facebook's news feed and Netflix's recommendation engine. Instead of manually encoding 1,000's of rules, the DL automatically extracts the rules from data.
2. Other techniques such as Bayesian Networks, Support Vector Machines, etc., take different approaches to finding patterns in data.
3. In 2012, during the ImageNet competition in computer vision, Hinton achieved the best accuracy in image recognition by an astonishing margin of more than 10% using DL and sparked renewed research.
4. Recently, Reinforcement Learning, which mimics the process of training through punishments and rewards, is in focus. In October 2015, DeepMind's AlphaGo, defeated the world champion in Go.
5. Every decade has seen focus on a different technique: Neural Networks in the late '50s and '60s, various symbolic approaches in the '70s, KB systems in the '80s, Bayesian Networks in the '90s, Support Vector machines in the '00s, and Neural Networks again in the '10s.
6. Nobody knows how to solve this problem.

What is Machine Learning (ML)?

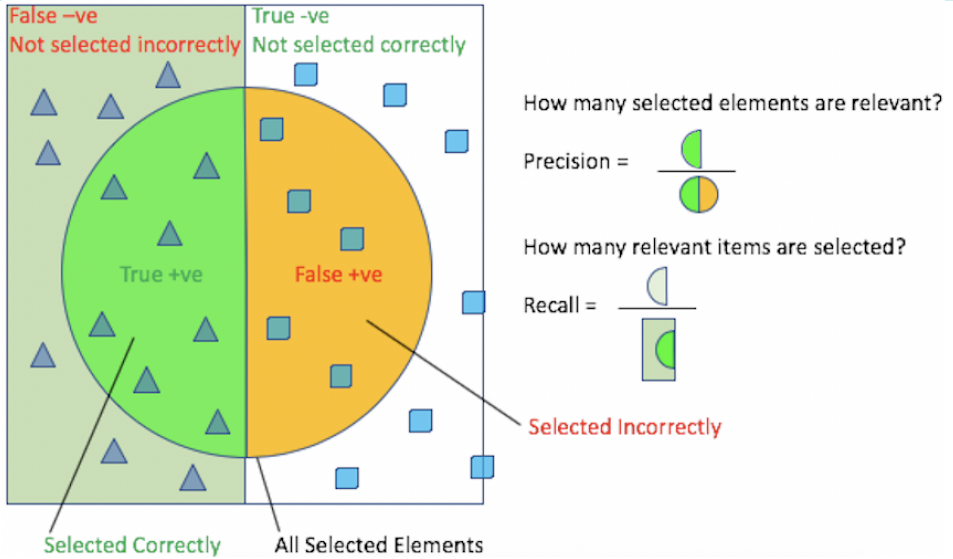
1. In 1959, Arthur Samuel (MIT), defined ML, a subset of AI, as a *"field of study that gives computers the ability to learn without being explicitly programmed".*



2. ML algorithms establishes a relationship between input X to output Y. A feature is an individual measurable attribute of a phenomenon that is being observed. Given training examples (X_i, Y_i) where X_i is the feature vector and Y_i the target variable, learn a function F i.e., $Y_i \approx F(X_i)$ for all i. For a new sample X predict Y using $F(X)$.
3. ML is effective for complex tasks where deterministic solution don't suffice, e.g., speech recognition, handwriting recognition, spam, fraud detection, etc. These cannot be solved manually in a large scale.

Types of Machine Learning & Basic Definitions

1. ML algorithms are classified as Supervised or Unsupervised.
2. In supervised learning, the machine learns from data that has already been tagged with the correct answer by an expert. This data is typically termed as the training set. E.g., separating spam mail.
3. In unsupervised learning, the machine groups unsorted information according to similarities without any training data. The machine automatically figures out the patterns on its own. E.g., playing GO.



$$Precision = \frac{t_p}{t_p + f_p}$$

$$Recall = \frac{t_p}{t_p + f_n}$$

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$

$$Harmonic\ Mean = F_1 = \frac{1}{2} \times \left(\frac{1}{Precision} + \frac{1}{Recall} \right)$$

Linear Regression

1. Regression is a statistical approach to find the relationship between variables between X_i and Y_i.
2. A common function used to model training data is a linear regression model. The model or the hypothesis is given by:

$$y_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} \cdots + \theta_n x_{in} \tag{1}$$

where x_{ij} is the observed value of the feature x_j, y_i is the predicted value of the outcome and θ_i are constants the values of which need to be determined. For now, we limit the the values of the features x_{ij} to numbers, positive or negative. Later on, we will study techniques on how to deal with the situation where the feature value is a string.

$$\text{Squared Error Loss (L)} = \frac{1}{2} \times \sum_{i=1}^{r} (\hat{y}_i - y_i)^2 \tag{2}$$

This loss function is always positive, there is a minimum, is monotonically increasing from that minimum value in both positive and negative directions. Such a function is called a convex function.

3. We need to minimize some loss function over the training data.

Gradient Descent

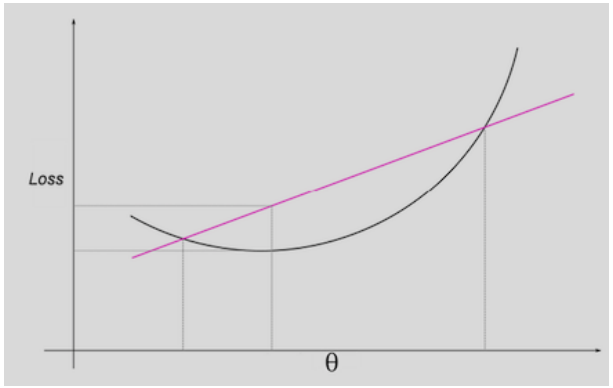


Figure: Convex Function

1. Minimize by setting the partial derivative of the loss wrt θ_j to zero.

$$\frac{\partial L}{\partial \theta_0} = \sum_{i=1}^{r} (\hat{y}_i - y_i) = 0 \quad \frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{r} (\hat{y}_i - y_i)\, x_{ij} = 0 \tag{3}$$

2. Randomly assigning values to θ_j. For a convex function it does not matter what the initial weights are as it will always converge. $\hat{y}$ and x_{ij} are observed and y_i is computed. This means the slope of the loss function can be readily computed.

Gradient Descent

Now for a given θ_j, and keeping all other θ_i where $i \neq j$ constant, we change the value of θ_j slightly and compute a new value of θ_j

$$\triangle\theta_j = \alpha\theta_j$$
$$\theta_j^{'} = \theta_j + \triangle\theta_j$$

where α is small constant and is called the learning rate. Since the slope of $\frac{\partial L}{\partial \theta_j}$ is already calculated from the previous equations, we can compute the new value of the loss as follows:

$$L^{'} = L - \frac{\partial L}{\partial \theta_j}(\alpha\theta_j)$$

If $L^{'} < L$, we continue incrementing θ_j as long as the loss decreases. When the direction changes and the loss increases, we have found the θ_j for which the loss is minimum. If $L^{'} > L$, we travel in the reverse direction and follow the same process. Like this, we compute the value of each and every θ_j.

Stochastic versus Batch Gradient Descent

1. Stochastic means a process that is linked with a random probability. In SGD, a single sample data is used to compute the θ_j. The sample is randomly shuffled and selected for performing the iteration.
2. In batch gradient descent, the gradient is calculated from the whole training set. If we have a huge dataset with millions of data points, running the batch gradient descent is expensive.
3. Compromise by computing the gradient for a mini batch at each step which performs better because of use of vectorization libraries and has smoother convergence as the gradient computed at each step uses more training examples. It is the algorithm of choice for neural nets.
4. In Batch Gradient Descent, the Batch Size refers to the total number of training examples present in a single batch and Epoch means that each sample in the training dataset has had an opportunity to update all the model parameters θ_j.
5. Our objective is to find a set of θ_j such that the loss is minimum.

Logistic Regression

In many cases the value of y_i need to be bounded between 0 and 1. In such cases, we use the logistic regression model as given below:

$$y_i = \frac{1}{1 + e^{-z}} \tag{4}$$

$$z_i = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \cdots + \theta_n x_n \tag{5}$$

- for z is large +ve number, $\frac{1}{e^z} = 0$; $y_i = 1$
- for z $= 0$, $y_i = 0.5$
- for z is large -ve number, $y_i = 0$

Hence, the value of y_i is bounded between 0 and 1 for z between -∞ and ∞.

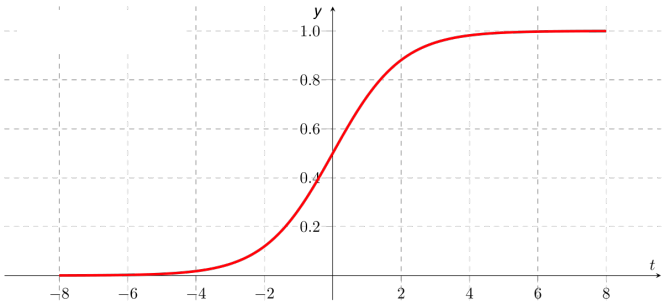
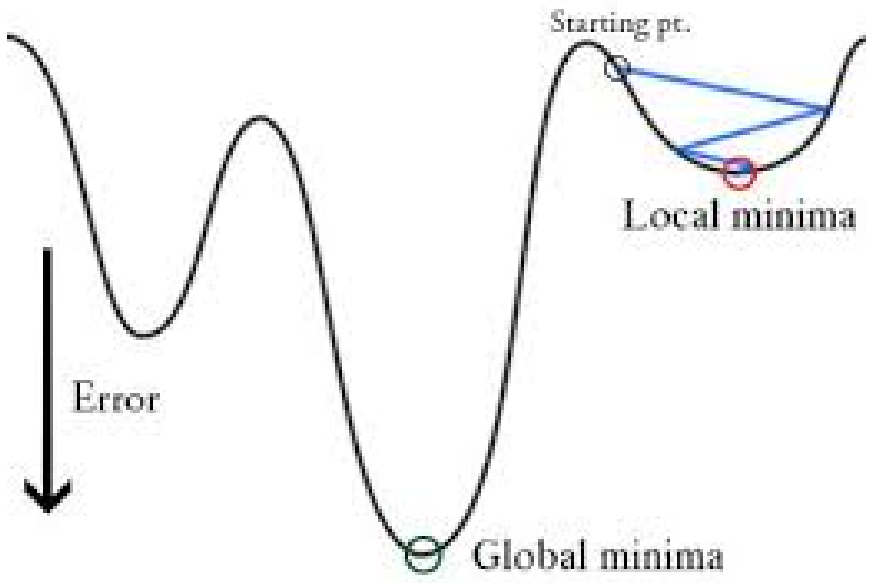


Figure: Sigmoid Curve

Logistic Regression

For logistic regression, Least Squared Error will result in a *non-convex* graph with local minimums and hence is not feasible.



Logistic Regression

Logistic regression is modeled as follows:

$$P(y|x,\theta) = \begin{cases} h_\theta(x) & \text{if y} = 1 \\ (1 - h_\theta(x)) & \text{if y} = 0 \end{cases} \tag{6}$$

The above function can be written compactly as follows:

$$P(y|x,\theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{7}$$

Assuming that the r training examples were generated independently, we can then write down the likelihood L of the parameters as:

$$L(\theta) = p(\overrightarrow{y}|x,\theta) = \prod_{i=1}^{r} P(y^{(i)}|x^{(i)},\theta)$$

$$\boxed{L(\theta) = \prod_{i=1}^{r} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}}$$

Log Likelihood

The log likelihood is given by:

$$l(\theta) = log\ L(\theta) = \prod_{i=1}^{r} y^{(i)} log(h(x^{(i)}) + (1 - y^{(i)}) log(1 - h(x^{(i)})) \tag{8}$$

Intuitively, we want to assign more punishment when the prediction is 1 while the actual is 0 and similarly when the prediction is 0 while the actual is 1. The loss function in logistic regression is doing exactly this and hence is called Logistic Loss.
The sigmoid function has an interesting property. It's derivative is given by:

$$\begin{aligned} g^{'}(x) &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2} \quad = \frac{1}{1+e^{-x}} \times \left(1 - \frac{1}{1+e^{-x}}\right) \\ &= \boxed{g(x)(1-g(x))} \end{aligned} \tag{9}$$

Log Likelihood

Now, consider just one training example (x,y), and take derivatives to derive the stochastic gradient ascent rule:

$$\begin{aligned}\frac{\partial(l(\theta))}{\partial\theta_j} &= y\frac{1}{g(\theta^T x))} - (1-y)\frac{1}{1-g(\theta^T x)}\frac{\partial}{\partial\theta_j}g(\theta^T x) \\ &= y\frac{1}{g(\theta^T x))} - (1-y)\frac{1}{1-g(\theta^T x)}g(\theta^T x)(1-g(\theta^T x))x_j \\ &= y(1-g(\theta_T x) - (1-y)g(\theta^T x)x_j \\ &= y - h_\theta(x)x_j\end{aligned}$$

With the above equation, we are now in a position to compute the gradient and follow the same steps to determine θ_j corresponding to minimum loss as in the case of linear regression.

Predictions & Errors

1. Predictions from the model will have differences or errors.

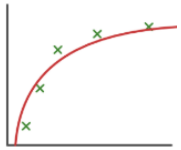


Figure: Correct Fit

2. Overfitting will occur when an excessive number of features are used than required.

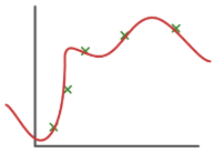
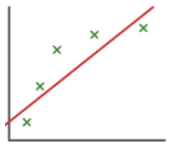


Figure: Overfit

3. Underfitting occurs when an insufficient number of features are used than required.



Regularization

1. Bias is the difference between average model prediction and the true target value and variance is the variation in predictions across different training data samples. Simple models with small number of features have high bias and low variance whereas complex models with large number of features have low bias and high variance.
2. Regularization is a technique used to avoid problem of overfitting. It prevents overfitting in linear models by a penalty term that penalizes large weight values.
3. A regression model that uses L1 regularization (or L1 Norm) technique is called Lasso Regression and model that uses L2 regularization (or L2 Norm) is called Ridge Regression.
4. The key difference between these two is the penalty term.

Regularization - Lasso & Ridge

1. Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds the absolute value of magnitude of coefficient as a penalty term to the loss function. Here the highlighted part represents L1 regularization element where λ is a constant.

$$L + \boxed{\lambda \sum_{j=1}^{n} |\theta_j|} \tag{10}$$

2. Ridge regression adds the squared magnitude of coefficient as a penalty term to the loss function. Here the highlighted part represents L2 regularization element.

$$L + \boxed{\lambda \sum_{j=1}^{n} \theta_j^2} \tag{11}$$

3. Lasso shrinks the less important feature's coefficient to zero thus removing some features altogether. Hence, this works well for feature selection where we have a large number of features. However, the L1-norm does not have an analytical solution. L2-norm solutions can be computed efficiently.

SoftMax

1. In the training data the values corresponding to the features x_j can have numerical values of different magnitudes. For example, one feature may have values ranging between $0 \ldots 10$, while another feature may have values ranging between $10,000 \cdots 100,000$. Some of these values can also be negative and the components will obviously not add up to 1.
2. The softmax function is a function that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities in the interval 0...1 with the components adding up to 1.

$$\sigma(z)_i = \frac{e_i^z}{\sum_{j=1}^{K} e^{z_j}} \tag{12}$$

The feature values are normalized with the softmax function prior to inputing in a regression model.

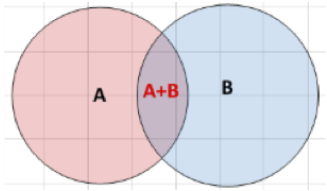
One Hot Encoding

1. In the linear regression and the logistic regression models that we studied so far can only deal with numerical data. It cannot handle features that have text as values. In one-hot encoding, the string encoded variable is replaced with new variables of boolean type. For example, a particular feature, say color, can have "red", "green" and "blue" as possible values. This feature color, will be replaced by 3 features, red, blue and green which hold the values 1 or 0. We can then encode color, in terms of red, blue and green as follows:

red	green	blue
1	0	0
0	1	0
0	0	1

Note that if the boolean value of the feature is 1, it must be 0 for all the other features

Conditional Probability - Naive Bayes



$$\mathbf{P(A|B)} = \frac{\boldsymbol{P(A \cap B)}}{\boldsymbol{P(B)}}$$

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)} \qquad P(B|A) = P(A|B) \times \frac{P(B)}{P(A)} \tag{13}$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Naive Bayes

$$
\begin{aligned}
P(Play = Yes) &= 9/14 \\
P(Play = No) &= 5/14 \\
P(Outlook = Sunny \mid Play = Yes) &= 2/9 \\
P(Temperature = Cool \mid Play = Yes) &= 3/9 \\
P(Humidity = High \mid Play = Yes) &= 3/9 \\
P(Wind = Strong \mid Play = Yes) &= 3/9 \\
P(Outlook = Sunny \mid Play = No) &= 3/5 \\
P(Temperature = Cool \mid Play = No) &= 1/5 \\
P(Humidity = High \mid Play = No) &= 4/5 \\
P(Wind = Strong \mid Play = No) &= 3/5
\end{aligned}
$$

Given, X = (Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong), do we play?

$$
\begin{aligned}
P(Yes \mid X) = P(Sunny \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \\
\times P(Strong \mid Yes) \times P(Play = Yes) = 0.0053
\end{aligned}
$$

$$
\begin{aligned}
P(No \mid x) = P(Sunny \mid No) \times P(Cool \mid No) \times P(High \mid No) \\
\times P(Strong \mid No) \times P(Play = No) = 0.0206
\end{aligned}
$$

Since $P(Yes \mid X) < P(No \mid X)$, we label X to be No.

Latent Dirichlet Allocation (LDA)

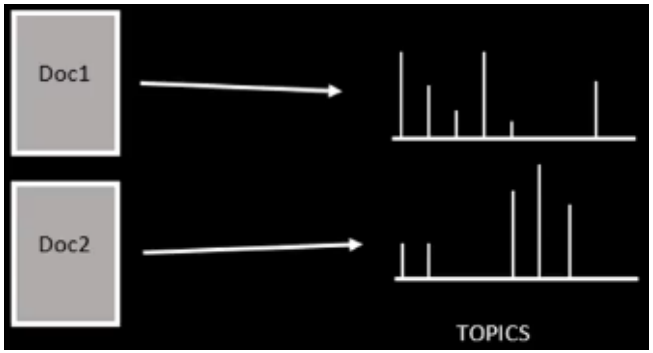


Figure: Topic Distribution

1. LDA is an unsupervised machine learning algorithm. LDA uses a generative process that treats a document as bag of words and learns the topics in the document.
2. The words are observable and the topics in the document are hidden and hence the term latent.
3. We are directly observing the words and not the topics, so the topics themselves are latent variables (along with the distributions themselves). Documents are probability distribution over latent topics and topics are probability distribution over words.

Latent Dirichlet Allocation (LDA)

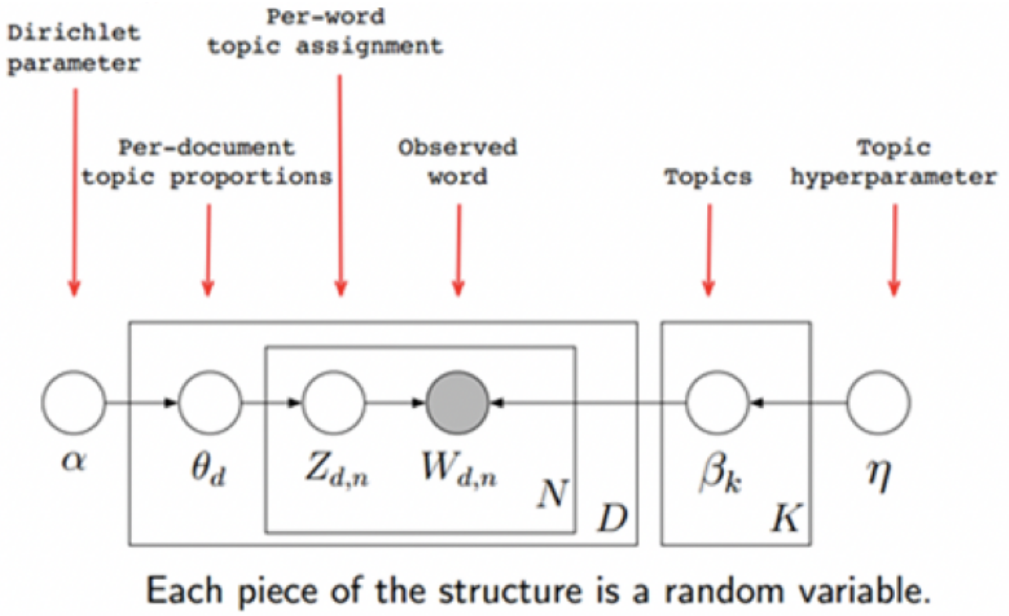


Figure: Plate Notation

The plate notation is a concise way of visually representing the dependencies among the model parameters. α is the parameter in the Dirichlet prior on the per document distribution. A high α implies that a document is likely to contain most of the topics. β is the parameter in the Dirichlet prior on the per topic word distribution. A high β implies that a topic is likely to contain most of the words.

LDA

1. w represents a word
2. W represents a document (a vector of words); $w = w_1, w_2, \cdots w_N$
3. α is the parameter of the Dirichlet distribution;

$$\alpha = (\alpha_1, \alpha_2, \cdots \alpha_n)$$

4. θ is per document topic proportions
5. Z is a vector of topics, where if the i th element of z is 1 then w draws from the i th topic D is the total set of documents
6. LDA assumes that new documents are generated by choosing a topic mixture for the document, (eg., 20% topic-1, 30%, topic-2 and 50% topic-3)) and generate words by first pick a topic based on the documnent's multinomial distribution and then pick a word based on the topics topic's multinomial distribution.

LDA Steps

1. The first step is to decide that there are k topics.
2. Next, from a given document, assign each word to of the k topics randomly.
3. Assume that all topic assignments except for the current word are correct. Calculate two ratios.
   - Proportion of words in the document d that are currently assigned to topic t.
   - Proportion of assignments to topic t over all documents that come from this word.
   - Assign the current word to the topic for which the probability p is maximum.

We continue repeating this process of re-assigning words until we reach a steady state. Mathematically, the LDA is expressed as follows:

$$p(\theta, z|w, \alpha, \beta) \propto p(\theta, z, w|\alpha, \beta) = p(w|z, \beta)p(z|\theta)p(\theta|\alpha) \tag{14}$$

Now, given a new document, how do we determine what topics does it consist of? We can score potential topics using the cross-entropy, i.e., KL distance.

Information & Uncertainty

1. In Claude Shannon's (MIT) information theory, one bit of information reduces the uncertainty by 2. Similarly, if 3 bits of information are sent, then the reduction in uncertainly by 2^3, i.e., 8. This is intuitive. With 3 bits, there could be 8 possible values and so if a particular set of bits are transmitted, 8 possibilities are eliminated with 1 certainty.
2. Information Content When the information is probabilistic, the self-information I_x, or Information Content of *measuring a random variable X as outcome x is defined as:*

$$I_x = log\left(\frac{1}{p(x)}\right) = -log(p(x)) \tag{15}$$

where $p(x)$ is probability mass function.

3. Shannon Entropy of the random variable X is defined as:

$$H(X) = \sum_x -p(x)log(p(x)) = E(I_x) \tag{16}$$

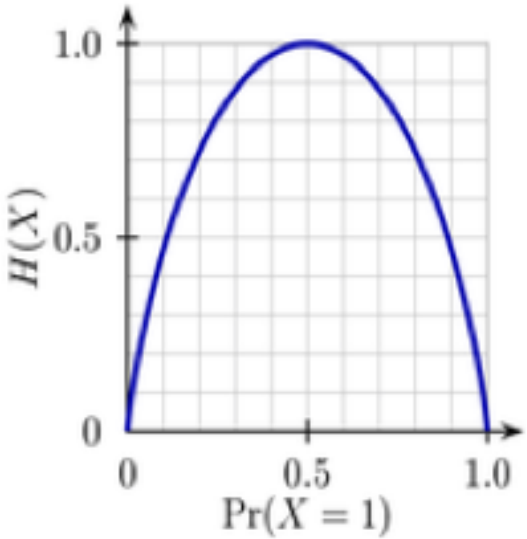
It is the *expected information content* of the measurement of X.

Example

1. Suppose S has 25 examples, 15 positive and 10 negatives [15+, 10-]. Then the entropy of S relative to this classification is computed as:

$$E(S) = -(15/25)log_2(15/25) - (10/25)log_2(10/25) = 0.67$$

In a 2 class system, The entropy is 0 if the outcome is *certain*. The entropy is maximum if we have no knowledge of the system, i.e., whemn any outcome is equally possible.



Cross Entropy - Kullback–Leibler (KL) divegence

1. Cross-Entropy is defined as:

$$H(p,q) = -\sum_i p(i) log_2 q(i) \tag{17}$$

where p is the true distribution and q is the predicted distribution. If the predictions are perfect, then the cross-entropy is same as the entropy. If the prediction differs, then there is a divergence which is known as *Kullback – Leibler (KL) divergence*. Hence,

$$Cross\ Entropy = Entropy + KL\ Divergence$$

$$KL\ Divergence = H(p,q) - H(p)$$

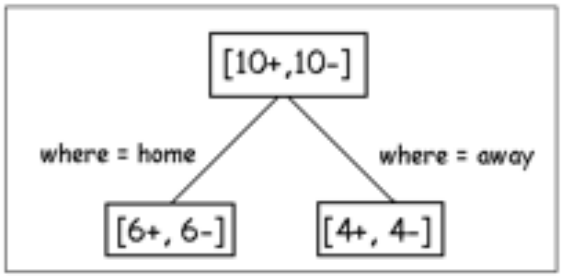
Hence, if the predicted distribution is closer to true distribution when KL divergence is low.

Decision Tree

1. Decision Tree is a supervised machine learning algorithm. In decision trees, the goal is to tidy the data. You try to separate your data and group the samples together in the classes they belong to. You know their label since you construct the trees from the training set. You maximize the purity of the groups as much as possible each time you create a new node of the tree (meaning you cut your set in two). Of course, at the end of the tree, you want to have a clear answer.
2. At each step, each branching, you want to decrease the entropy, so this quantity is computed before the cut and after the cut. If it decreases, the split is validated and we can proceed to the next step, otherwise, we must try to split with another feature or stop this branch.
3. Before and after the decision, the sets are different and have different sizes. Still, entropy can be compared between these sets, using a weighted sum.

Example

Consider the following table for win/loss of soccer games at home and away.



The entropy is:

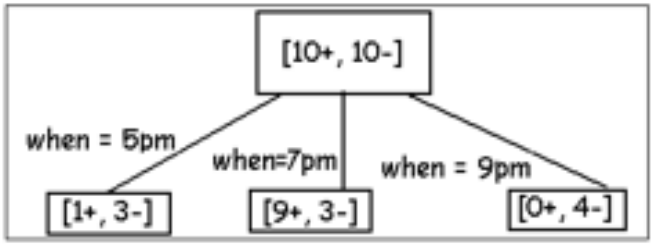
$$H\left(\frac{6}{12},\frac{6}{12}\right) = -\frac{6}{12}log_2\left(\frac{6}{12}\right) - \frac{6}{12}log_2\left(\frac{6}{12}\right) = 0.69$$

$$H\left(\frac{4}{8},\frac{4}{8}\right) = -\frac{4}{8}log_2\left(\frac{4}{8}\right) - \frac{4}{8}log_2\left(\frac{4}{8}\right) = 0.69$$

$$H = -\frac{12}{20} \times H\left(\frac{6}{12},\frac{6}{12}\right) - \frac{8}{20}H\left(\frac{4}{8},\frac{4}{8}\right) = 0.69$$

Example

Partitioning by when, we have:



$$H(5pm) = H\left(\frac{1}{4}, \frac{3}{4}\right) = -\frac{1}{4}log_2\left(\frac{1}{4}\right) - \frac{3}{4}log_2\left(\frac{3}{4}\right) = 0.56$$

$$H(7pm) = H\left(\frac{9}{12}, \frac{3}{12}\right) = -\frac{9}{12}log_2\left(\frac{9}{12}\right) - \frac{3}{12}log_2\left(\frac{3}{12}\right) = 0.56$$

$$H(9pm) = H\left(\frac{0}{4}, \frac{4}{4}\right) = -\frac{0}{4}log_2\left(\frac{0}{4}\right) - \frac{4}{4}log_2\left(\frac{4}{4}\right) = 0.00$$

Entropy after partition is:

$$H = -\frac{4}{20}H\left(\frac{1}{4}, \frac{3}{4}\right) - \frac{12}{20}H\left(\frac{12}{20}, \frac{3}{4}\right) - \frac{4}{20}H\left(\frac{4}{20}, \frac{3}{4}\right) = 0.45$$

Hence, the Information gain is 0.69 - 0.45 = 0.24.

Decision Trees

1. We can apply the same approach for other columns to find the most dominant factor on decision. This way we construct the decision tree that has the least entropy.
2. Decision trees can generate understandable rules and classify without much computation. It can handle both continuous and categorical variables. Also, it provides a clear indication of which fields are most important for prediction or classification.
3. However, it is not suitable for prediction of continuous attributes. It performs poorly with many class and small data. It is computationally expensive to train.

Random Forest

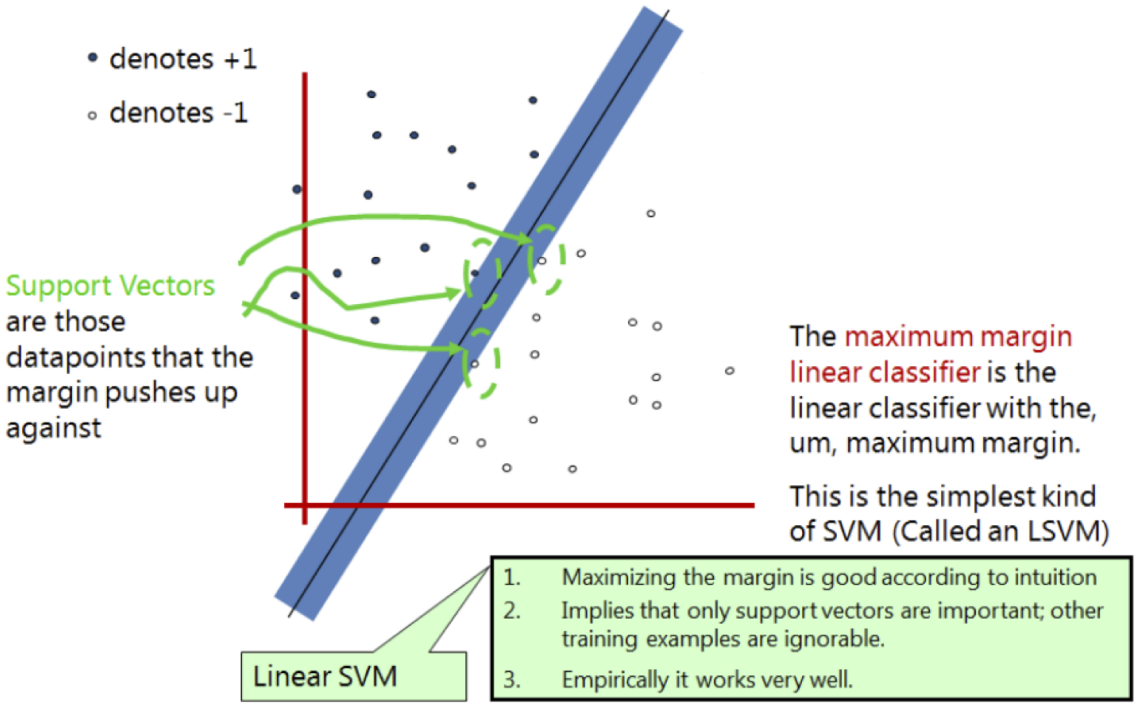
1. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. The fundamental idea behind a random forest is to combine many decision trees into a single model. Individually, predictions made by decision trees (or humans) may not be accurate, but combined together, the predictions will be closer to the mark on average.
2. Why the name 'random forest?' Well, much as people might rely on different sources to make a prediction, each decision tree in the forest considers a random subset of features when forming questions and only has access to a random set of the training data points.
3. This increases diversity in the forest leading to more robust overall predictions and the name 'random forest.' When it comes time to make a prediction, the random forest takes an average of all the individual decision tree estimates. (This is the case for a regression task, such as our problem where we are predicting a continuous value of temperature.

Clustering

1. Clustering is an unsupervised learning technique
2. Create the first K initial clusters from the dataset by choosing K rows of data randomly from the dataset.
3. Each record is assigned to the nearest cluster (the cluster which it is most similar to) using a measure of distance or similarity.
4. Re-calculate the arithmetic mean for each cluster.
5. Repeat step 2.
6. Stop when arithmetic mean for the clusters stabilize.

Support Vector Machines (SVM)

1. SVM is a supervised machine learning algorithm. SVM performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels.
2. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables.



SVM

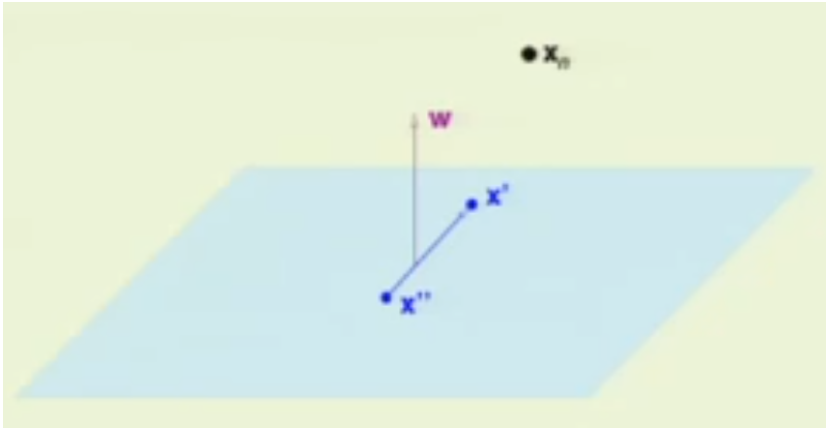


Figure: Plane

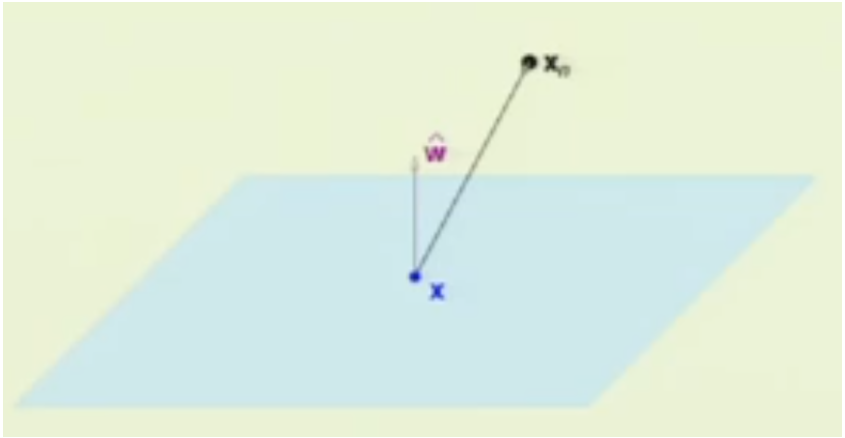


Figure: Normal to the Plane

SVM

1. The equation of a plane is given by:

$$w^T x + b = 0 \tag{18}$$

For two points x' and x'' on the plane, we have:

$$w^T x^{'} + b = 0$$
$$w^T x^{''} + b = 0$$
$$w^T (x^{'} - x^{''}) = 0$$

The normal distance d of a point x_n from x is given by:

$$d = ||\hat{w}(x_n - x)|| = \frac{1}{||w||}(w^T x_n + b - w^T x - b) = \frac{2}{||w||} \tag{19}$$

SVM

Our goal is to maximize the distance which is the same as minimizing $\frac{1}{2}w^T w$ subject to the condition:

$$wx_i + b \geq 1 \text{ for } y_i = +1 \tag{20}$$
$$wx_i + b \leq 1 \text{ for } y_i = -1 \tag{21}$$

i.e., for all i:

$$y_i(wx_i + b) \geq 1 \tag{22}$$

We will now use the Lagrange formulation to minimize:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}w^T w - \sum^{N} \alpha_n(y_n(w^T x_n + b) - 1) \tag{23}$$

with respect to w and b and maximize with respect to each $\alpha_n > 0$

SVM

$$\nabla \mathcal{L} = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \tag{24}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0 \tag{25}$$

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{26}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \tag{27}$$

$$\boxed{\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} - \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m} \tag{28}$$

maximize above eqn w.r.t α subject to $\alpha_x > 0$ *for* $n = 1, 2, \cdots N$.

SVM

$$\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} - \sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m \tag{29}$$

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n \tag{30}$$

$$\min_{\alpha} \frac{1}{2}\alpha^T] \begin{vmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \cdots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \cdots & y_2 y_N x_2^T x_N \\ \cdots & \cdots & \cdots & \cdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \cdots & y_N y_N x_N^T x_N \end{vmatrix} \alpha + (-1^T)\alpha \tag{31}$$

subject to $y^T \alpha = 0; 0 < \alpha \leq \infty$.

$$\min_{\alpha} \boxed{\frac{1}{2}\alpha^T Q \alpha - 1^T \alpha} \tag{32}$$

SVM

subject to Karush Kuhn Tucker (KKT) condition: $y^T \alpha = 0; \alpha \geq 0$. We then compute w:

$$\boxed{w = \sum_{n=1}^{N} \alpha_n y_n x_n} \tag{33}$$

and solve for b:

$$\boxed{y_n(wx_n + b) = 1} \tag{34}$$

Non Linearity

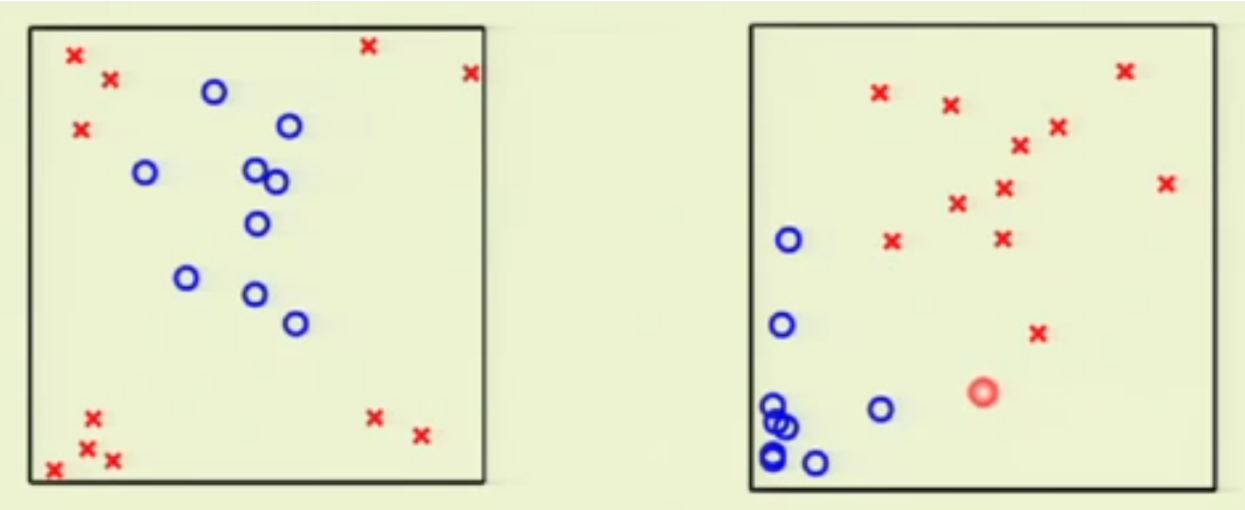


Figure: Transform to linear space

For non-linear systems, we transform to z space:

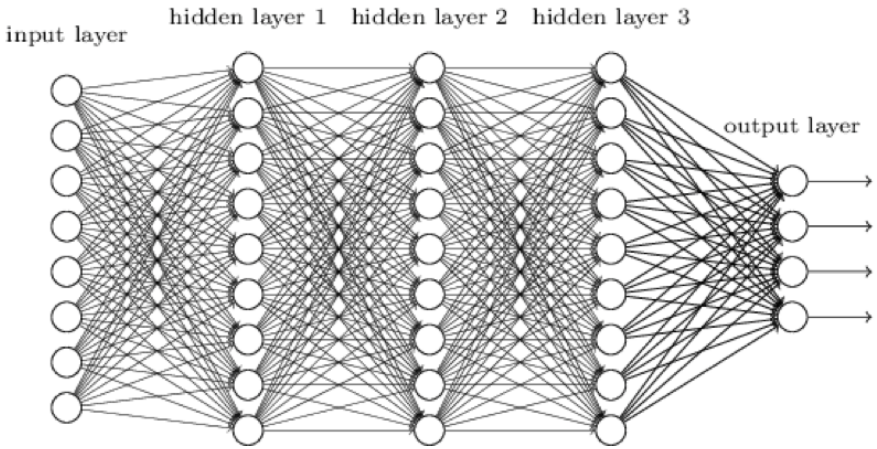
$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} - \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m z_n^T z_m \tag{35}$$

and we will have support vectors in the transformed space.

Deep Neural Net

1. Deep Learning Success Stories - Image Recognition, Speech Comprehension, Chatbot
2. DNNs are suitable where the raw underlying features are not individually interpretable. This success is attributed to their ability to learn hierarchical representations, unlike traditional methods that rely upon hand-engineered features.
3. DNNs are greedily built by stacking restricted Boltzmann machines, and Convolutional Neural Networks, which exploit the local dependency of visual information, and have demonstrated record-setting results on many important applications.
4. Neural networks are powerful learning models that achieve state-of-the-art results in a wide range of supervised and unsupervised machine learning tasks for classification & regression.
5. Simple linear models tend to under-fit, and often under utilize computing resources with 8-GPU Hydra.

Deep Neural Net



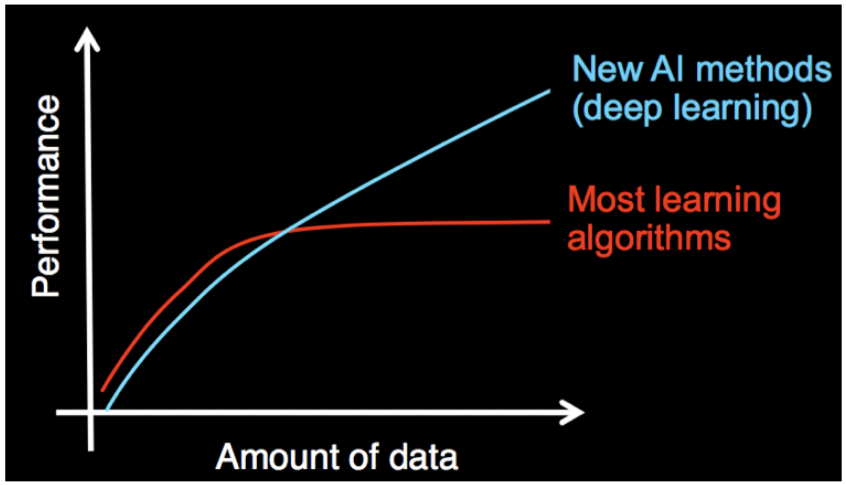
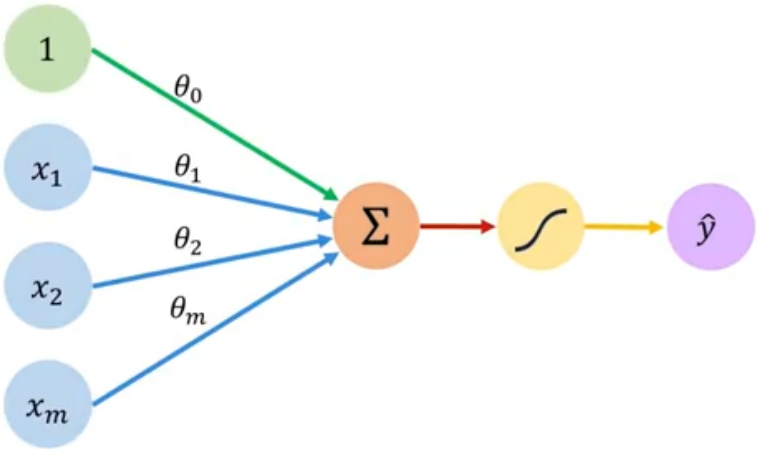
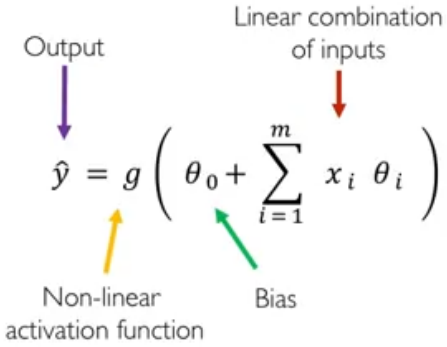


Figure: Deep Learning

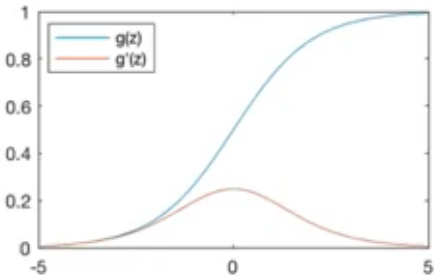
Deep Neural Net



Inputs Weights Sum Non-Linearity Output



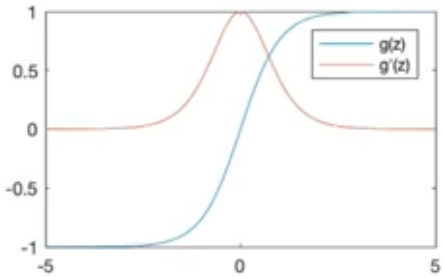
Sigmoid Function



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

$$g'(z) = g(z)(1 - g(z))$$

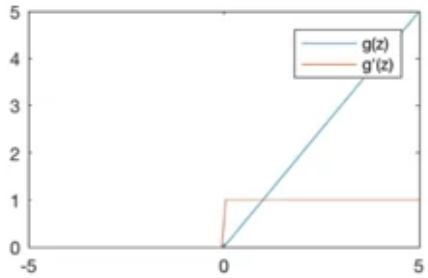
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

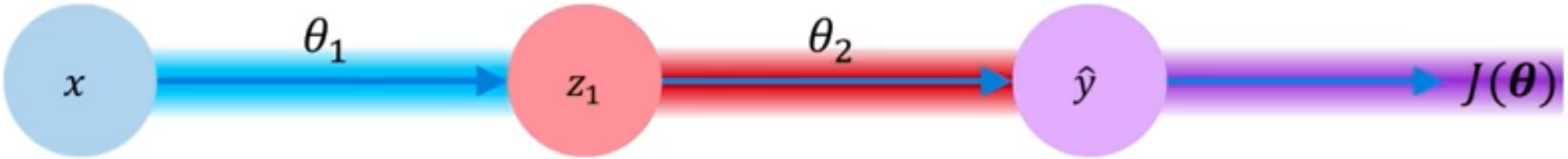
Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

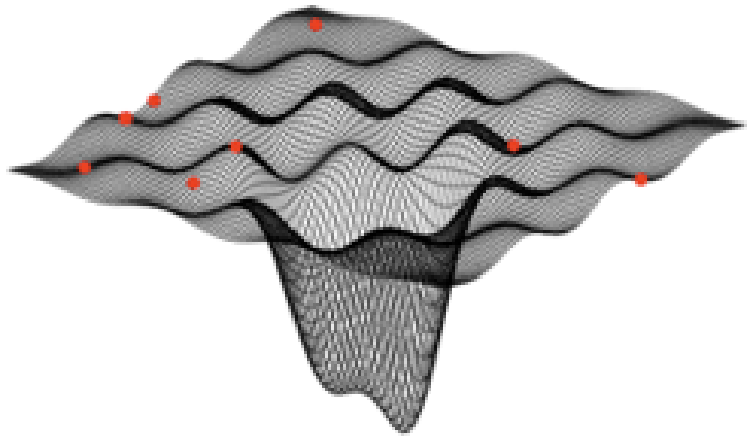
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Back Propagation



$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} = \frac{\partial J(\boldsymbol{\theta})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial \theta_1}$$

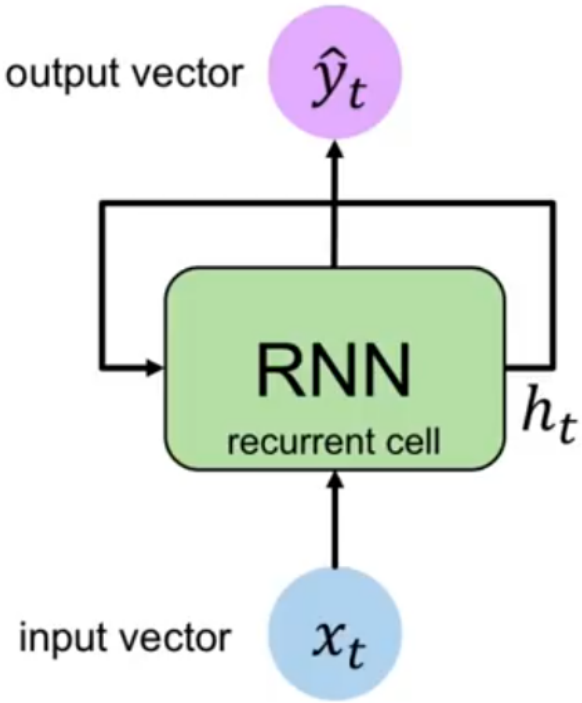
Figure: Loss Minimization



Recurrent Neural Network (RNN)

1. Dialogue systems, self-driving cars, robotic surgery, speech, etc. among others require an explicit model of sequentiality or time – a combination of classifiers or regressors cannot provide these functionalities.
2. SVM, logistic regression, feedforward networks have proved very useful without explicitly modeling time. But the assumption of independence precludes modeling long range dependencies.
3. Frames from video, snippets of audio, and words pulled from sentences – the independence assumption fails.
4. RNNs are connectionist models with the ability to selectively pass information across sequence steps while processing sequential data one element at a time. They can model input and/or output consisting of sequences of elements that are not independent. In other words can handle variable length sequences, track long term dependencies, maintain information about order and share parameters across the sequence.

RNN



Apply a **recurrence relation** at every time step to process a sequence:

$$\underbrace{h_t}_{\text{cell state}} = \underbrace{f_W}_{\substack{\text{function} \\ \text{parameterized} \\ \text{by W}}} (\underbrace{h_{t-1}}_{\text{old state}}, x_t)$$

Figure: Same function and set of parameters are used in every step

Source - MIT 6.S191- Recurrent Neural Networks

RNN

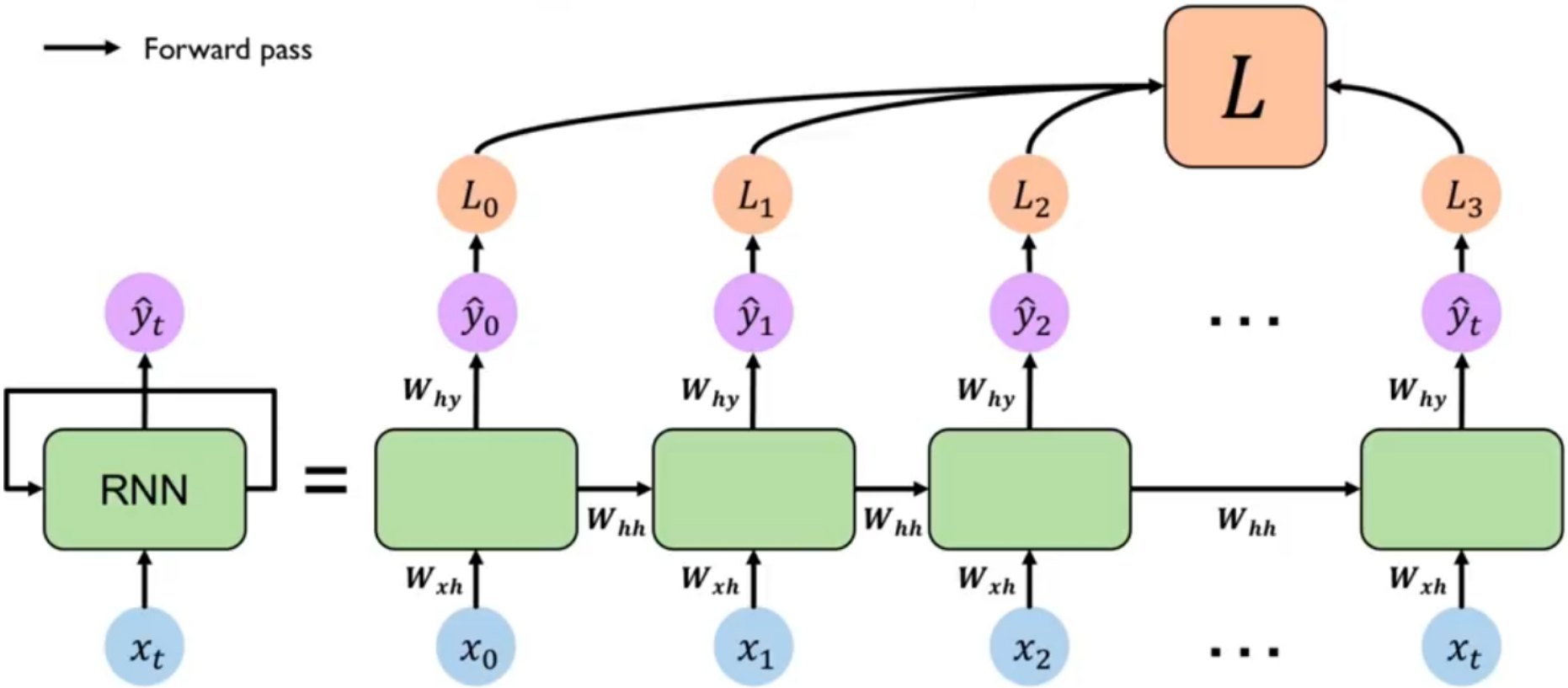


Figure: Computational Graph

RNN

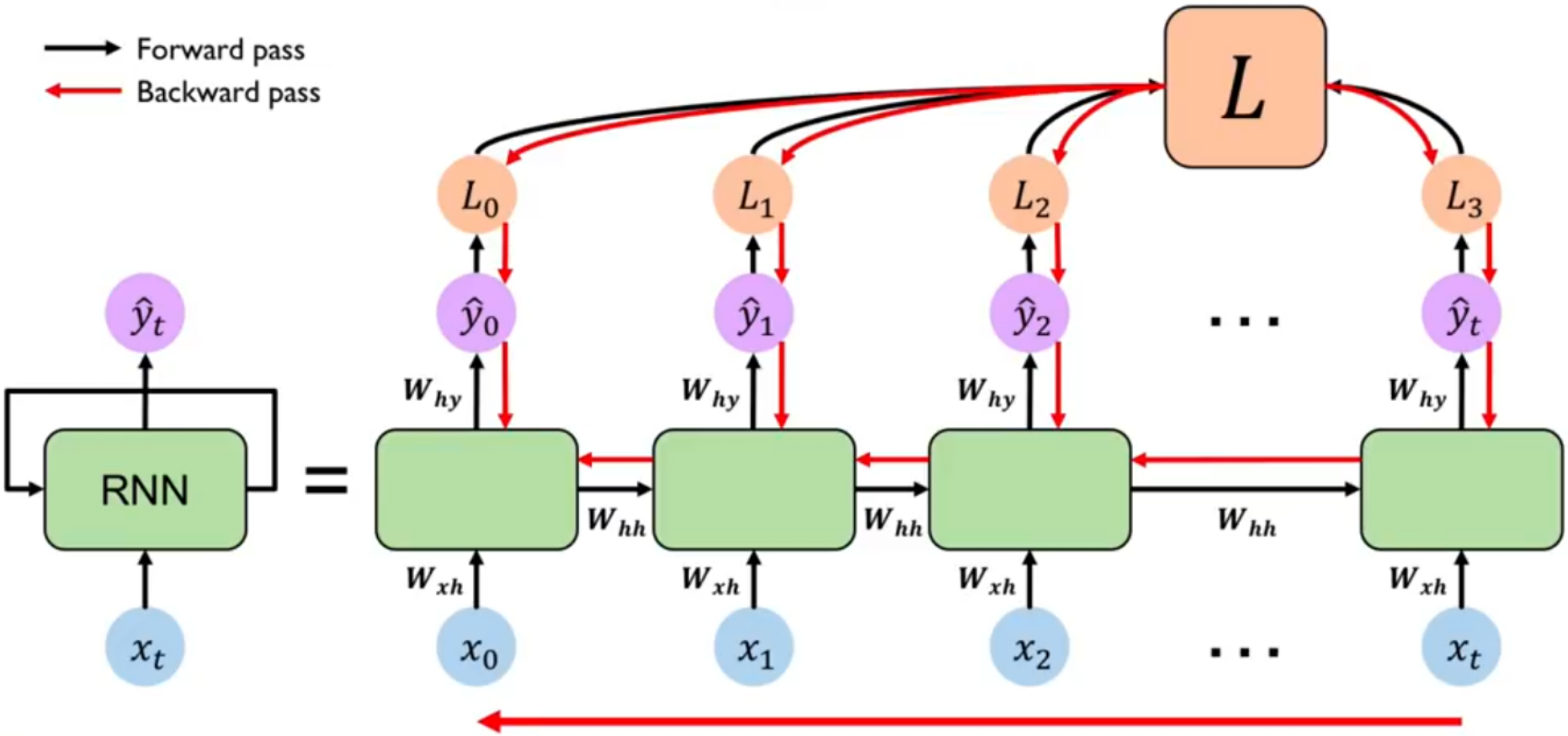


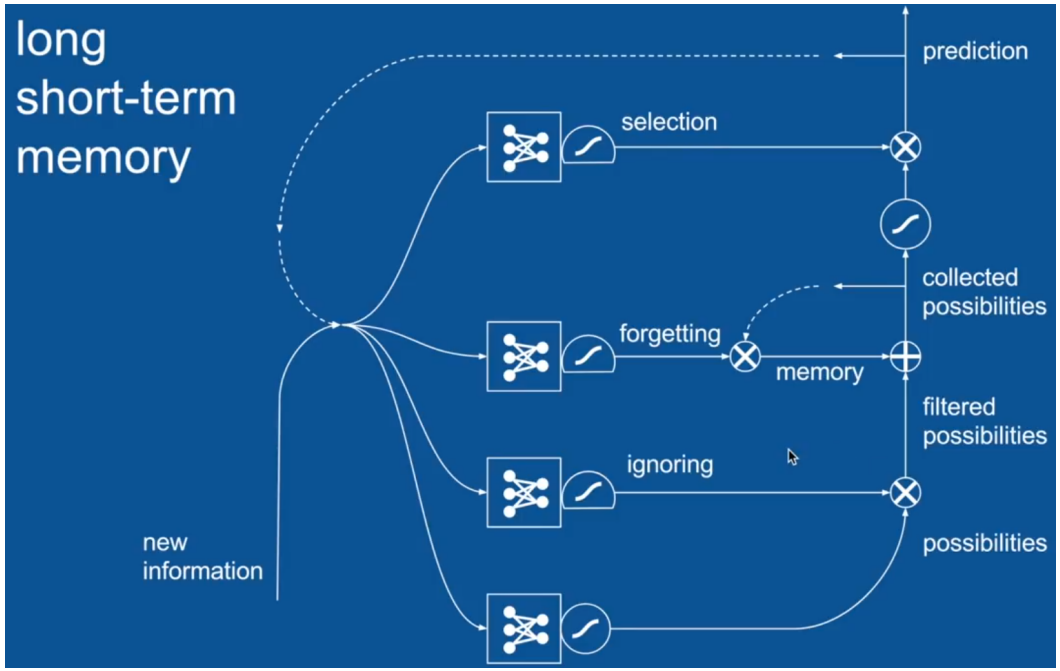
Figure: Backpropagation over time

The Problem of Vanishing / Exploding Gradients

1. Learning with recurrent networks is challenging due to the difficulty of learning long-range dependencies, [Bengio et al. 1994], [Hochreiter et al, 2001].
2. Problems of vanishing and exploding gradients occur when backpropagating errors across many time steps.
3. Which of the two phenomena occurs depends on whether the weight of the recurrent edge $|wjj| > 1$ or $|wjj| < 1$ and on the activation function in the hidden node. For a sigmoid activation function, the vanishing gradient problem is more pressing, but with a rectified linear unit $max(0, x)$, it is easier to imagine the exploding gradient.

Long Short Term Memory (LSTM) - Hochreiter & Schmidhuber

1. Maintain a separate cell state
2. Use gates
   1. Forget
   2. Selectively update
   3. Ouput
3. Backpropagation does not require matrix multiplication



Reinforcement Learning

1. $\hat{P}$ is sampled and is NOT predicted.
2. P_i is predicted and is a function of the weights and is differentiable with respect to the weights.
3. Cross Entropy Loss has to be minimized, this will lead to determining the weights.
4. Policy Gradients
   1. Discounted Rewards for the i^{th} move. R_i reduces geometrically over previous steps.
   $$loss_i = -R_i log(P_i^{chosen}) \tag{36}$$
   2. Normalized Rewards across a batch of moves
   $$R_i = \frac{R_i - \bar{R}}{stdev(R)} \tag{37}$$

Generative Adversarial Network (GAN)

1. GAN is a neural network architecture that allows neural networks to generate data and learn a probability distribution of a data set by pitting two neural networks against each other.
2. The generator acts like a painting forger. It tries to create images that look like real. The discriminator tries to detect the fakes.
3. The generator keeps getting better at making fakes, while the discriminator gets better at detecting fakes. The 2 models compete, and after many iterations, the generator creates images indistinguishable from the real data set.