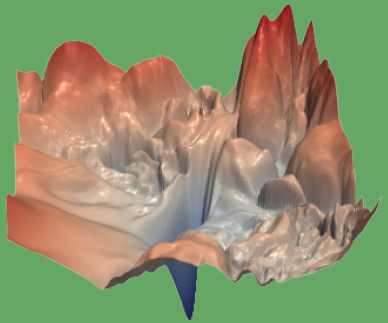
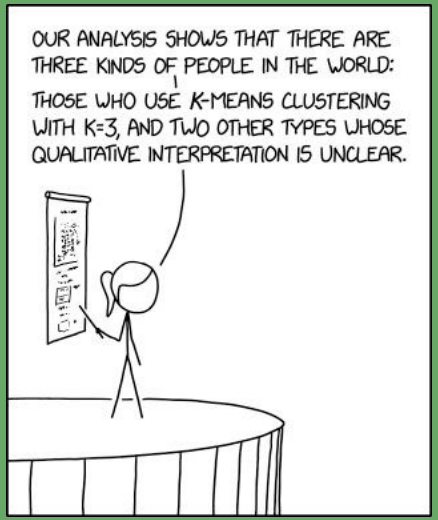
Deep Learning



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.

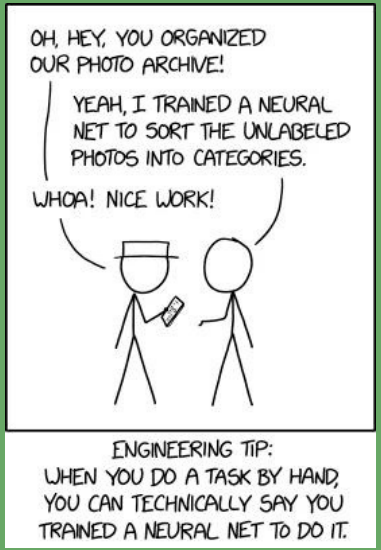
Problems with the traditional approach

- **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



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:

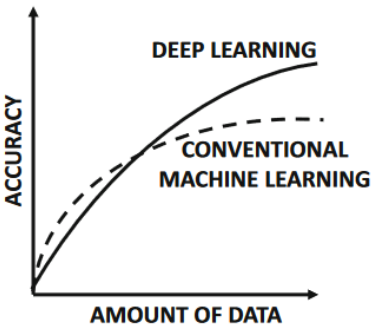
Ideas behind a neural network

- **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.

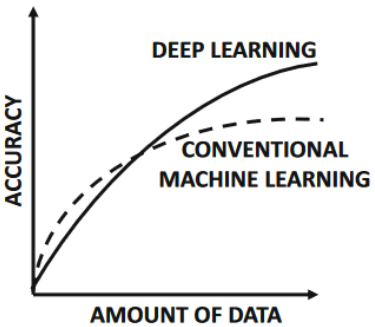
A practical comparison with traditional ML



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.

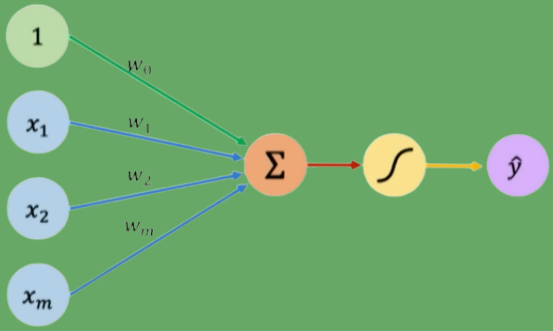
A practical comparison with traditional ML



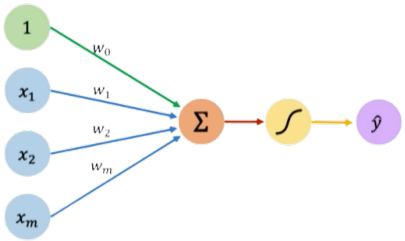
Comparision based on amount of data features.
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- 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



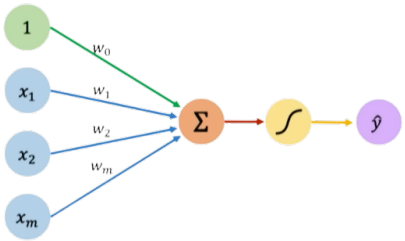
Non-Linearity is the key



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.

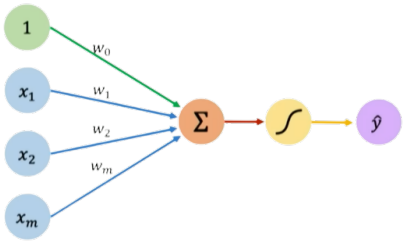
Non-Linearity is the key



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 $\hat{Y} = \sigma(W^T X + b_0)$

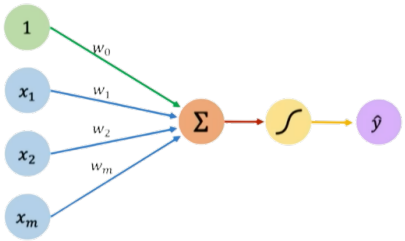
Non-Linearity is the key



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.

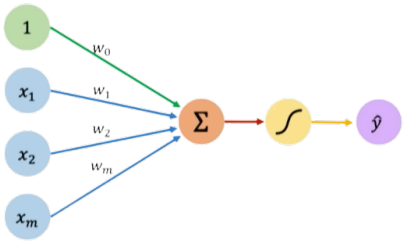
Non-Linearity is the key



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.

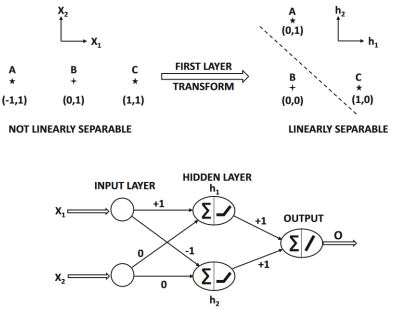
Non-Linearity is the key



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.

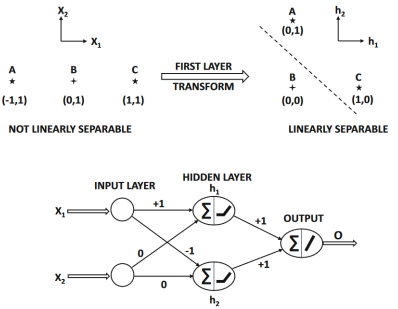
Non-Linearity is the key



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.)

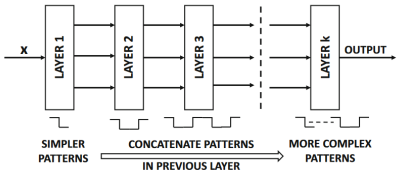
Non-Linearity is the key



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.

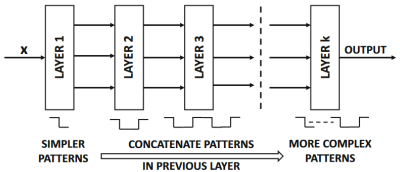
Increasing depth



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

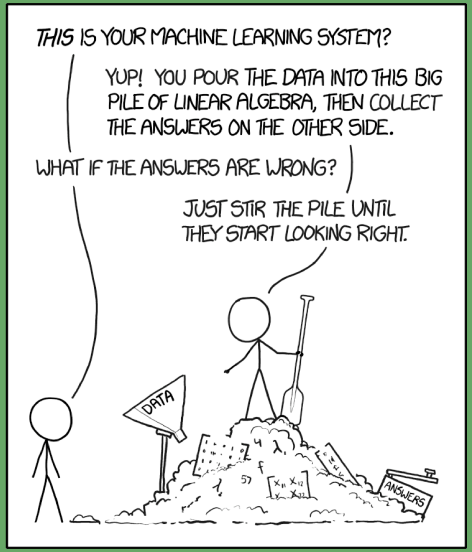


Why is depth needed
Src: Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

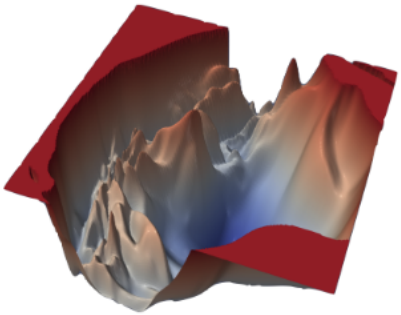
- While a single hidden layer is enough, making a deep neural network allows us to have more complex decision boundaries with relatively less number of nodes
- It should be noted how non-linearity discussed above plays an important role: Irrespective of number of layers, composition of linear functions 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



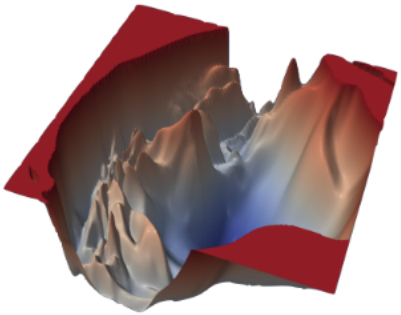
Loss Function



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.

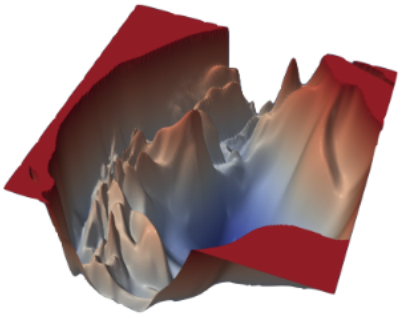
Loss Function



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 $\mathcal{L}(\{\hat{Y}_i\}, \{Y_i\})$

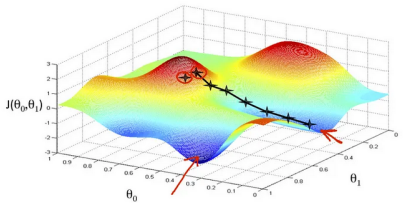
Loss Function



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

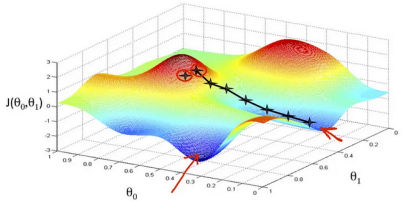


Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- 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



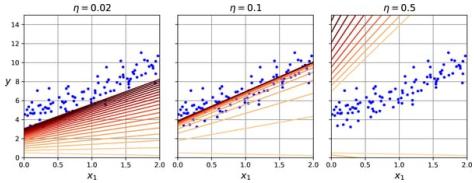
Gradient descent
Src: https://towardsdatascience.com/an-intuitive-explanation-of-gradient-descent-83adf68c9c33

- 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.

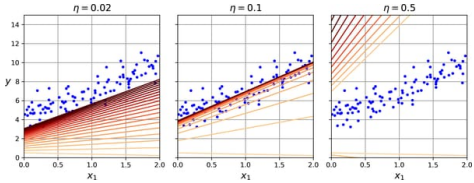
Learning Rate



Variations in learning rates
Src: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- Fixing η is quite tricky.

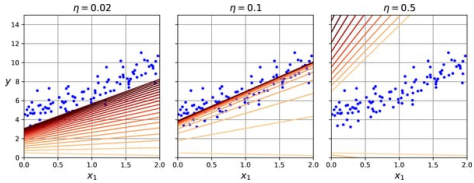
Learning Rate



Variations in learning rates
Src: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

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

Learning Rate



Variations in learning rates
Src: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- 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 a neural network to be an acyclic directed graph

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

Backpropagation

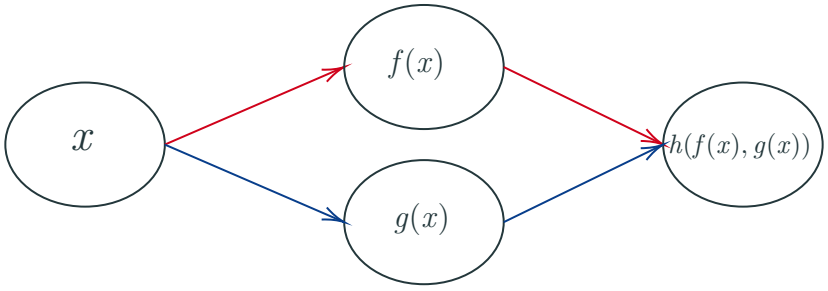
Consider a neural network to be an acyclic directed graph

- 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 a neural network to be 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.



$$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial f} \cdot \frac{\partial f}{\partial x} + \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial x}$$

Backpropagation

Roughly, the alogirithm works as follows:

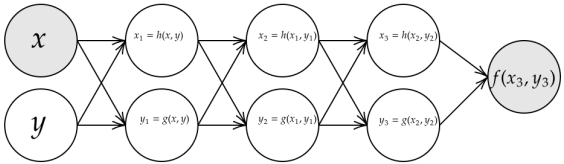
- Initialize the weights

Backpropagation

Roughly, the alogirithm works as follows:

- Initialize the weights
- Calculate $\mathcal{L}$. Note, that this step occurs from the input towards the output and is known as the forward phase.
- Calculate gradient. This process occurs from the output towards the input and is known as the backward phase.

Reverse Mode Auto Differentiation

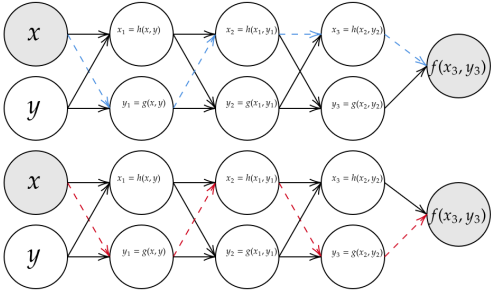


$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x_3} \cdot \frac{\partial x_3}{\partial x} + \frac{\partial f}{\partial y_3} \cdot \frac{\partial y_3}{\partial x}$$

$$= \frac{\partial f}{\partial x_3} \cdot \left(\frac{\partial x_3}{\partial x_2} \frac{\partial x_2}{\partial x} + \frac{\partial x_3}{\partial y_2} \frac{\partial y_2}{\partial x} \right) + \frac{\partial f}{\partial y_3} \cdot \left(\frac{\partial y_3}{\partial x_2} \frac{\partial x_2}{\partial x} + \frac{\partial y_3}{\partial y_2} \frac{\partial y_2}{\partial x} \right)$$

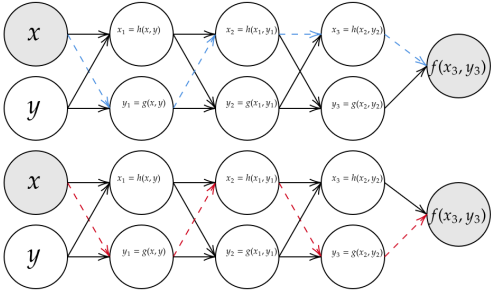
The whole idea relies on the fact that while calculating gradients, parts of the calculation are repeated.

Reverse Mode Auto Differentiation



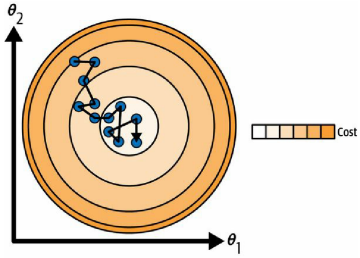
Sum is taken over the product along each path, over all the paths. It is trivial to note that parts of the paths are repeated.

Reverse Mode Auto Differentiation



Sum is taken over the product along each path, over all the paths. It is trivial to note that parts of the paths are repeated.In the toy example on the left, the paths are essentially the same, varying only for the second to last node.

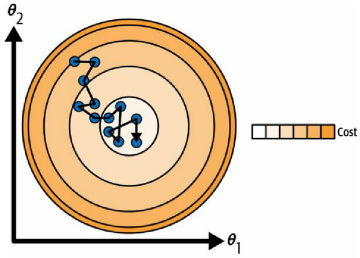
Gradient Descent Statergy-Stochastic Gradient Descent



Stochstic Gradient descent
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- Instead of calculating the total loss, we calculate the loss from a randomly picked sample(or a batch in case of batch gradient descent)

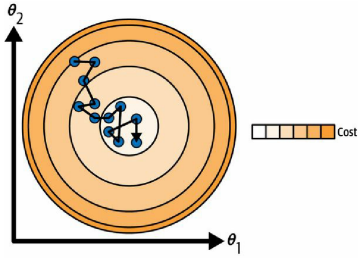
Gradient Descent Statergy-Stochastic Gradient Descent



Stochstic Gradient descent
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- Instead of calculating the total loss, we calculate the loss from a randomly picked sample(or a batch in case of batch gradient descent)
- This decreases the computational time.

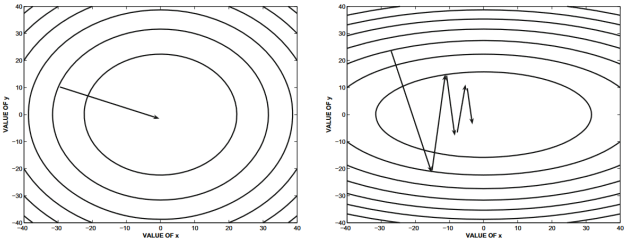
Gradient Descent Statergy-Stochastic Gradient Descent



Stochstic Gradient descent
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- Instead of calculating the total loss, we calculate the loss from a randomly picked sample(or a batch in case of batch gradient descent)
- This decreases the computational time.
- Think instead of taking slower but confident steps, we take faster but less-confident steps.

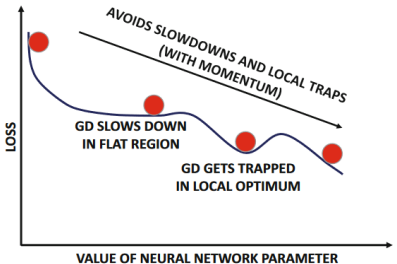
Gradient Descent Statergy- Normalization



Why normalize?
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

Normalizing features is a way to make the descent smoother. It essentially lowers the changes in direction orthogonal to the minima.

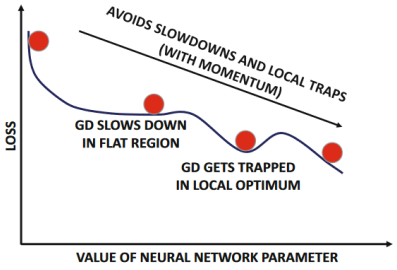
Gradient Descent Statergy- Mommentum



Need for momentum
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- A Momentum term might be used in gradient descent where consideration is made for a moving average ”velocity” of the descent.

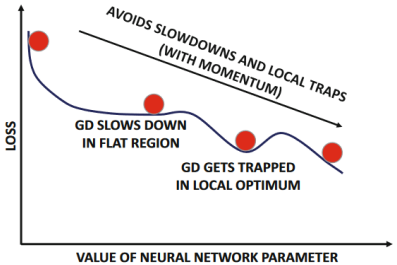
Gradient Descent Statergy- Mommentum



Need for momentum
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- A Momentum term might be used in gradient descent where consideration is made for a moving average ”velocity” of the descent.
- Particularly helps when there are local minimas and flat regions.

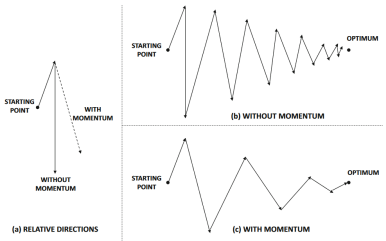
Gradient Descent Statergy- Mommentum



Need for momentum
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- A Momentum term might be used in gradient descent where consideration is made for a moving average ”velocity” of the descent.
- Particularly helps when there are local minimas and flat regions.
- Helps when there is a lot of ”zig-zag” but the descent heads in a certain direction.

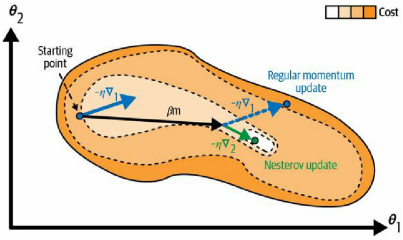
Gradient Descent Statergy- Mommentum



Need for momentum
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- A Momentum term might be used in gradient descent where consideration is made for a moving average ”velocity” of the descent.
- Particularly helps when there are local minimas and flat regions.
- Helps when there is a lot of ”zig-zag” but the descent heads in a certain direction.

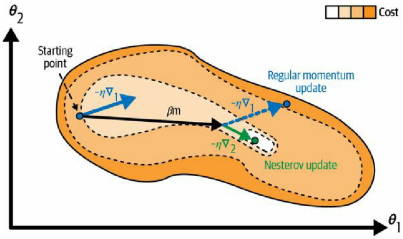
Gradient Descent Statergy- Nestov Mommentum



Nestov momentum
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- Nestov-momentum is similar to momentum with some scout ahead

Gradient Descent Statergy- Nestov Mommentum



Nestov momentum
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- Nestov-momentum is similar to momentum with some scout ahead
- Knowing what's coming up ahead further helps in correcting the direction of descent.

Gradient Descent Statergy- Adaptive Learning Rates

- We try to let different parameters have different learning rates.

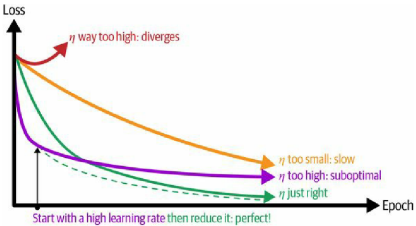
Gradient Descent Statergy- Adaptive Learning Rates

- We try to let different parameters have different learning rates.
- The idea is that parameters with large partial derivatives are often oscillating and zigzagging, whereas parameters with small partial derivatives tend to be more consistent but move in the same direction.

Gradient Descent Statergy- Adaptive Learning Rates

- We try to let different parameters have different learning rates.
- The idea is that parameters with large partial derivatives are often oscillating and zigzagging, whereas parameters with small partial derivatives tend to be more consistent but move in the same direction.
- Algorithms include AdaGrad, RMSProp and AdaDelta. Parametric algorithms combined with momentum considerations also exist: RMSProp with Nesterov Momentum, ADAM and it's variants like AdaMax, Nadam and AdamW. A quick comparison table is avilable at *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc by Aurélien Géron*

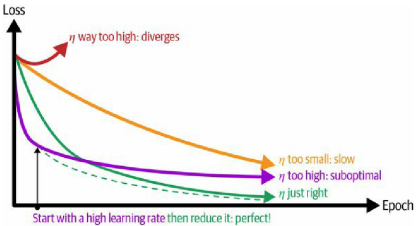
Gradient Descent Statergy- Learning Rate Scheduling



Nestov momentum
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- We have the learning rate high at first so that it gets a relative idea about where the minima lies

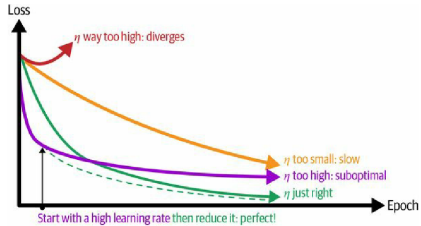
Gradient Descent Statergy- Learning Rate Scheduling



Nestov momentum
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- We have the learning rate high at first so that it gets a relative idea about where the minima lies, and then we lower it to find it with more accuracy

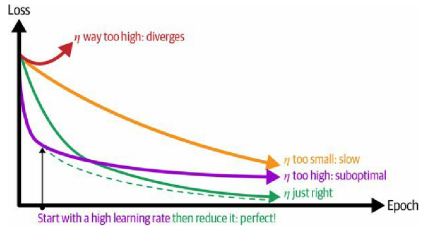
Gradient Descent Statergy- Learning Rate Scheduling



Nestov momentum
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- We have the learning rate high at first so that it gets a relative idea about where the minima lies, and then we lower it to find it with more accuracy
- Scheduling is done on numbers of iterations completed(each iteration is also called an *epoch*)

Gradient Descent Statergy- Learning Rate Scheduling



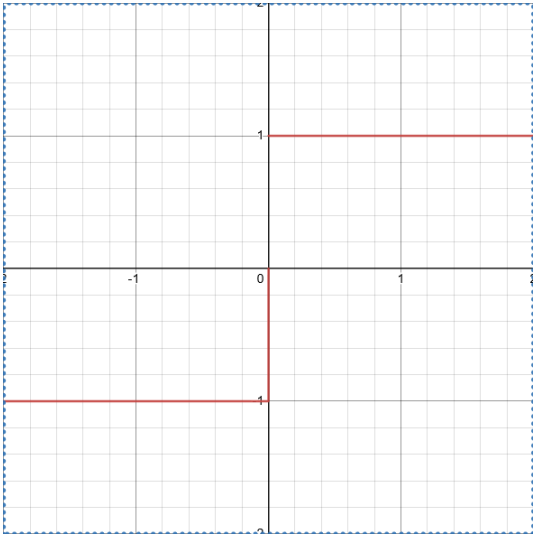
Nestov momentum
Src:Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media, Inc, Aurélien Géron

- We have the learning rate high at first so that it gets a relative idea about where the minima lies, and then we lower it to find it with more accuracy
- Scheduling is done on numbers of iterations completed(each iteration is also called an *epoch*)

Gradient Descent Statergy- One cycle scheduling

- The basic idea behind one cycle scheduling is to start with a low learning rate, gradually increase it to a maximum value, and then decrease it again to a low value. This approach helps the network explore a wide range of learning rates, allowing it to quickly converge to a good solution and potentially escape from local minima.
- By using a cyclical learning rate schedule, one cycle scheduling aims to strike a balance between exploration and exploitation in the learning process. It enables the network to quickly explore a wide range of learning rates at the beginning and then gradually refine its weights as the learning rate decreases.

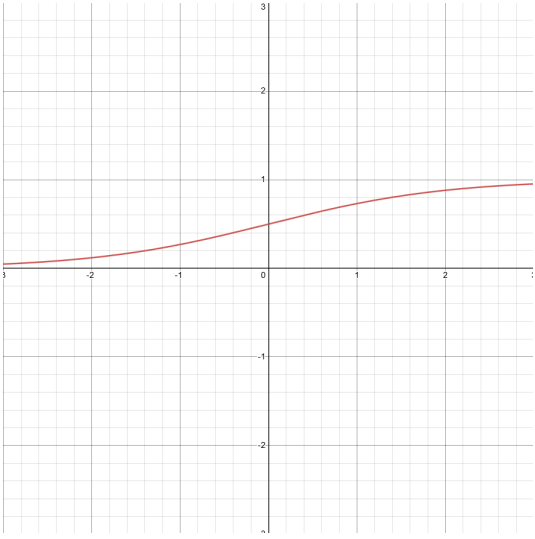
Gradient Descent Strategy- Choosing an activation function



$sgn(x)$

One of the first activation functions to be considered was $sgn(x)$ mostly because this was the function based on which our neurons operate.

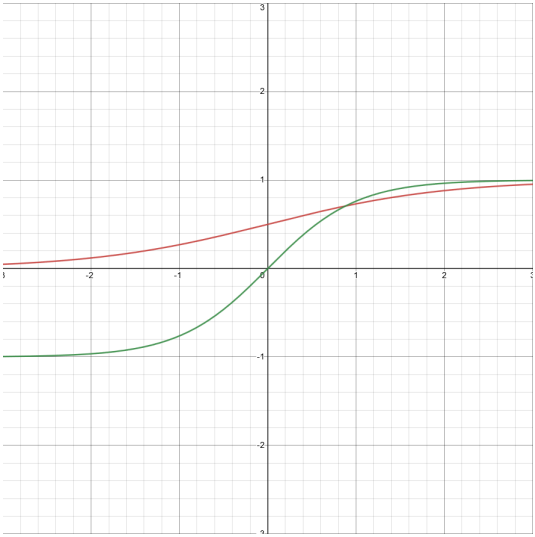
Gradient Descent Strategy- Choosing an activation function



sigmoid

But soon better activation functions were found like the sigmoid function

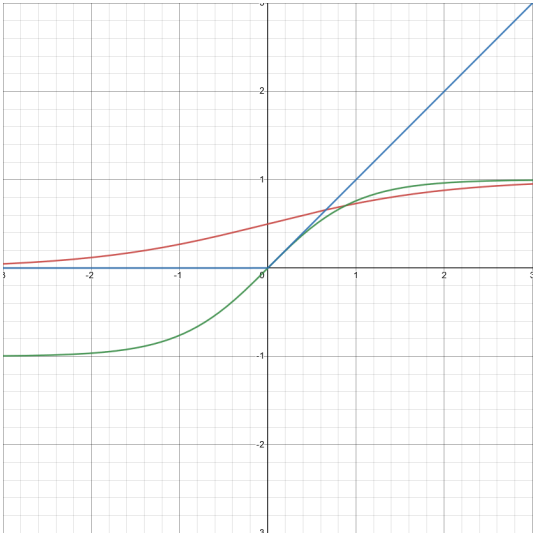
Gradient Descent Strategy- Choosing an activation function



sigmoid, tanh

But soon better activation functions were found like the sigmoid function, tanh function

Gradient Descent Strategy- Choosing an activation function



sigmoid, tanh, relu

But soon better activation functions were found like the sigmoid function, tanh function and relu function.

Gradient Descent Strategy- Dead Neurons and Vanishing & Exploding Gradients

sigmoid, tanh, relu

- Most activation function saturated. Therefore, high values meant no gradient ”propagated” forward. This leads to neurons which rarely activated and became ”dead”.

Gradient Descent Strategy- Dead Neurons and Vanishing & Exploding Gradients

sigmoid, tanh, relu

- Most activation function saturated. Therefore, high values meant no gradient "propagated" forward. This leads to neurons which rarely activated and became "dead".
- The effect of derivatives magnifies down the layers. If the order of magnitude is above 1(say 10) then it explodes(after 10 layers it will become 10^{10}).

Gradient Descent Strategy- Dead Neurons and Vanishing & Exploding Gradients

sigmoid, tanh, relu

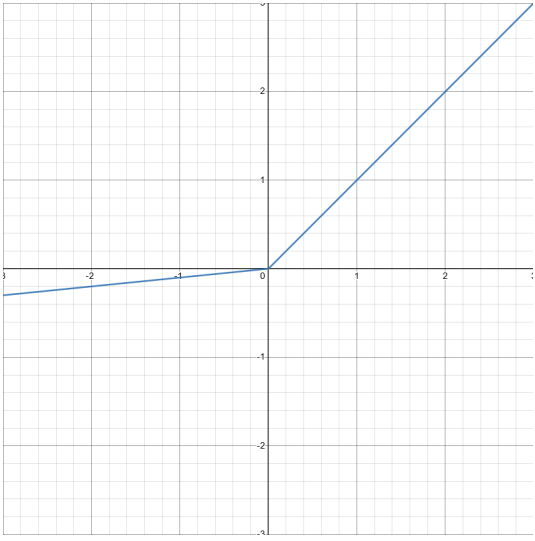
- The effect of derivatives magnifies down the layers. If the order of magnitude is below 1(say $1/10$) then it vanishes(after 10 layers it will become 10^{-10})

Gradient Descent Strategy- Dead Neurons and Vanishing & Exploding Gradients

sigmoid, tanh, relu

- The effect of derivatives magnifies down the layers. If the order of magnitude is below 1(say $1/10$) then it vanishes(after 10 layers it will become 10^{-10})
- From a more practical viewpoint, this might lead to overflow if the gradient explodes. If it is vanishing, then there might not be enough precision to handle the values.

Gradient Descent Strategy- Better Activation Functions

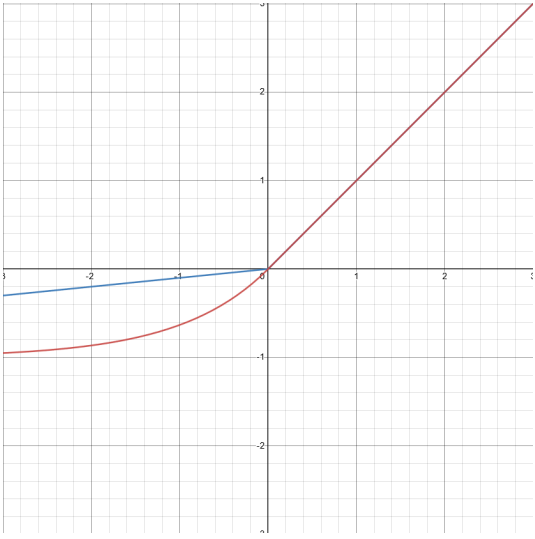


leaky ReLu

One way to alleviate those problems is to use better activation function which are in a way variants of ReLu. Some examples include

- Leaky ReLu $(\max(\alpha z, z))$

Gradient Descent Strategy- Better Activation Functions

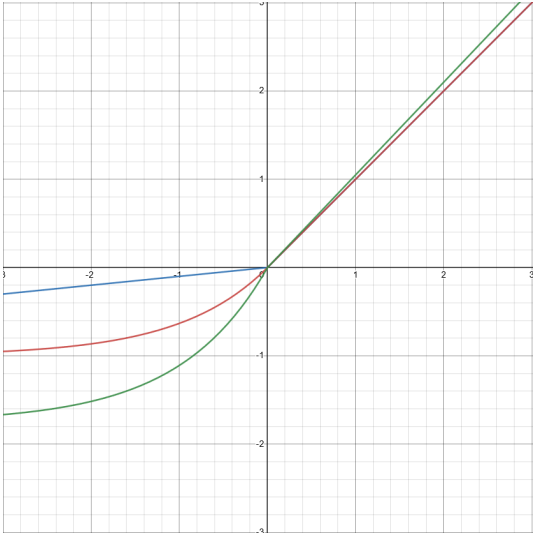


leaky ReLu, ELU

One way to alleviate those problems is to use better activation function which are in a way variants of ReLu. Some examples include

- Leaky ReLu $(\max(\alpha z, z))$
- ELU$((\alpha(e^z - 1)$if $z < 0$ else $z))$

Gradient Descent Strategy- Better Activation Functions

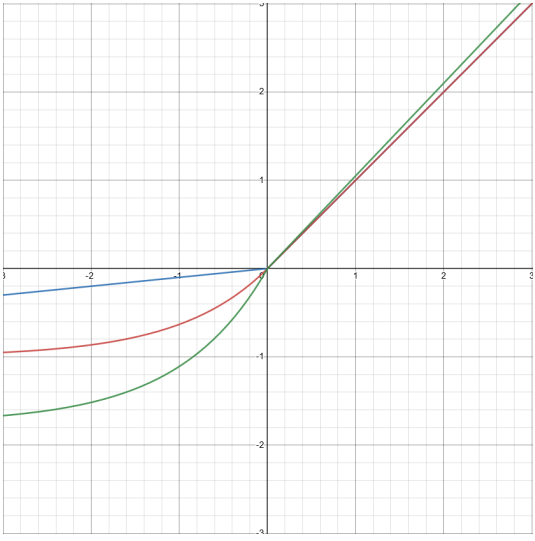


leaky ReLu, ELU, SELU

One way to alleviate those problems is to use better activation function which are in a way variants of ReLu. Some examples include

- Leaky ReLu ($\max(\alpha z, z)$)
- ELU($(\alpha(e^z - 1)$if $z < 0$ else $z)$)
- SELU($1.05ELU_{1.67}$)

Gradient Descent Strategy- Better Activation Functions



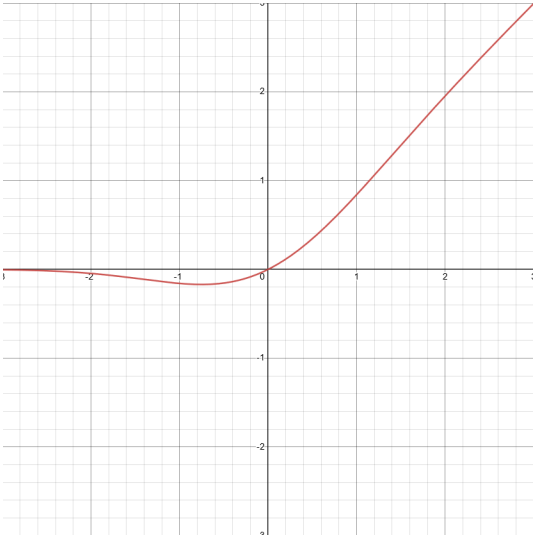
leaky ReLu, ELU, SELU

One way to alleviate those problems is to use better activation function which are in a way variants of ReLu. Some examples include

- Leaky ReLu ($\max(\alpha z, z)$)
- ELU($(\alpha(e^z - 1)$if $z < 0$ else $z)$)
- SELU($1.05ELU_{1.67}$)

One can also treat α as a parameter to be learned by back prop.

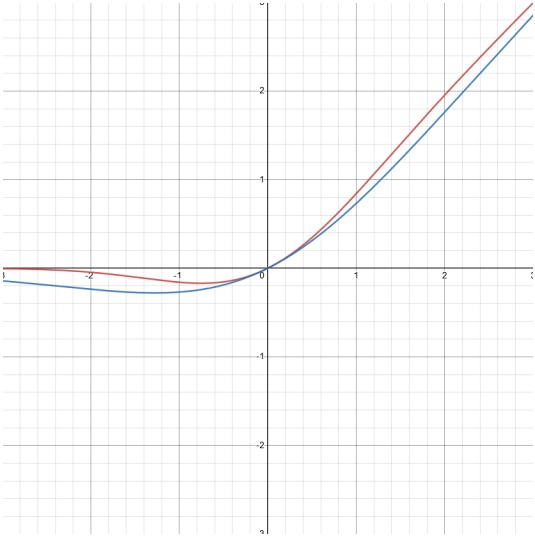
Gradient Descent Strategy- Better Activation Functions



Other activation functions also exists such as

- GELU $(z\Phi(z))$where Φ is the CDF of the standard normal

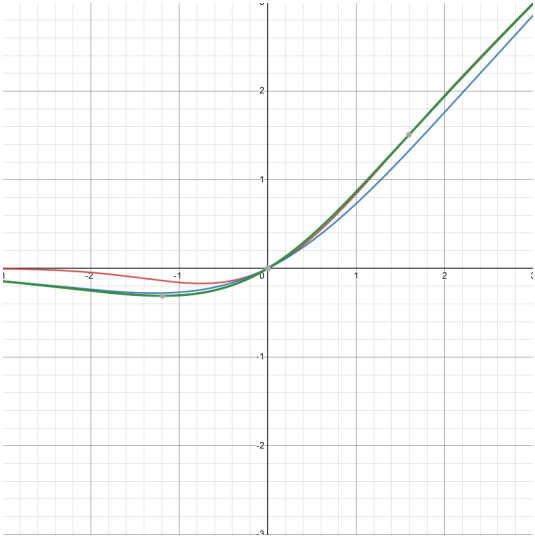
Gradient Descent Strategy- Better Activation Functions



Other activation functions also exists such as

- GELU $(z\Phi(z))$where Φ is the CDF of the standard normal.
- SWISH $(z\sigma(z))$where σ is the sigmoid function.

Gradient Descent Strategy- Better Activation Functions



Other activation functions also exists such as

- GELU $(z\Phi(z))$where Φ is the CDF of the standard normal.
- SWISH $(z\sigma(z))$where σ is the sigmoid function.
- MISH $(z\tanh(\ln(1+exp(z))))$where σ is the sigmoid function.

Gradient Descent Strategy- Weight Initialization

- Initially weights were initialized by picking from a standard normal distribution.

Gradient Descent Strategy- Weight Initialization

- Initially weights were initialized by picking from a standard normal distribution.
- The difference in the variance of the outgoing nodes and incoming nodes(if there is a difference in number of nodes between layers) cause practical problems in the flow of gradients.

Gradient Descent Strategy- Weight Initialization

- Initially weights were initialized by picking from a standard normal distribution.
- The difference in the variance of the outgoing nodes and incoming nodes(if there is a difference in number of nodes between layers) cause practical problems in the flow of gradients.
- A good compromise is to take average of number of outgoing edges and number of incoming edges and normalize the variance of the normal distribution used by it. This is called Glorot initialization

Gradient Descent Strategy- Weight Initialization

- Initially weights were initialized by picking from a standard normal distribution.
- The difference in the variance of the outgoing nodes and incoming nodes(if there is a difference in number of nodes between layers) cause practical problems in the flow of gradients.
- A good compromise is to take average of number of outgoing edges and number of incoming edges and normalize the variance of the normal distribution used by it. This is called Glorot initializationThe same principal when used with uniform distribution is called He initialization.

Gradient Descent Strategy- Gradient Clipping

- In case one of the components of the gradient is much lager than other we might want to normalize it to control our descent.

Gradient Descent Strategy- Gradient Clipping

- In case one of the components of the gradient is much lager than other we might want to normalize it to control our descent.
- One way it to just limit it at some maximum value.

Gradient Descent Strategy- Gradient Clipping

- In case one of the components of the gradient is much lager than other we might want to normalize it to control our descent.
- One way it to just limit it at some maximum value.
- Other methods include normalizing the whole vector with respect to a norm.

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.But depth comes with its own share of problems.

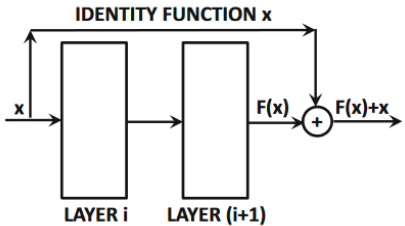
Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.But depth comes with its own share of problems.
- Deeper networks have a harder time converging.

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.But depth comes with its own share of problems.
- Deeper networks have a harder time converging.
- If activation functions are not combined properly then there might be presence of local minimas

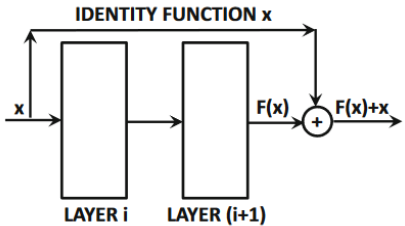
Aspects of a deep neural network-skip connections



Skip connections
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- We might allow nodes from deeper layers to have direct access to nodes from shallower layers while *skipping* the intermediate layers.

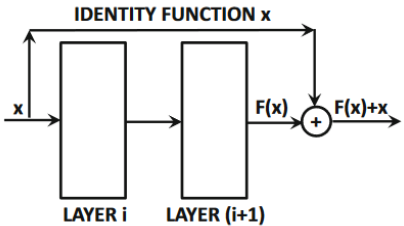
Aspects of a deep neural network-skip connections



Skip connections
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- We might allow nodes from deeper layers to have direct access to nodes from shallower layers while *skipping* the intermediate layers.
- This provides a kind of highway for the deeper layers to access simpler features.

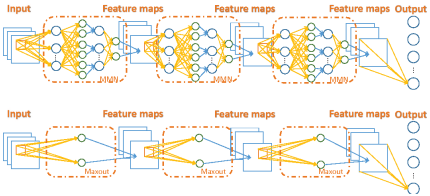
Aspects of a deep neural network-skip connections



Skip connections
Src:Neural Networks and Deep Learning: A Textbook, Charu C. Aggarwal

- We might allow nodes from deeper layers to have direct access to nodes from shallower layers while *skipping* the intermediate layers.
- This provides a kind of highway for the deeper layers to access simpler features.
- This is especially used in CNNs.

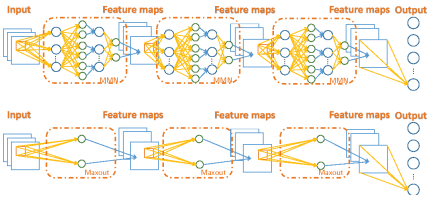
Aspects of a deep neural network-maxout networks



Max-out connections
Src:Improving deep neural networks with multilayer maxout networks, Weichen Sun, Fei Su, Leiquan Wang

- In a certain sense this is type of ensemble method.

Aspects of a deep neural network-maxout networks



Max-out connections
Src:Improving deep neural networks with multilayer maxout networks, Weichen Sun, Fei Su, Leiquan Wang

- In a certain sense this is type of ensemble method.
- Two sets of weights W_1, W_2 are trained for each input.
- The activation is set to $\sigma\left(\max(W_1 \cdot X_{in}, W_2 \cdot X_{in}) + b_0\right)$

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.But depth comes with its own share of problems.

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.But depth comes with its own share of problems.
- Deeper networks have a harder time converging.

Aspects of a deep neural network-navigating the loss landscape

- Generally, as far as decision boundaries are concerned, adding depth gives ore bang for the buck than adding more nodes in a single layer.But depth comes with its own share of problems.
- Deeper networks have a harder time converging.
- If activation functions are not combined properly then there might be presence of local minimas

References