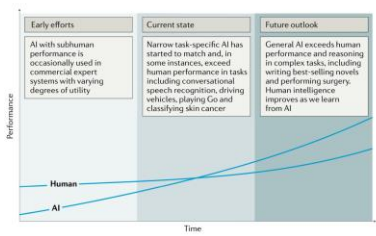
Deep Learning Intuition

Ahmed Hosny

HARVARD MEDICAL SCHOOL | BRIGHAM AND WOMEN'S HOSPITAL | DANA-FARBER CANCER INSTITUTE

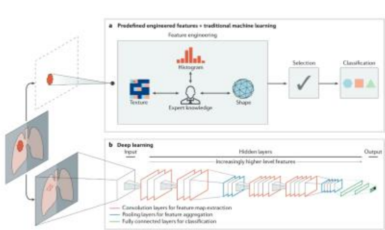
SAM Joint Imaging-Therapy Scientific Symposium (Certificate Series Session 2)
Machine Learning for Radiomics - Wednesday, 8/1/2018 10:15 AM - 12:15 PM

Deep Learning



Ahmed Hosny, Chintan Parmar, John Quackenbush, Lawrence H Schwartz and Hugo JWL Aerts
Artificial Intelligence in Radiology
Nature Reviews Cancer - 2018

Deep Learning



Ahmed Hosny, Chintan Parmar, John Quackenbush, Lawrence H Schwartz and Hugo JWL Aerts
Artificial Intelligence in Radiology
Nature Reviews Cancer - 2018

What is the intuition behind neural networks?

How do neural networks learn?

How to train neural networks?

What is the intuition behind neural networks?

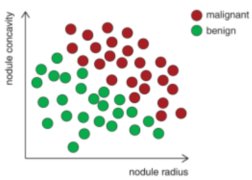
How do neural networks learn?

How to train neural networks?

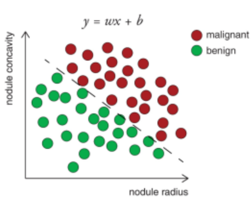
Machine Learning: 4 Main Components

- Data
- Model/Representation
- Cost/Error/Loss of model
- Model Optimizer

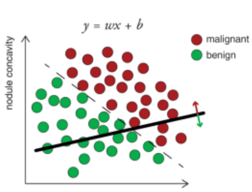
Logistic Regression



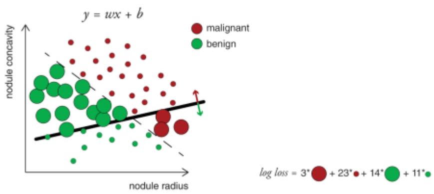
Logistic Regression



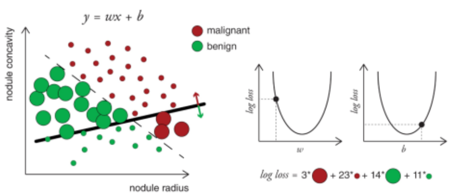
Logistic Regression



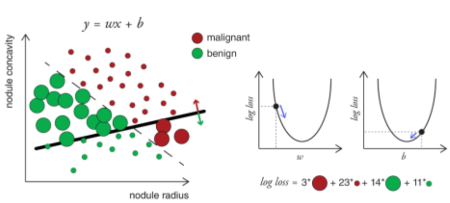
Logistic Regression



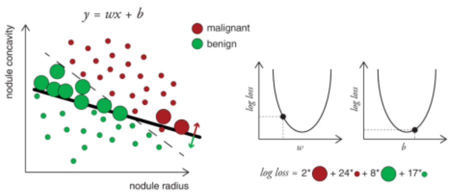
Logistic Regression



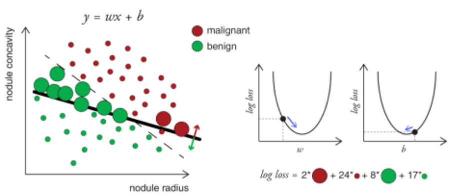
Logistic Regression



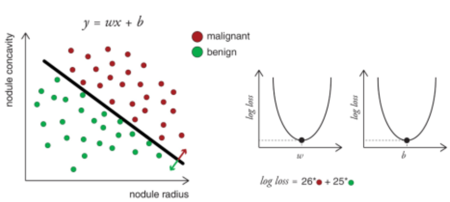
Logistic Regression



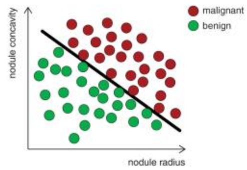
Logistic Regression



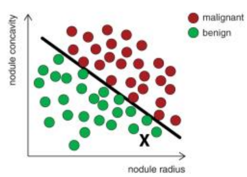
Logistic Regression



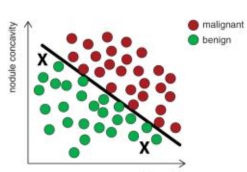
Dealing with Edge Conditions



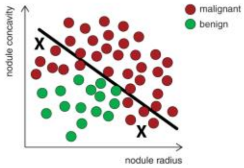
Dealing with Edge Conditions



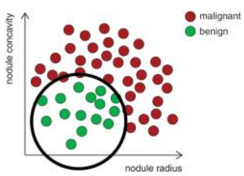
Dealing with Edge Conditions



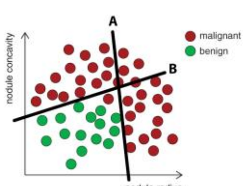
Dealing with Edge Conditions



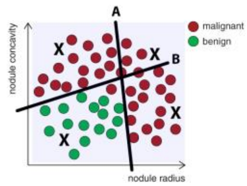
Dealing with Edge Conditions



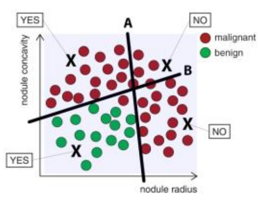
Dealing with Edge Conditions



Quadrant Questioning

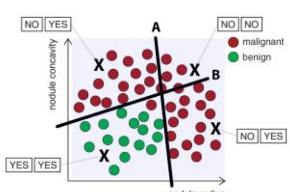


Quadrant Questioning



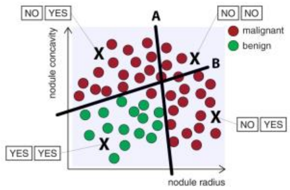
- Line A, over or under?

Quadrant Questioning



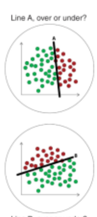
- Line A, over or under?
- Line B, over or under?

Quadrant Questioning

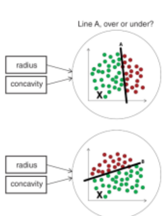


- Line A, over or under?
- Line B, over or under?
- Both YES?

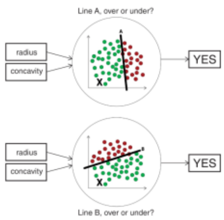
Graph Representation



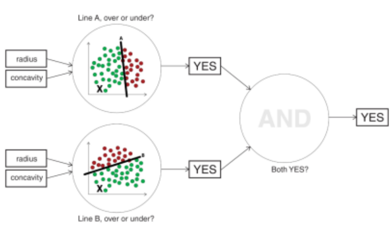
Graph Representation



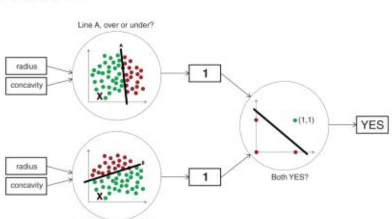
Graph Representation



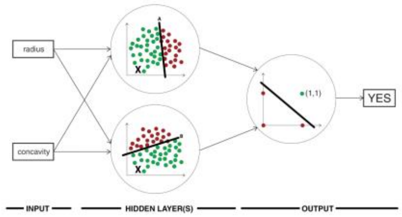
Graph Representation



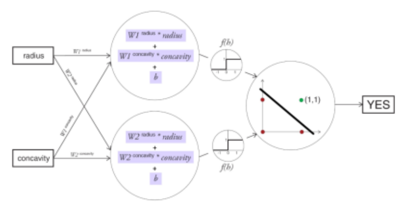
Graph Representation



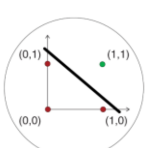
A Neural Network



A Neural Network

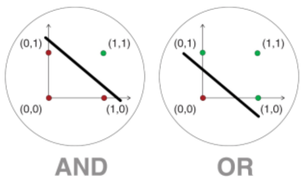


XOR Perceptron

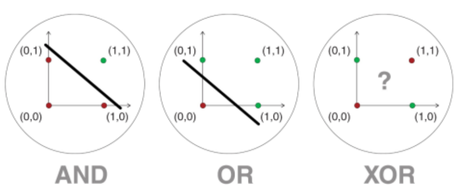


AND

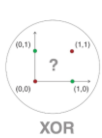
XOR Perceptron



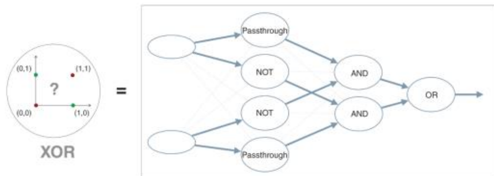
XOR Perceptron



XOR Perceptron

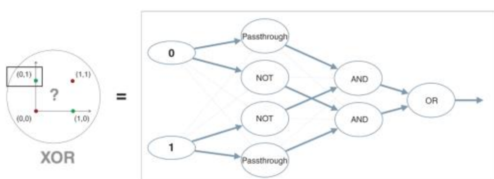


XOR Perceptron



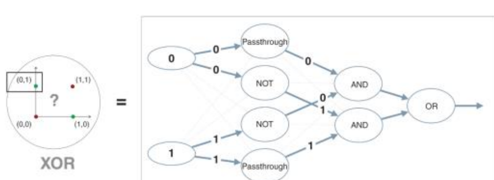
Lesson 2: Intro To Neural Networks
Udacity Deep Learning Nanodegree Program

XOR Perceptron



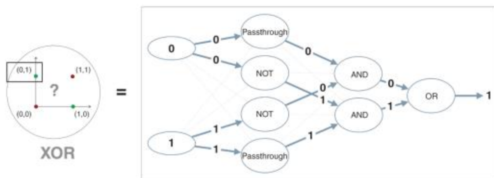
Lesson 2: Intro To Neural Networks
Udacity Deep Learning Nanodegree Program

XOR Perceptron



Lesson 2: Intro To Neural Networks
Udacity Deep Learning Nanodegree Program

XOR Perceptron



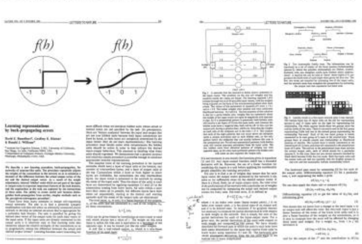
Lesson 2: Intro To Neural Networks
Udacity Deep Learning Nanodegree Program

What is the intuition behind neural networks?

How do neural networks learn?

How to train neural networks?

Backpropagation & Gradient Descent in Neural Networks



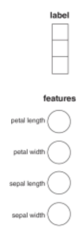
David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams
Learning Representations by Back-propagating Errors
Nature - 1986

Iris Dataset

R.A. Fisher
The Use of Multiple Measurements in Taxonomic Problems
Annual Eugenics - 1936

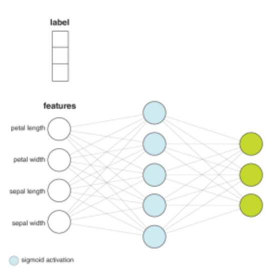
How Neural Networks Learn

✓ **Data:** iris dataset



How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network



sigmoid

$$S(x) = \frac{1}{1+e^{-x}}$$

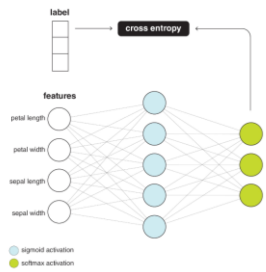
softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

How Neural Networks Learn

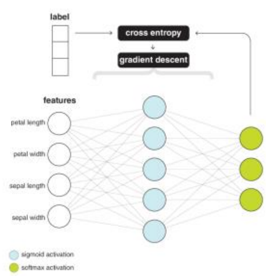
- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy

↑ **likelihood**
=
↑ **log(likelihood)**
=
↓ **-log(likelihood)**



How Neural Networks Learn

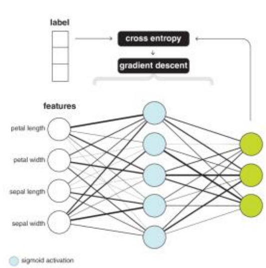
- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent



How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

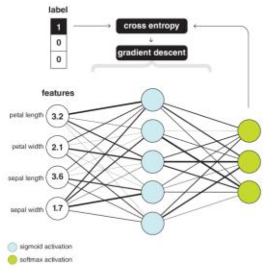
1. parameter initialization



How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network
✓ **Loss:** cross entropy
✓ **Optimizer:** gradient descent

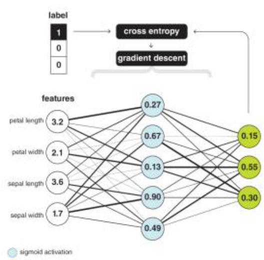
1. parameter initialization
2. data input



How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network
✓ **Loss:** cross entropy
✓ **Optimizer:** gradient descent

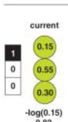
1. parameter initialization
2. data input
3. forward propagation

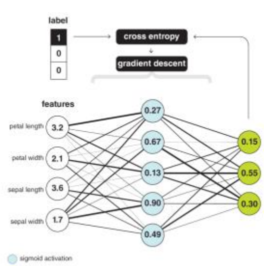


How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network
✓ **Loss:** cross entropy
✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation



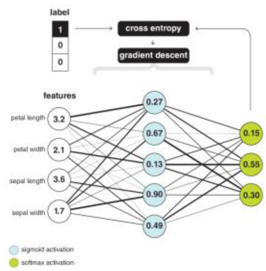


How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation

	current	best
1	0.15	1.00
0	0.55	0.00
0	0.30	0.00
	-log(0.15) 0.82	-log(1) 0.00

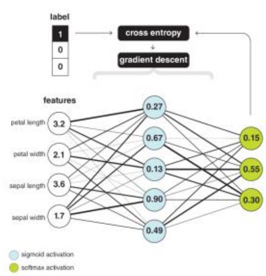


How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation

	current	best	worst
1	0.15	1.00	0.00
0	0.55	0.00	0.60
0	0.30	0.00	0.40
	-log(0.15) 0.82	-log(1) 0.00	-log(0) -(-∞) = ∞

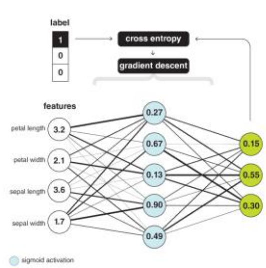


How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

$$\Delta w \propto -\text{gradient}$$
$$= -\frac{\partial L}{\partial w}$$
$$= -\eta \frac{\partial L}{\partial w}$$

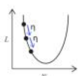


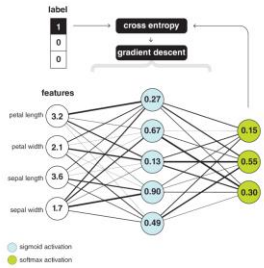
How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network
✓ **Loss:** cross entropy
✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

$$\Delta w \propto -\text{gradient} = -\frac{\partial L}{\partial w} = -\eta \frac{\partial L}{\partial w}$$



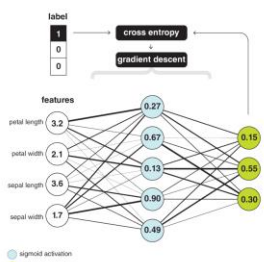


How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network
✓ **Loss:** cross entropy
✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

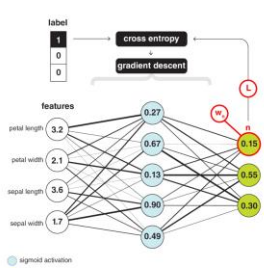
f(f(input))



How Neural Networks Learn

✓ **Data:** iris dataset
✓ **Model:** 3-layer neural network
✓ **Loss:** cross entropy
✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

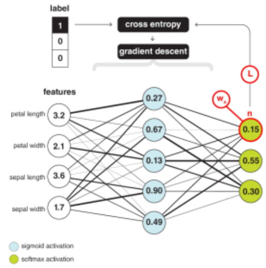


How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

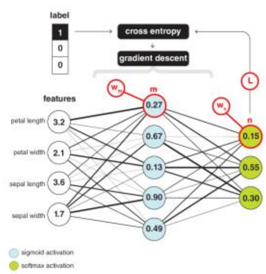
$$\Delta w_n = -\eta \frac{\partial L}{\partial w_n}$$
$$= -\eta \frac{\partial L}{\partial so}\frac{\partial so}{\partial n}\frac{\partial n}{\partial w_n}$$



How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

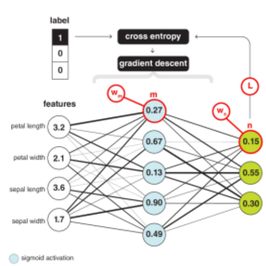


How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates

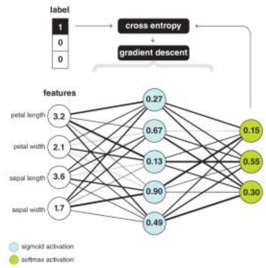
$$\Delta w_m = -\eta \frac{\partial L}{\partial w_m}$$
$$= -\eta \frac{\partial L}{\partial so}\frac{\partial so}{\partial n}\frac{\partial n}{\partial si}\frac{\partial si}{\partial m}\frac{\partial m}{\partial w_m}$$



How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

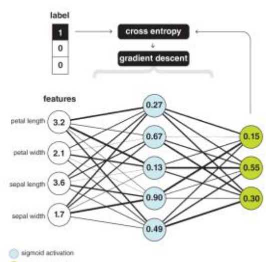
1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates



How Neural Networks Learn

- ✓ **Data:** iris dataset
- ✓ **Model:** 3-layer neural network
- ✓ **Loss:** cross entropy
- ✓ **Optimizer:** gradient descent

1. parameter initialization
2. data input
3. forward propagation
4. loss calculation
5. backpropagation + updates
6. repeat 2,3,4, & 5



Gradient Descent Flavors

vanilla gradient descent - entire dataset
stochastic gradient descent - random batch of samples (IID)
online gradient descent - (need not be IID)

Gradient Descent Flavors

vanilla gradient descent - entire dataset
stochastic gradient descent - random batch of samples (IID)
online gradient descent - (need not be IID)

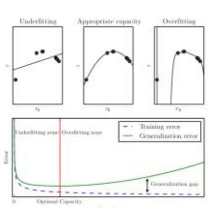
learning rate **batch size** **# of epochs**

What is the intuition behind neural networks?

How do neural networks learn?

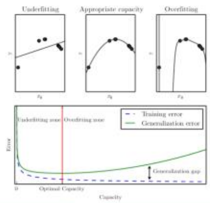
How to train neural networks?

The Perfect Fit



Ian Goodfellow, Yoshua Bengio & Aaron Courville
Deep Learning
MIT Press - 2016

The Perfect Fit



parameters vs **hyperparameters**

Ian Goodfellow, Yoshua Bengio & Aaron Courville
Deep Learning
MIT Press - 2016

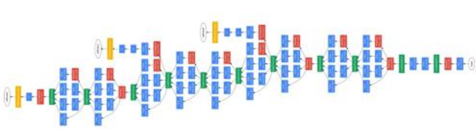
Hyperparameters

model-specific vs **optimizer-specifc**

model-specific	optimizer-specifc
architecture	learning rate
activations	batch size
initializations	# of epochs
loss functions	
optimizers	
regularizers	

Architecture

# of layers
# of units/layer



Christian Szegedy, Wei Liu, Yangqing Jia, et al.
Going Deeper with Convolutions (GoogleNet/Inception)
CVPR - 2015

Activations

step
- ✘ non-differentiable

Activations

step
- ✘ non-differentiable

sigmoid
- ✔ smooth + step-like
- ✔ good activations close to 0
- ✔ activations are bound 0~1
- ✘ vanishing gradients

Activations

step
- ✘ non-differentiable

sigmoid
- ✔ smooth + step-like
- ✔ good activations close to 0
- ✔ activations are bound 0~1
- ✘ vanishing gradients

tanh
- ✔ scaled sigmoid
- ✔ stronger activations
- ✘ vanishing gradients

Yann LeCun, Leon Bottou, Genevieve B. Orr & Klaus -Robert Müller
Efficient BackProp
Neural Networks: Tricks of the Trade - 1998

Activations

step
✗ non-differentiable

sigmoid
✓ smooth + step-like
✓ good activations close to 0
✓ activations are bound 0~1
✗ vanishing gradients

tanh
✓ scaled sigmoid
✓ stronger activations
✗ vanishing gradients

ReLU
✓ sparse activations
✓ efficient
✗ dead nodes
✗ not bound

Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton
ImageNet Classification with Deep Convolutional Neural Networks
Advances in Neural Information Processing - NIPS 2012

Activations

step
✗ non-differentiable

sigmoid
✓ smooth + step-like
✓ good activations close to 0
✓ activations are bound 0~1
✗ vanishing gradients

tanh
✓ scaled sigmoid
✓ stronger activations
✗ vanishing gradients

ReLU
✓ sparse activations
✓ efficient
✗ dead nodes
✗ not bound

Leaky ReLU
✓ no dead nodes

Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification
International Conference on Computer Vision - ICCV 2015

Activations

step
✗ non-differentiable

sigmoid
✓ smooth + step-like
✓ good activations close to 0
✓ activations are bound 0~1
✗ vanishing gradients

tanh
✓ scaled sigmoid
✓ stronger activations
✗ vanishing gradients

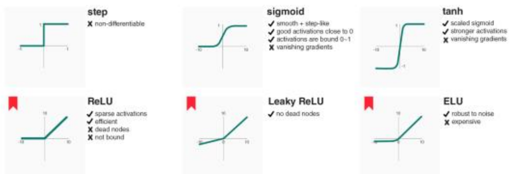
ReLU
✓ sparse activations
✓ efficient
✗ dead nodes
✗ not bound

Leaky ReLU
✓ no dead nodes

ELU
✓ robust to noise
✗ expensive

Djork-Arné Clevert, Thomas Unterthiner & Sepp Hochreiter
Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)
International Conference on Computer Vision - ICCV 2015

Activations

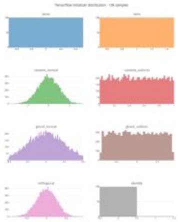


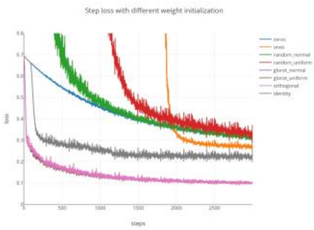
Initializations

0 - stuck at a saddle point
constants - difficult to break the symmetry
large random values - small gradients, slow convergence

Initializations

Initializations





Initializations

Name	α	β	γ	Reference
Constant	$\alpha = 0$	$\beta = 0$	$\gamma \geq 0$	used by [ZF14]
→ Xavier/Glorot uniform	$\alpha = \sqrt{\frac{6}{n_{in}+n_{out}}}$	$\beta = 0$	$\gamma = 0$	[GB10]
→ Xavier/Glorot normal	$\alpha = 0$	$\beta = \left(\frac{2}{(n_{in}+n_{out})}\right)^2$	$\gamma = 0$	[GB10]
→ He	$\alpha = 0$	$\beta = \frac{2}{n_{in}}$	$\gamma = 0$	[HZRS15b]
Orthogonal	—	—	$\gamma = 0$	[SMG13]
LSUV	—	—	$\gamma = 0$	[MM15]

Table B.2.: Weight initialization schemes of the form $w \sim \alpha \cdot \mathcal{U}[-1, 1] + \beta \cdot \mathcal{N}(0, 1) + \gamma$. n_{in}, n_{out} are the number of units in the previous layer and the next layer. Typically, biases are initialized with constant 0 and weights by one of the other schemes to prevent unit-coadaptation. However, dropout makes it possible to use constant initialization for all parameters.
LSUV and Orthogonal initialization cannot be described with this simple pattern.

Martin Thoma
Analysis and Optimization of Convolutional Neural Network Architectures
https://arxiv.org/pdf/1707.09725.pdf

Loss Functions

regression - mean squared error

multiclass classification - categorical cross entropy

pixel classification - dice/ Wasserstein dice coefficient

Optimizers

stochastic gradient descent + momentum

Optimizers

stochastic gradient descent + momentum

adaptive gradient (AdaGrad)

John Duchi, Elad Hazan & Yoram Singer
Adaptive Subgradient Methods for Online Learning and Stochastic Optimization
Journal of Machine Learning Research - 2011

Optimizers

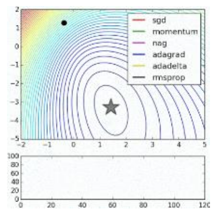
stochastic gradient descent + momentum

adaptive gradient (AdaGrad)

root mean square propagation (RMSProp)

Geoffrey Hinton
Coursera: Neural Networks for Machine Learning - Lecture 6
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Optimizers



https://imgur.com/a/Hqolp

Regularizers

L1, L2 regularization

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2}|W|$$

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2}W^2$$

Regularizers

L1, L2 regularization

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2}|W|$$

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2}W^2$$

https://alistapart.com/article/accessibility-whack-a-mole by Eleanor Ratliff

Regularizers

dropout

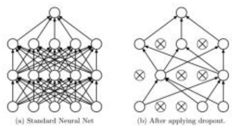
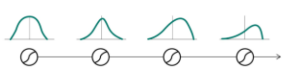


Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever & Ruslan Salakhutdinov
Dropout: A Simple Way to Prevent Neural Networks from Overfitting
Journal of Machine Learning Research - 2014

Regularizers

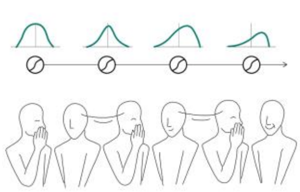
batch normalization



Sergey Ioffe & Christian Szegedy
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
Journal of Machine Learning Research - 2014

Regularizers

batch normalization



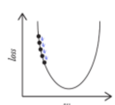
Sergey Ioffe & Christian Szegedy
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
Journal of Machine Learning Research - 2014

Optimizer-specifc Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...

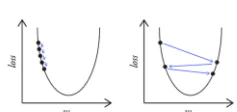
Optimizer-specifc Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...



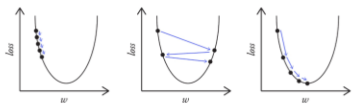
Optimizer-specifc Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...



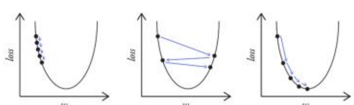
Optimizer-specifc Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...



Optimizer-specifc Hyperparameters

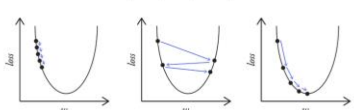
learning rate
0.1, 0.01, 0.001, 0.0001,...



batch size
16, 32, 64, 128,...

Optimizer-specifc Hyperparameters

learning rate
0.1, 0.01, 0.001, 0.0001,...



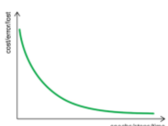
batch size
16, 32, 64, 128,...

# of epochs
early stopping

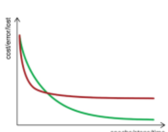
Babysitting your Network

https://lossfunctions.tumblr.com/

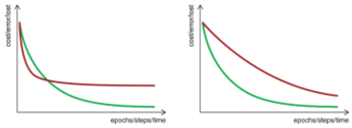
Debugging through Learning Curves



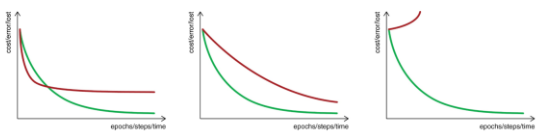
Debugging through Learning Curves



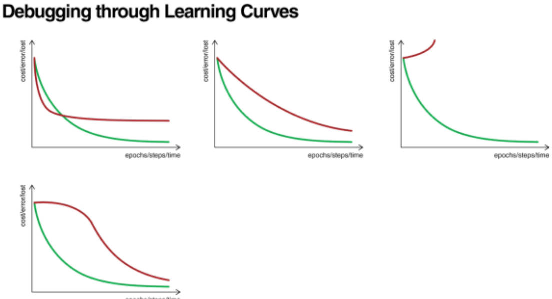
Debugging through Learning Curves



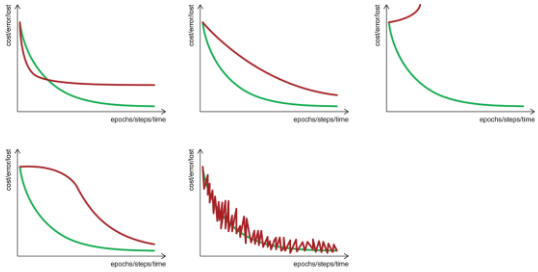
Debugging through Learning Curves



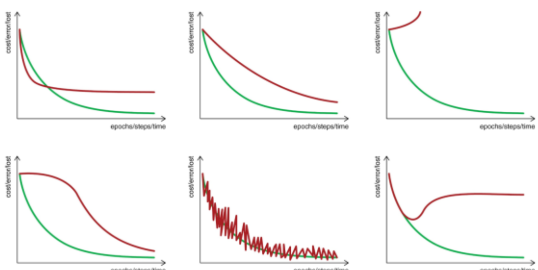
Debugging through Learning Curves



Debugging through Learning Curves



Debugging through Learning Curves



Thank you!