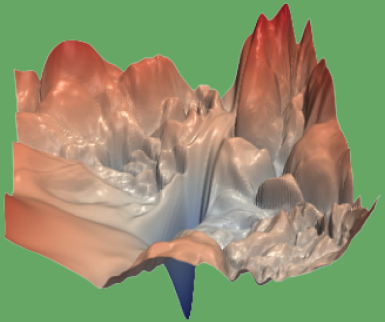
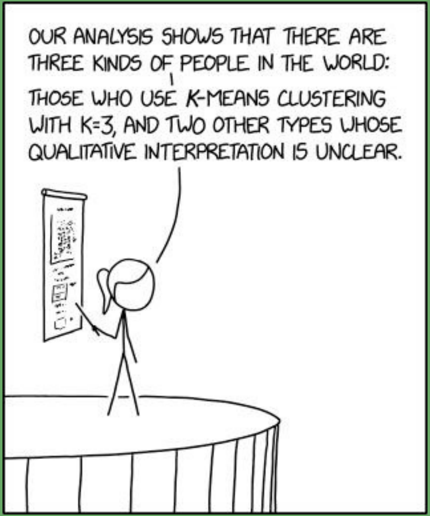
Deep Learning

AAKASH GHOSH
19MS129



Presentation made for the completion of Course MA5115

Traditional ML



Traditional Machine learning

The older/traditional way of applying machine learning consisted of the following steps:

- **Feature Extraction**: Features which can be used to discriminate between classes is captured. This step usually requires in-field knowledge about the problem.

Traditional Machine learning

The older/traditional way of applying machine learning consisted of the following steps:

- **Feature Extraction**: Features which can be used to discriminate between classes is captured. This step usually requires in-field knowledge about the problem.
- **Model Selection**: A model is selected which trains on the extracted features. Ensable methods can be used to boost performance

Traditional Machine learning

The older/traditional way of applying machine learning consisted of the following steps:

- **Feature Extraction**: Features which can be used to discriminate between classes is captured. This step usually requires in-field knowledge about the problem.
- **Model Selection**: A model is selected which trains on the extracted features. Ensable methods can be used to boost performance
- **Cross-validation/Tested**: The model is tested/cross-validated on with held data

Problems with the traditional approach

- **Feature Extraction**: This step requires in-field knowledge. It is very difficult to study a whole new branch of knowledge for a single problem.

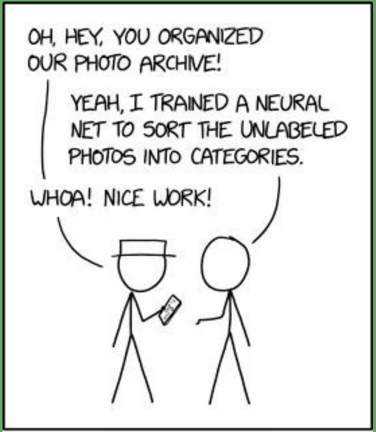
Problems with the traditional approach

- **Feature Extraction**: This step requires in-field knowledge. It is very difficult to study a whole new branch of knowledge for a single problem.
- **Amount of Data**: In the current era, the amount of data sometimes is simply so large that meaningful features are hard to extract

Problems with the traditional approach

- **Feature Extraction**: This step requires in-field knowledge. It is very difficult to study a whole new branch of knowledge for a single problem.
- **Amount of Data**: In the current era, the amount of data sometimes is simply so large that meaningful features are hard to extract
- **Unorganized Data**: Feature extraction is hard in unorganized data (such as a literary works).

The idea behind deep learning



ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

Ideas behind a neural network

- **Pattern recognition**: Almost all the problems with traditional Ml lies in proper feature selection. We now take a different approach: We train the machine to learn what the necessary patterns/features are.

Ideas behind a neural network

- **Pattern recognition**: Almost all the problems with traditional Ml lies in proper feature selection. We now take a different approach: We train the machine to learn what the necessary patterns/features are.
- **Inspiration**: For this task we take inspiration from the best pattern recognizing machine that we know:

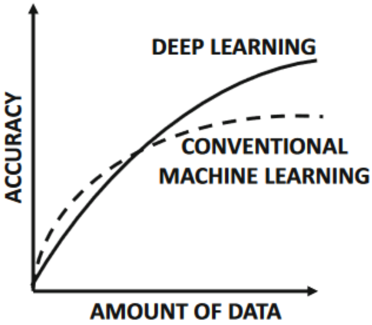
Ideas behind a neural network

- **Pattern recognition**: Almost all the problems with traditional Ml lies in proper feature selection. We now take a different approach: We train the machine to learn what the necessary patterns/features are.
- **Inspiration**: For this task we take inspiration from the best pattern recognizing machine that we know: **our brains**.

Ideas behind a neural network

- **Pattern recognition**: Almost all the problems with traditional Ml lies in proper feature selection. We now take a different approach: We train the machine to learn what the necessary patterns/features are.
- **Inspiration**: For this task we take inspiration from the best pattern recognizing machine that we know: **our brains**.
- **Overview**: We make a perceptron, which is essentially the digital analog of a neuron. We connect multiple neurons together(similar to our brain) and look at what pattern and amount of activation leads to best result.

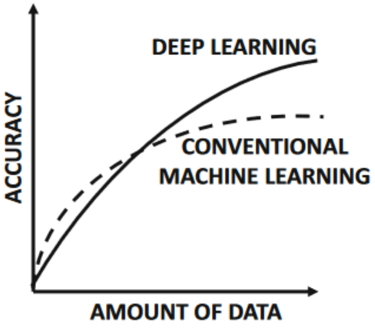
A practical comparison with traditional ML



Comparision based on amount of data features.
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- Traditional ML models show better prediction when the amount of features involved is small. Features can be individually engineered and interpreted.

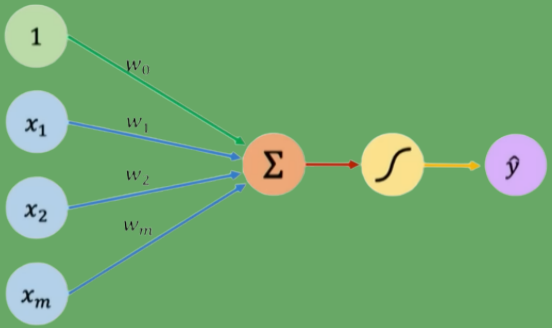
A practical comparison with traditional ML



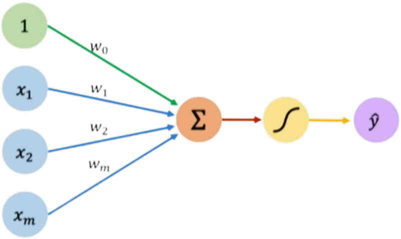
Comparision based on amount of data features.
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- Traditional ML models show better prediction when the amount of features involved is small. Features can be individually engineered and interpreted.
- Deep learning models are better when data is unstructured or there are a lot of features which need to be considered. With proper construction and training almost any decision boundaries can be learned.

The Perceptron



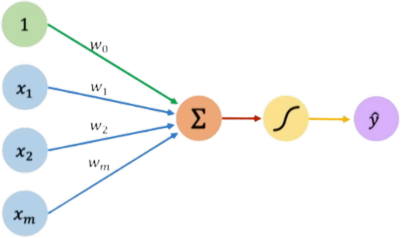
Non-Linearity is the key



A perceptron
Src: MIT Introduction to Deep Learning,6.S191,Lec-1

- The idea behind a perceptron is that a perceptron will give a high value as output when it recognises a certain feature.

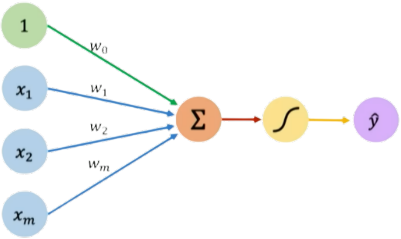
Non-Linearity is the key



A perceptron
Src: MIT Introduction to Deep Learning,6.S191,Lec-1

- The idea behind a perceptron is that a perceptron will give a high value as output when it recognises a certain feature.
- Mathematically for an input vector X it outputs $\hat{Y} = \sigma(W^T X + b_0)$

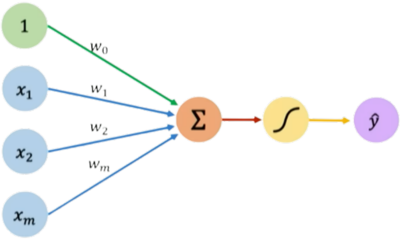
Non-Linearity is the key



A perceptron
Src: MIT Introduction to Deep Learning,6.S191,Lec-1

- The idea behind a perceptron is that a perceptron will give a high value as output when it recognises a certain feature.
- Mathematically for an input vector X it outputs $\hat{Y} = \sigma(W^T X + b_0)$
- Multiple perceptrons are wired together(again, much like our brain) to identify richer features.

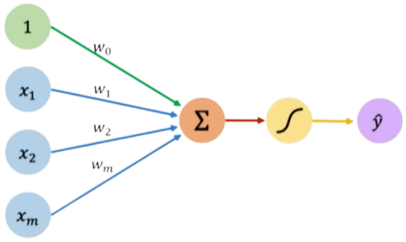
Non-Linearity is the key



A perceptron
Src: MIT Introduction to Deep Learning,6.S191,Lec-1

- The idea behind a perceptron is that a perceptron will give a high value as output when it recognises a certain feature.
- Mathematically for an input vector X it outputs $\hat{Y} = \sigma(W^T X + b_0)$
- Multiple perceptrons are wired together(again, much like our brain) to identify richer features.
- At the very core, Machine Learning works by constructing proper decision boundaries.

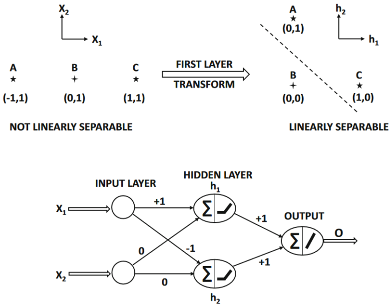
Non-Linearity is the key



A perceptron
Src: MIT Introduction to Deep Learning,6.S191,Lec-1

- The idea behind a perceptron is that a perceptron will give a high value as output when it recognises a certain feature.
- Mathematically for an input vector X it outputs $\hat{Y} = \sigma(W^T X + b_0)$
- Multiple perceptrons are wired together(again, much like our brain) to identify richer features.
- At the very core, Machine Learning works by constructing proper decision boundaries.
- A very simple calculation shows if σ is linear then the decision boundary is also linear. It therefore makes sense to use non-linear σ.

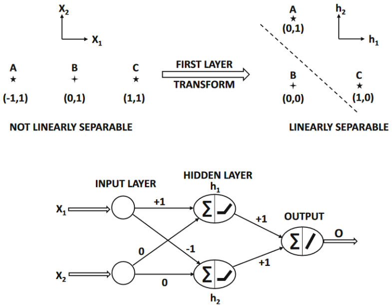
Non-Linearity is the key



Non-linearity in action
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- As shown in the figure on left, classes which can't be separated by linear boundaries can be separated by non-linear functions.(Think SVM but the kernals are learned.)

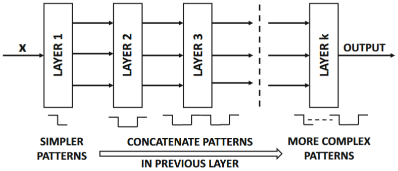
Non-Linearity is the key



Non-linearity in action
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- As shown in the figure on left, classes which can't be separated by linear boundaries can be separated by non-linear functions.(Think SVM but the kernals are learned.)
- It can be theoretically shown that almost all boundary function can be separated by a 2 layered neural network.

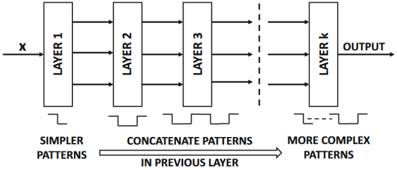
Increasing depth



Why is depth needed
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- While a single hidden layer is enough, making a deep neural network allows us to have more complex decision boundaries with relatively less number of nodes

Increasing depth

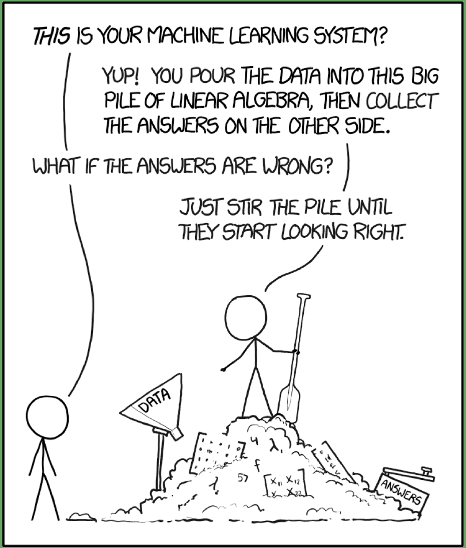


Why is depth needed
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

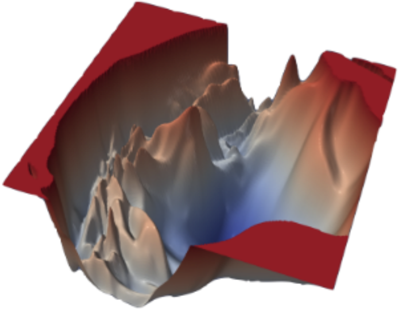
- While a single hidden layer is enough, making a deep neural network allows us to have more complex decision boundaries with relatively less number of nodes
- It should be noted how non-linearity discussed above plays an important role: Irrespective of number of layers, composition of linear functions is linear. On the other hand compositions of non-linear function can lead to richer families of functions. (*Ref: https://www.desmos.com/calculator/m645ggyl2i*)

Training a neural network

Loss functions,Backpropagation, Reversemode Auto-Diff, Practical problems and hardware considerations



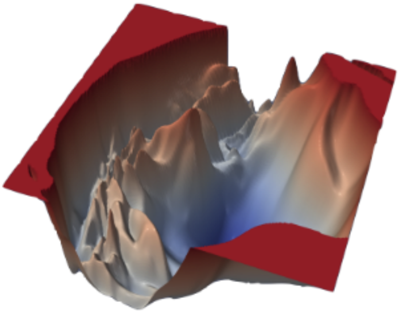
Loss Function



A slice of the loss landscape of resnet-110 with no skip connections Src: Visualizing the Loss Landscape of Neural Nets, Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein

- We wish to calculate weights $w_i, 1 \leq i \leq n$ so that the predicted values $\hat{Y}$ are as close as possible(to an extent) to the actual values Y.

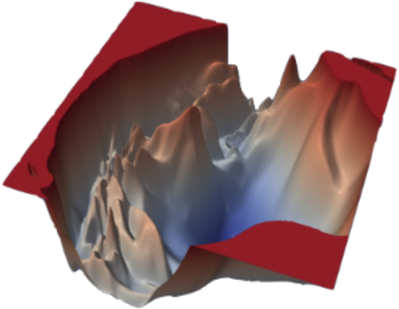
Loss Function



A slice of the loss landscape of resnet-110 with no skip connections Src: Visualizing the Loss Landscape of Neural Nets, Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein

- We wish to calculate weights $w_i, 1 \leq i \leq n$ so that the predicted values $\hat{Y}$ are as close as possible(to an extent) to the actual values Y.To do this we try to minimize a loss function $\mathcal{L}(\{\hat{Y}_i\}, \{Y_i\})$

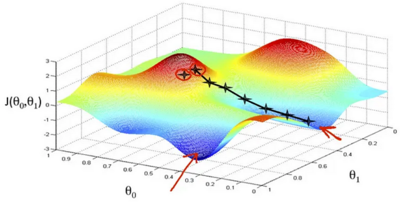
Loss Function



A slice of the loss landscape of resnet-110 with no skip connections Src: Visualizing the Loss Landscape of Neural Nets, Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein

- We wish to calculate weights $w_i, 1 \leq i \leq n$ so that the predicted values $\hat{Y}$ are as close as possible(to an extent) to the actual values Y.To do this we try to minimize a loss function $\mathcal{L}(\{\hat{Y}_i\}, \{Y_i\})$
- Common choices of $\mathcal{L}$ include MSE for continuous variables and cross-loss entropy for categorical variables.

Gradient descent

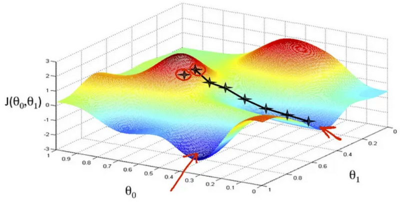


Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- To find the correct set of weights, we use a greedy approach. We check the surrounding landscape of the weight(i.e. calculate the gradient) and take a step in the direction which leads to maximum decrease in $\mathcal{L}$. Mathematically, we have:

$$w_i = wi - \eta \frac{\partial \mathcal{L}}{\partial w_i}$$

Gradient descent



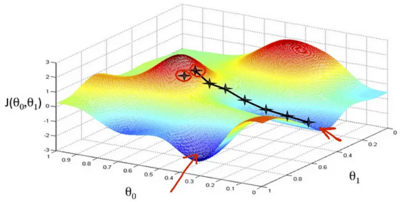
Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- To find the correct set of weights, we use a greedy approach. We check the surrounding landscape of the weight(i.e. calculate the gradient) and take a step in the direction which leads to maximum decrease in $\mathcal{L}$. Mathematically, we have:

$$w_i = wi - \eta \frac{\partial \mathcal{L}}{\partial w_i}$$

- η is known as the learning rate.

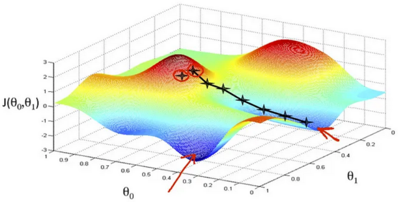
Learning Rate



Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- Fixing η is quite tricky.

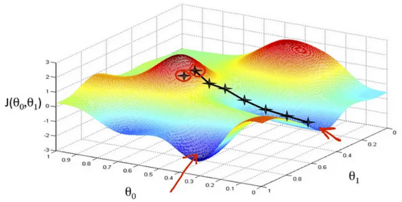
Learning Rate



Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- Fixing η is quite tricky.
- If η is small we get stuck at local minima

Learning Rate



Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- Fixing η is quite tricky.
- If η is small we get stuck at local minima
- If η is large we overshoot our targets and never converge.
- The best way to do things is to use a adaptive learn rate. Some methods(parametric, non-parametric and hybrid are discussed later, once we cover backpropagation)

Backpropagation

Consider an acyclic directed graph.

- We want to calculate the partial derivative of a node with respect to the other.

Backpropagation

Consider an acyclic directed graph.

- We want to calculate the partial derivative of a node with respect to the other.
- The way to do this is to use the chain rule

Backpropagation

Consider an acyclic directed graph.

- We want to calculate the partial derivative of a node with respect to the other.
- The way to do this is to use the chain rule
- When we use the chain rule, we sum the partial derivative along all the paths joining the two nodes
- It is easy to see that the computation needed explodes as the depth increases
- The way to navigate this problem is to use dynamic programming.

References