Deep Learning

Loo Tiang Kuan, Leonard

A0098795B

supervised by

Associate Professor Ji Hui

co-supervised by

Assistant Professor Alvina Goh Siew Wee

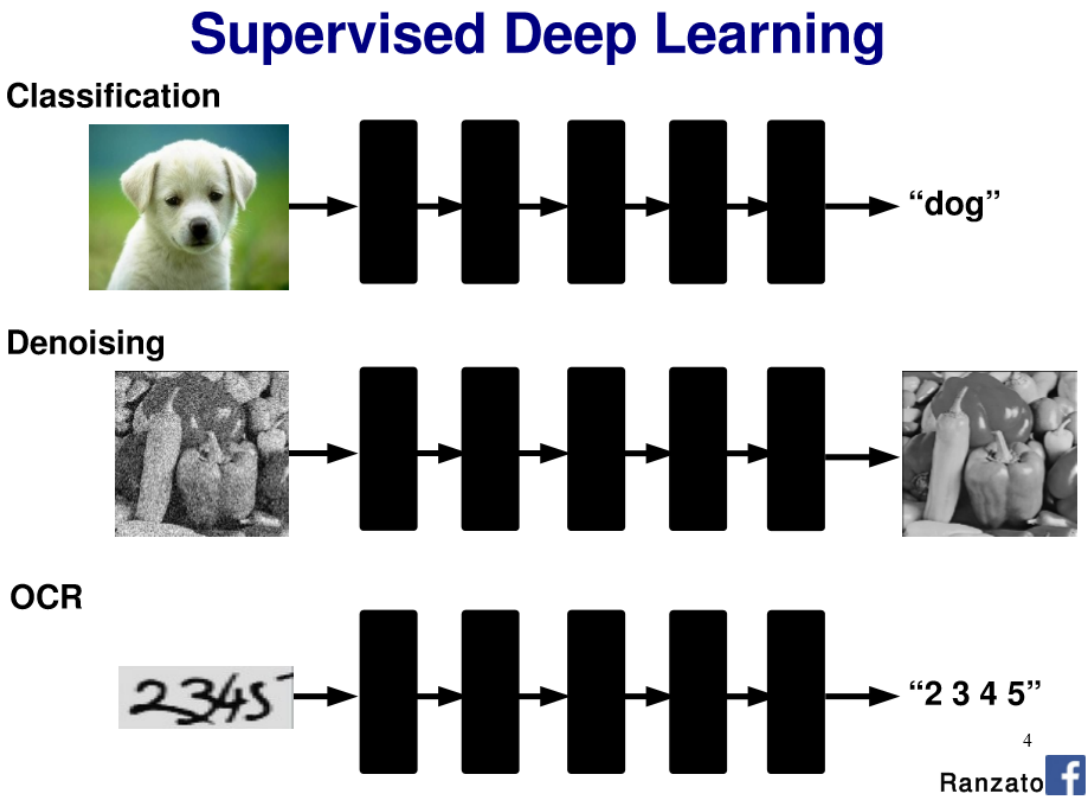
May 5, 2017

Today's Agenda

1. Introduction to Deep Learning
2. Neural Networks
3. Modifications
4. Regularization

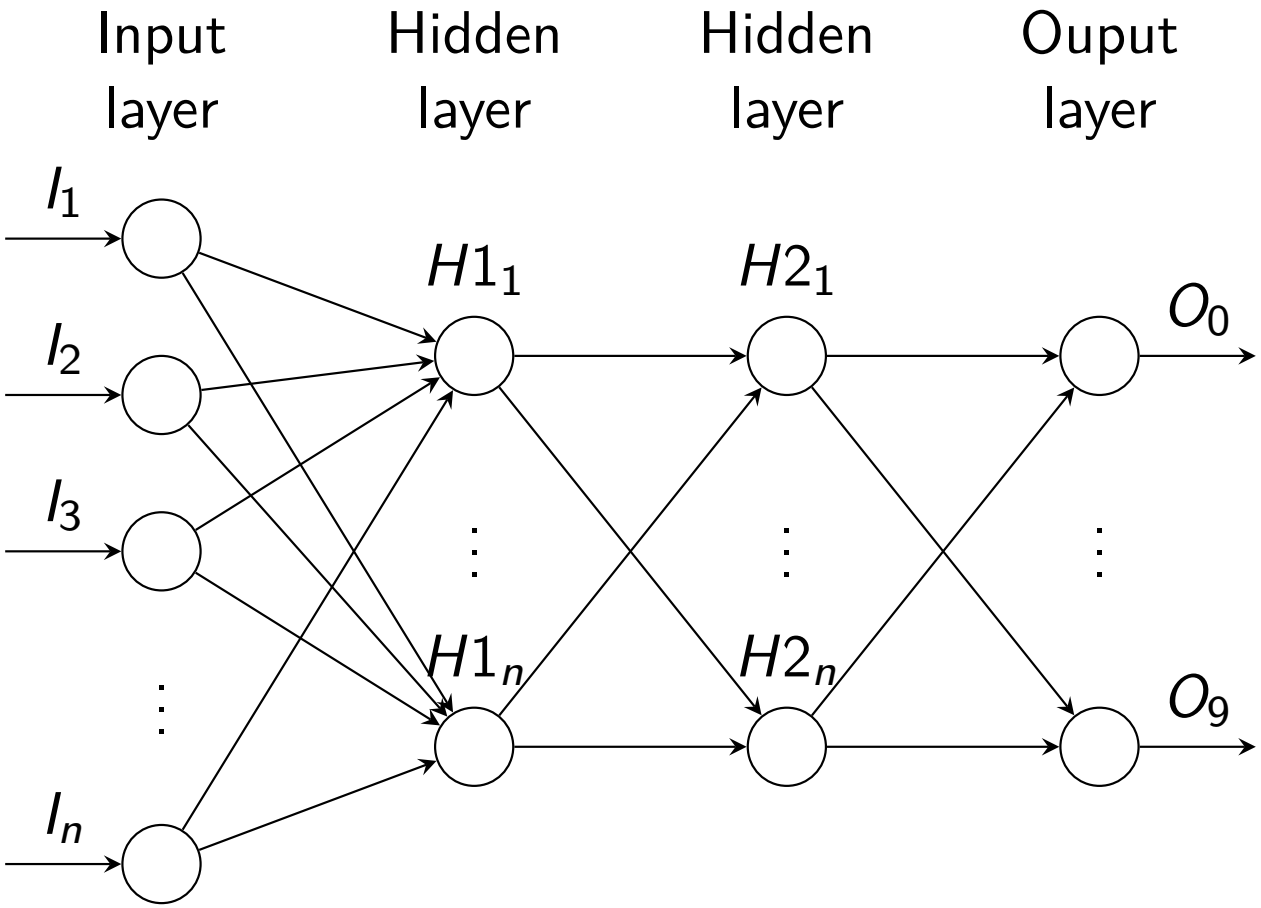
Outline

Recent developments

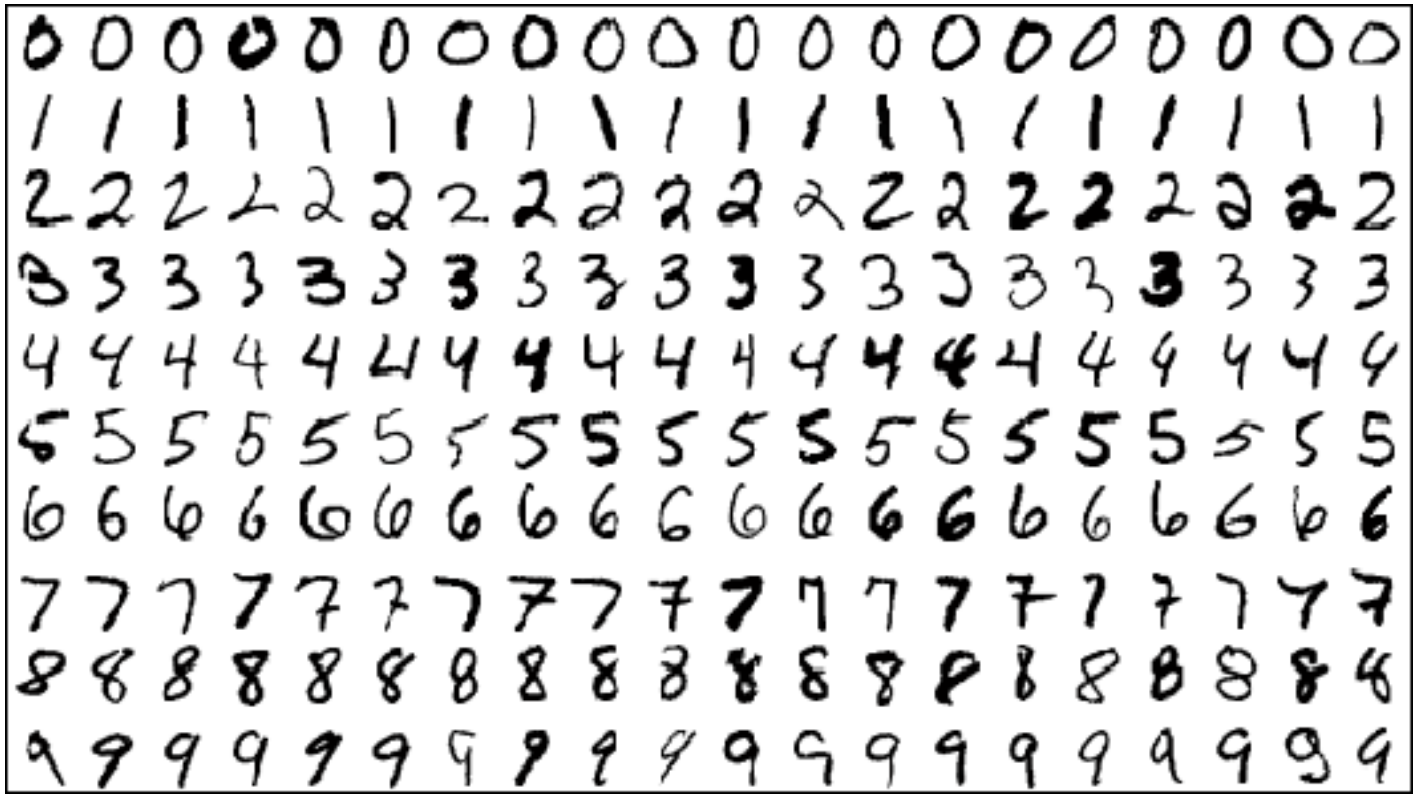


Outline

Overview


Input layer
Hidden layer
Hidden layer
Ouput layer
I_1
I_2
I_3
I_n
$H1_1$
$H1_n$
$H2_1$
$H2_n$
O_0
O_9

MNIST Data



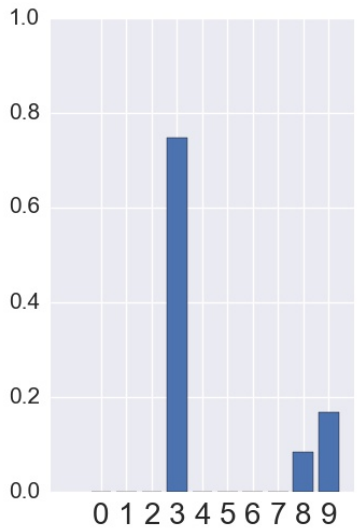
MNIST Handwritten Database

50000 Training Data

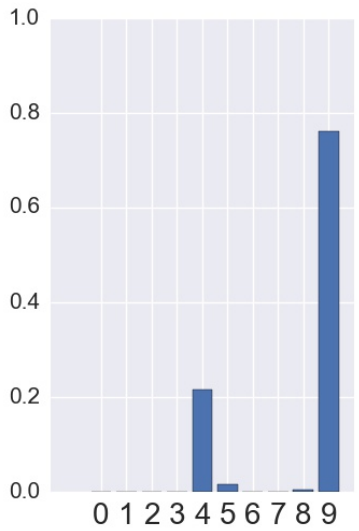
10000 Validation Data

10000 Test Data

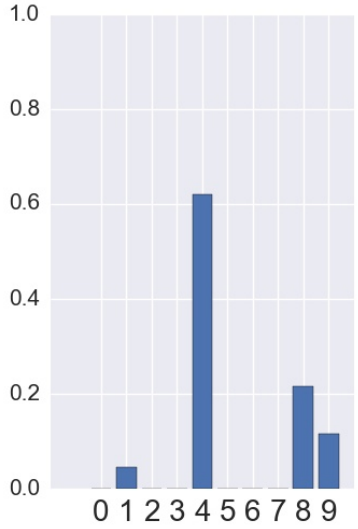
Overview



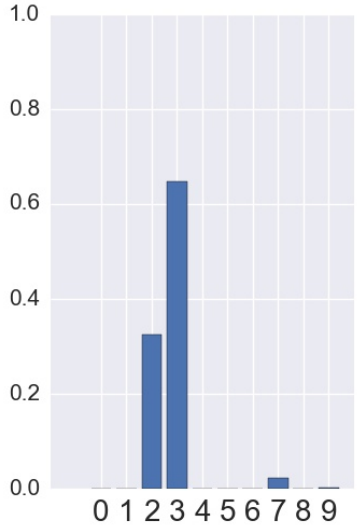
True Label: 3



True Label: 4



True Label: 9



True Label: 2

Overview

Architecture	**Activation**	**Cost**	**Algorithm**
Dense	Sigmoid	Quadratic	Gradient Descent (GD)
Convolutional	Softmax	Cross Entropy	Stochastic GD
	Tanh	Log-likelihood	Backpropagation
	ReLU		

Improvements

Regularization

Dropout

Initialisation

Overview

Session 1
Activations
Cost
Stochastic Gradient Descent
Backpropagation
Overfitting
Regularization

Session 2
Initialisation
Hyper-parameters
Variants of SGD
More Regularisers
Intro to ConvNets

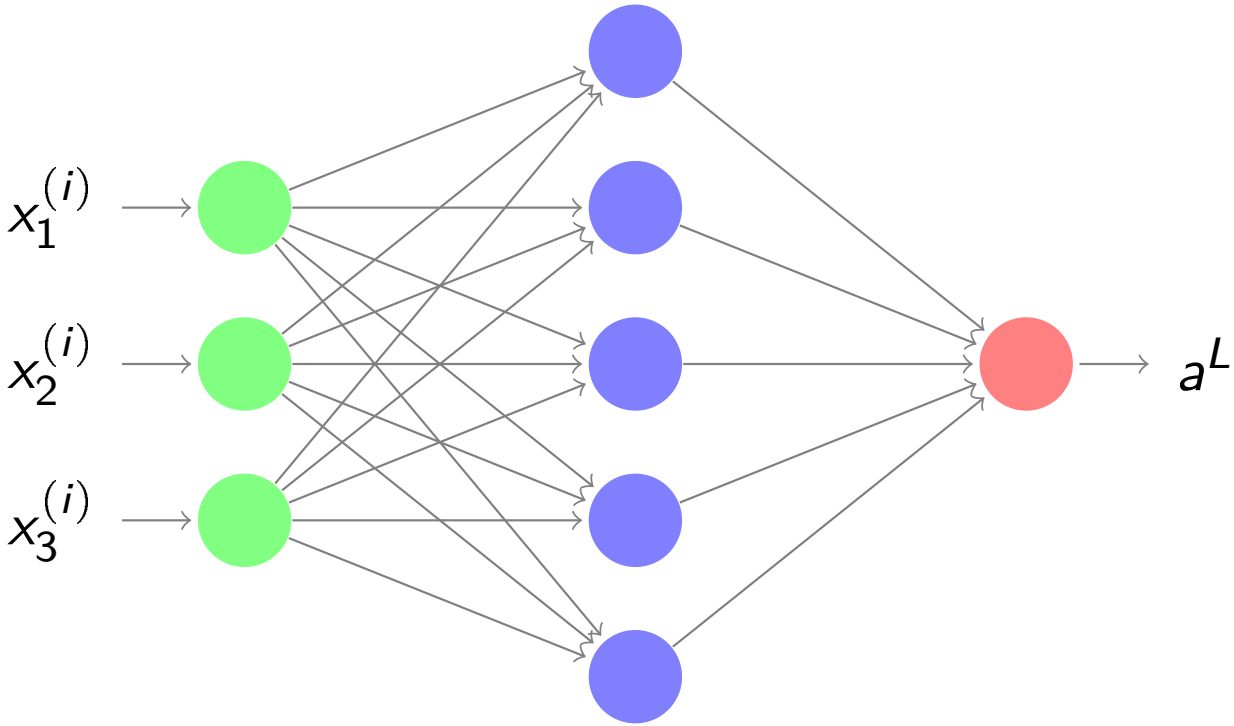
Session 3
Popular ConvNet Architectures
AlexNet
ZFNet
VGG
GoogLeNet
ResNet
Denoising

Session 4/5
Autoencoders
Transposed Convolution
Generative Network
Generative Adversarial Network

Outline

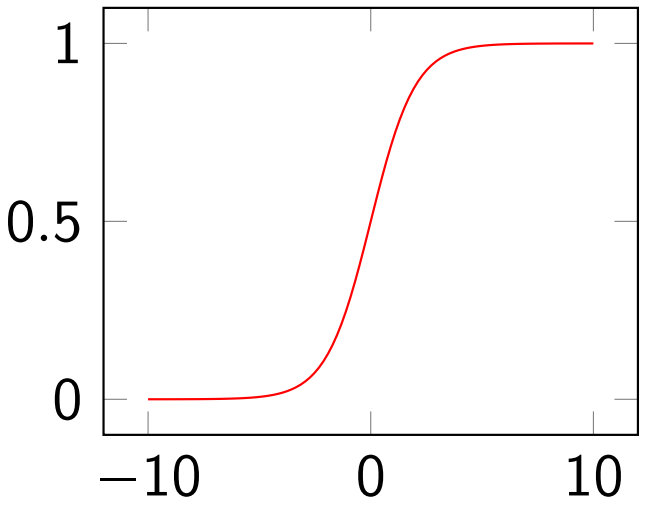
Outline

Perceptron



$$a^l = \begin{cases} 1, & W^l a^{l-1} + b^l \geqslant 0 \\ 0, & W^l a^{l-1} + b^l < 0 \end{cases}$$

Sigmoid

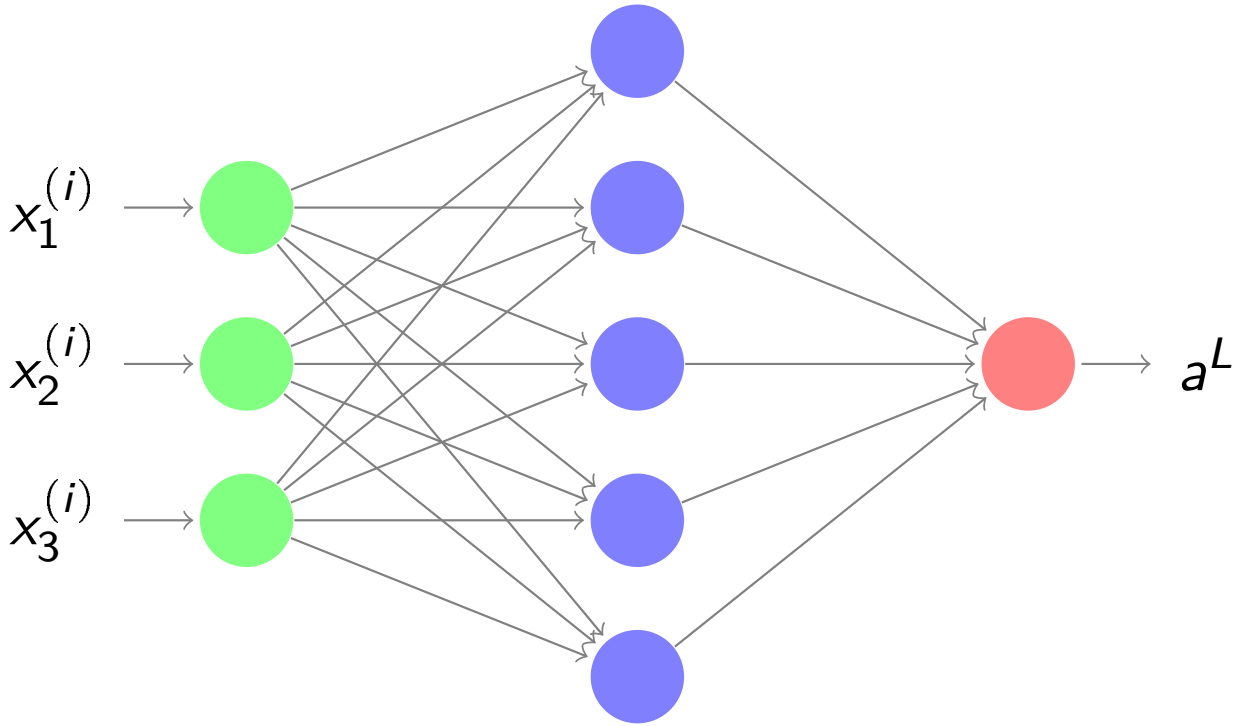


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$a^l = \sigma(z^l) \approx \begin{cases} 1, & z \to \infty \\ 0, & z \to -\infty \end{cases}$$

Outline

Quadratic Cost



$$C(\mathbf{x};\Theta) = C(\mathbf{x},\mathbf{w},\mathbf{b}) = \frac{1}{2n}\sum_{i=1}^{n}\left\|y(\mathbf{x}^{(i)}) - a^L(\mathbf{x}^{(i)})\right\|_2^2,$$

$$\text{where } \mathbf{x} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(n)}\right\}$$

Outline

Outline

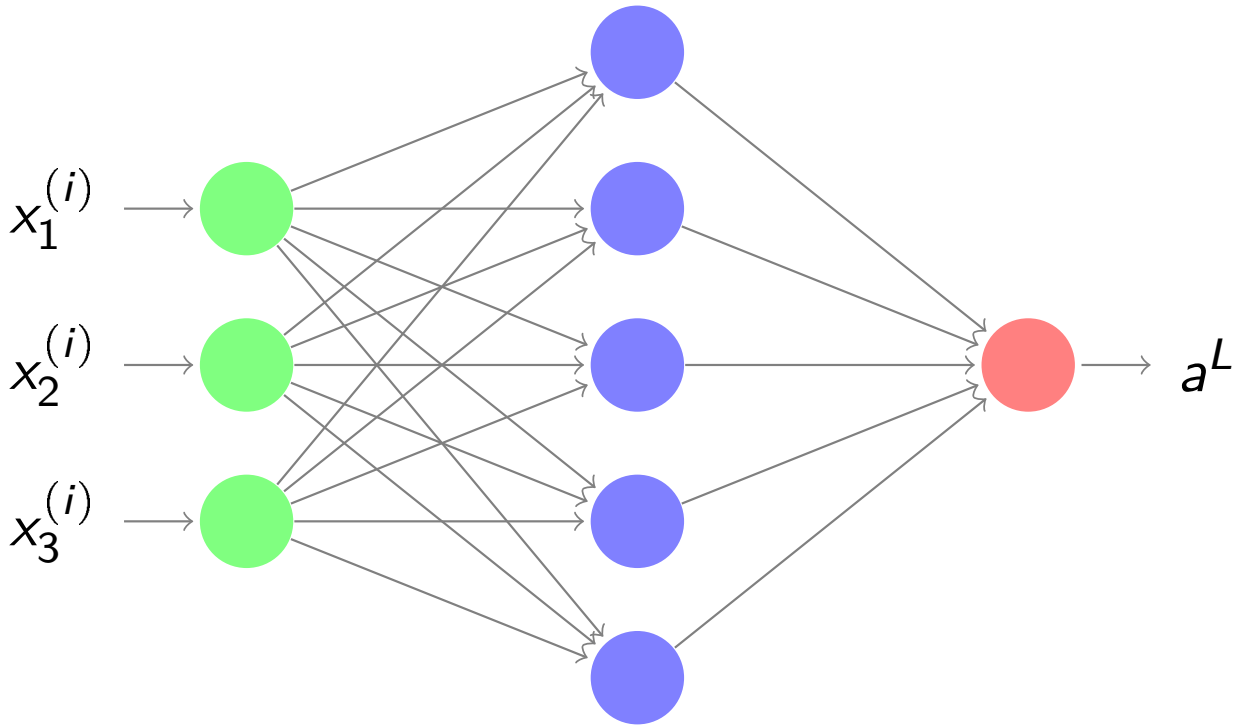
Gradient descent

For each $\mathbf{x}^{(i)}$,

$$
\begin{aligned}
C(\mathbf{x}^{(i)}, w, b) &= \frac{1}{2}\left[y(\mathbf{x}^{(i)}) - a^2(\mathbf{x}^{(i)})\right]^2 \\
\frac{\partial C}{\partial w^{(2)}} &= \left(\frac{\partial C}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}\right)\frac{\partial z^{(2)}}{\partial w^{(2)}} \\
\frac{\partial C}{\partial b^{(2)}} &= \left(\frac{\partial C}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}\right)\frac{\partial z^{(2)}}{\partial b^{(2)}} \qquad (1) \\
\frac{\partial C}{\partial w^{(1)}} &= \left(\frac{\partial C}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}\right)\left(\frac{\partial z^{(2)}}{\partial a^{(1)}}\frac{\partial a^{(1)}}{\partial z^{(1)}}\right)\frac{\partial z^{(1)}}{\partial w^{(1)}} \\
\frac{\partial C}{\partial b^{(1)}} &= \left(\frac{\partial C}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial z^{(2)}}\right)\left(\frac{\partial z^{(2)}}{\partial a^{(1)}}\frac{\partial a^{(1)}}{\partial z^{(1)}}\right)\frac{\partial z^{(1)}}{\partial b^{(1)}}
\end{aligned}
$$

Backpropagation (3).

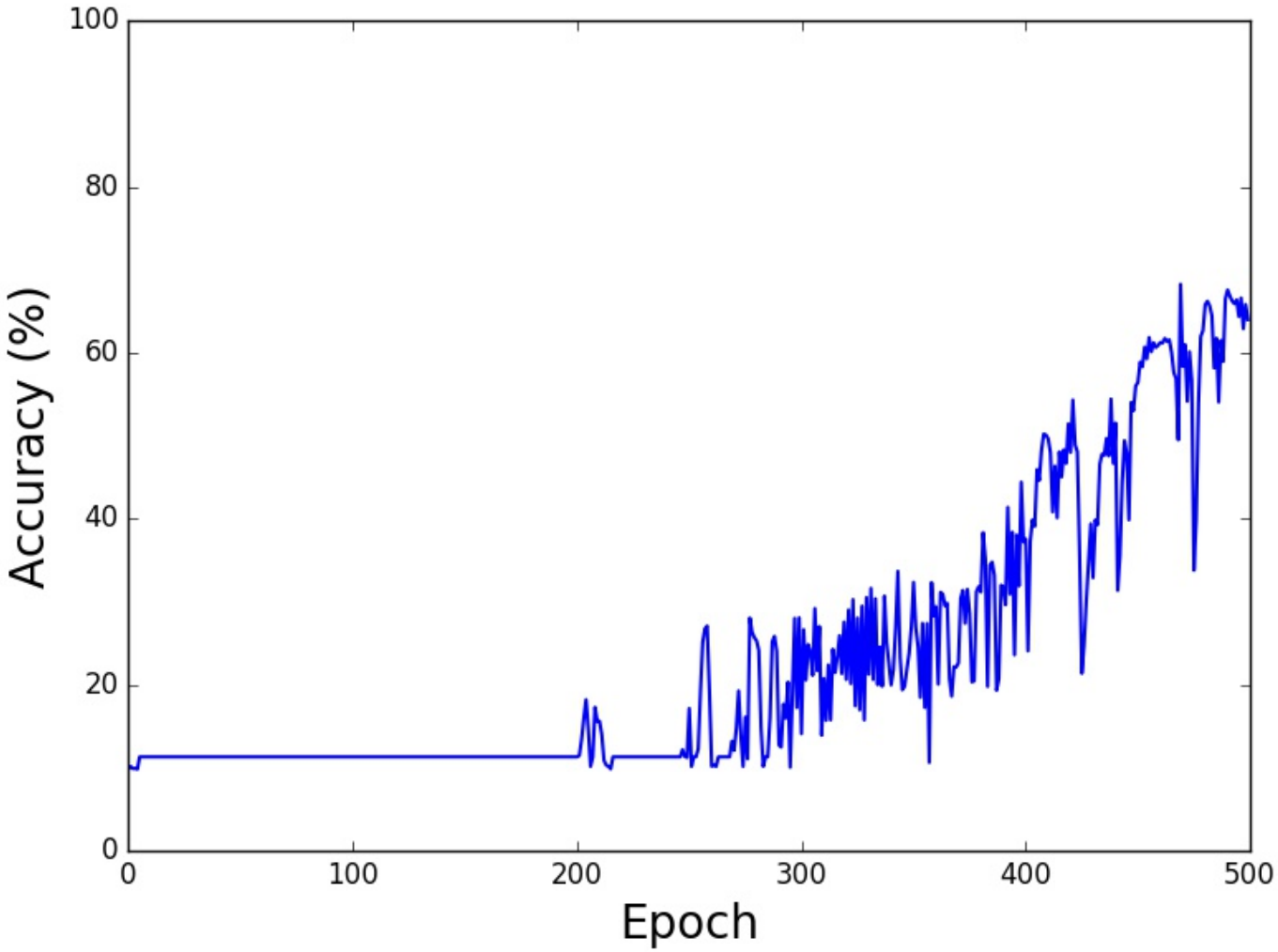
Gradient descent



$$C = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}|y - a^L|^2 \qquad w^{(k+1)} = w^{(k)} - \eta\nabla_{w^{(k)}}C(\mathbf{x})$$

$$\nabla C(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\nabla C(\mathbf{x}^{(i)}) \qquad b^{(k+1)} = b^{(k)} - \eta\nabla_{b^{(k)}}C(\mathbf{x})$$

Gradient Descent



Slow Learning!

Outline

Stochastic Gradient Descent

Randomly sample into m mini-batches of size k:

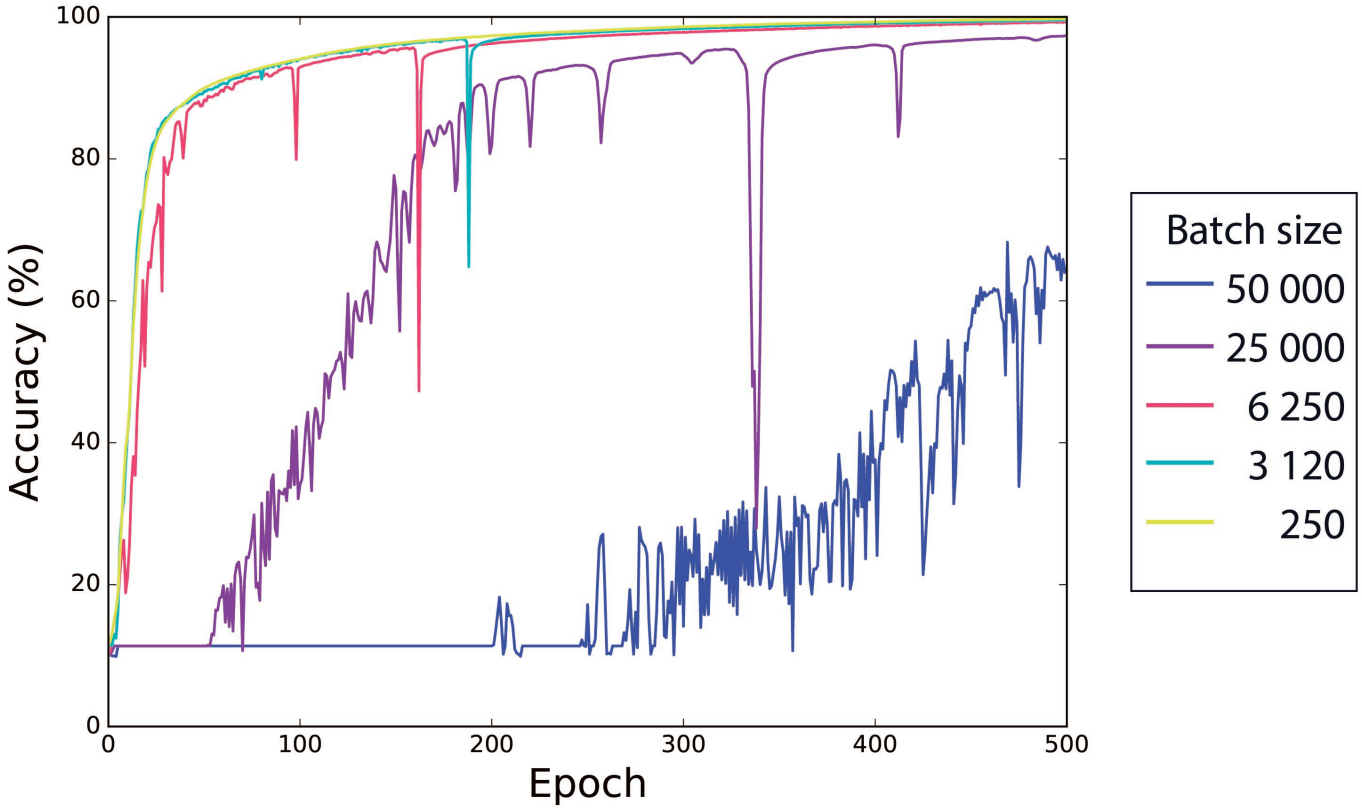
$$\begin{aligned}\mathbf{x} &= \left\{\{x^{(1)}, x^{(2)}, ..., x^{(k)}\}, \{x^{(k+1)}, x^{(k+2)}, ..., x^{(2k)}\}, ..., \{x^{(n-k+1)}, ..., x^{(n)}\}\right\} \\ &= \{M_1, M_2, ..., M_m\}\end{aligned}$$

$$\nabla C(\mathbf{x}) \approx \frac{1}{m}\sum_{i=1}^{m} \nabla C(\mathbf{x}_{M_j}) \text{ , for each mini-batch } M_j$$

Parameters update for each mini-batch M_i:

$$\begin{aligned} w &\leftarrow w - \frac{\eta}{m}\frac{\partial C}{\partial w} \\ b &\leftarrow b - \frac{\eta}{m}\frac{\partial C}{\partial b} \end{aligned} \tag{2}$$

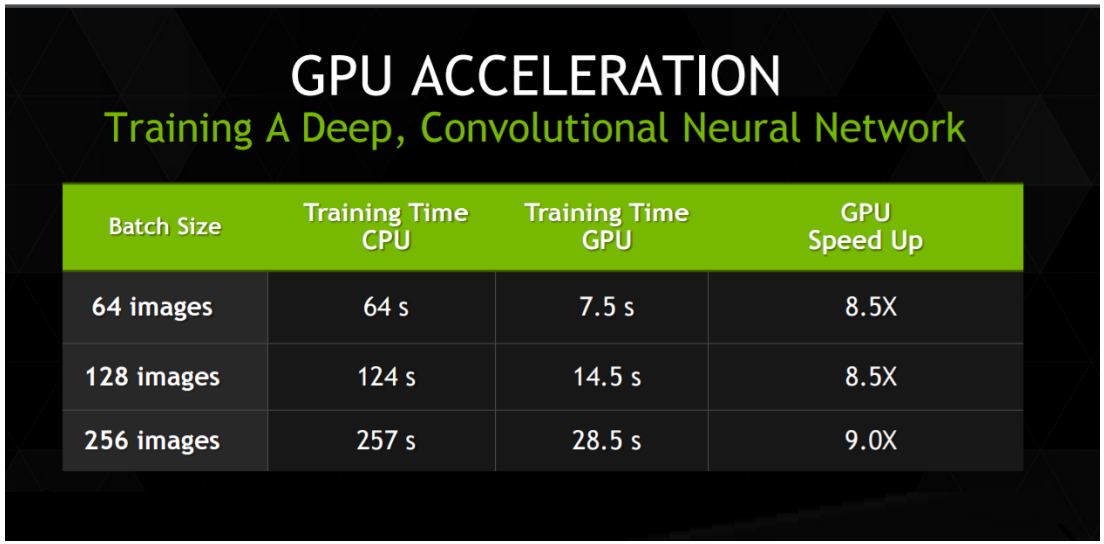
Stochastic Gradient Descent



1 Epoch - All $\{M_1, M_2, ...\}$ are used.

Learning accelerates with more minibatches(smaller minibatch size).

Stochastic Gradient Descent



http://www.nvidia.com
Deep Learning For Image Classification (Slide 28)

Remark:

- $w^{(k+1)} = w^{(k)} - \frac{\eta}{m} \frac{\partial C}{\partial w^{(k)}}$,
  need to adjust the learning rate η accordingly.
- Batch-size vs GPU computation

Outline

Backpropagation

Recall the gradient descent in Eqn (1).

Definition (Error term δ_j^l)

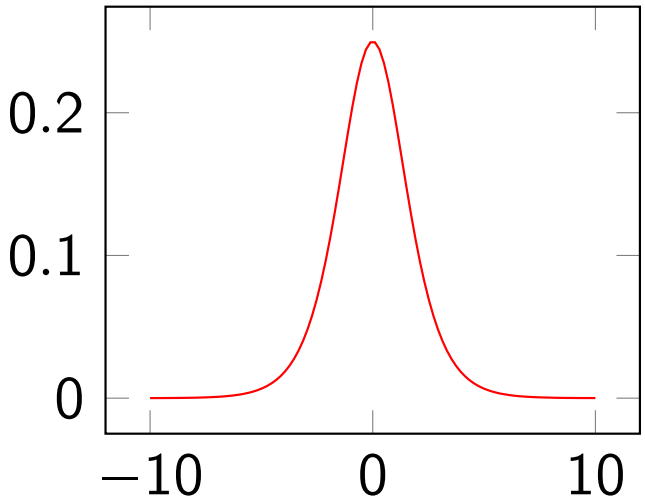
$$\delta_j^l = \frac{\partial C}{\partial z_j^l}.$$

We have the four following Backpropagation equations

$$\begin{aligned}
\delta_j^l &= \frac{\partial C}{\partial a_j^l} g'(z_j^l). \\
\delta^l &= \left([W^{l+1}]^T \delta^{l+1} \right) \odot g'(z^l). \\
\frac{\partial C}{\partial b_i^l} &= \delta_i^l. \\
\frac{\partial C}{\partial w_{ij}^l} &= a_j^{l-1} \delta_i^l.
\end{aligned} \tag{3}$$

Outline

Saturation



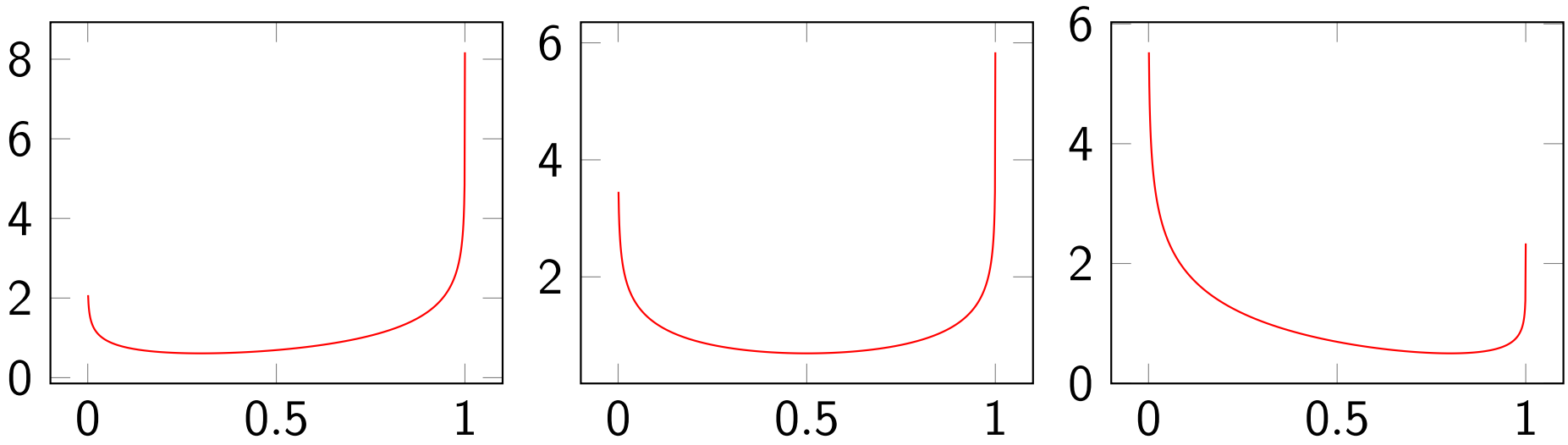
$$C(\mathbf{x}, \mathbf{w}, \mathbf{b}) = \frac{1}{2n} \left\| \mathbf{y} - \mathbf{a}^L \right\|_2^2$$

$$\frac{\partial C}{\partial w^L} = a^{L-1}(a^L - y)\sigma'(z) \to 0$$

$$\frac{\partial C}{\partial b^L} = (a^L - y)\sigma'(z) \to 0$$

Outline

Cross Entropy Cost



$$C(\mathbf{x}, \mathbf{w}, \mathbf{b}) = -\left[\left(y \log a^L\right) + (1-y) \log\left(1 - a^L\right)\right]$$

$$\nabla C(\mathbf{x}, \mathbf{w}, \mathbf{b}) = -\left[\frac{y}{a^L} - \frac{1-y}{1-a^L}\right]$$

$$\frac{\partial C}{\partial w^L} = a^{L-1}(a^L - y)$$

$$\frac{\partial C}{\partial b^L} = a^L - y$$

Other Activations and Cost Functions

Softmax function:

$$\text{softmax}(z^l) = \frac{e^{z^l_j}}{\sum_k e^{z^l_k}}$$

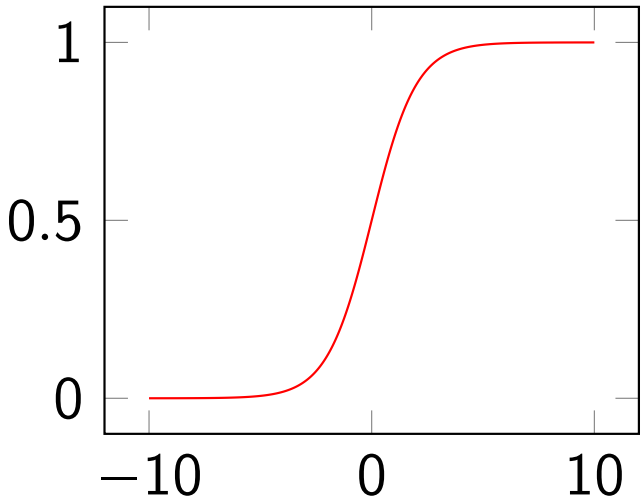
Log-likelihood Cost:

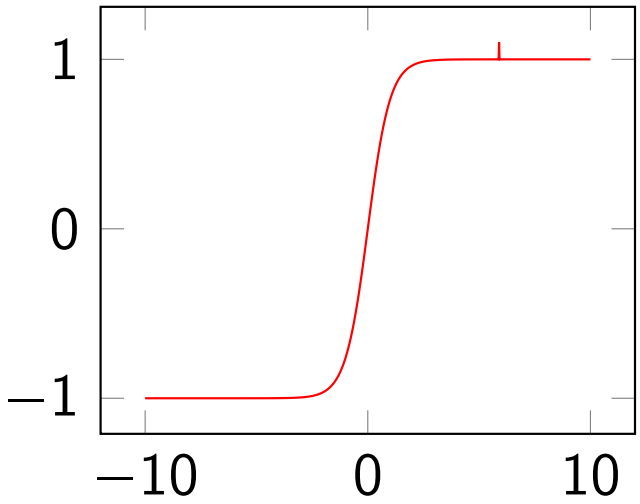
$$C = -\frac{1}{n}\sum_x \sum_k y_k \log a^L_k(x), \text{ for k classes}$$

$$\frac{\partial C}{\partial w^L} = a^{L-1}(a^L - y)$$
$$\frac{\partial C}{\partial b^L} = a^L - y$$

Outline

tanh





$$\sigma(z) = \frac{1+\tanh(\frac{z}{2})}{2}$$

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

Pros

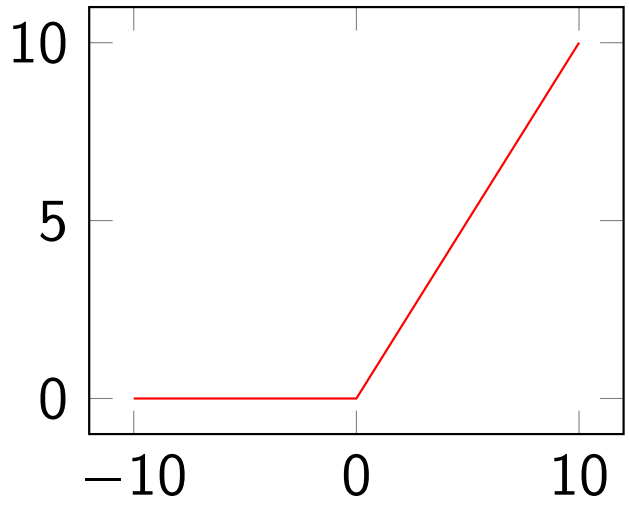
$\tanh(z) \in (-1, 1)$

weights can increase/decrease

Cons

Possibly saturate greater

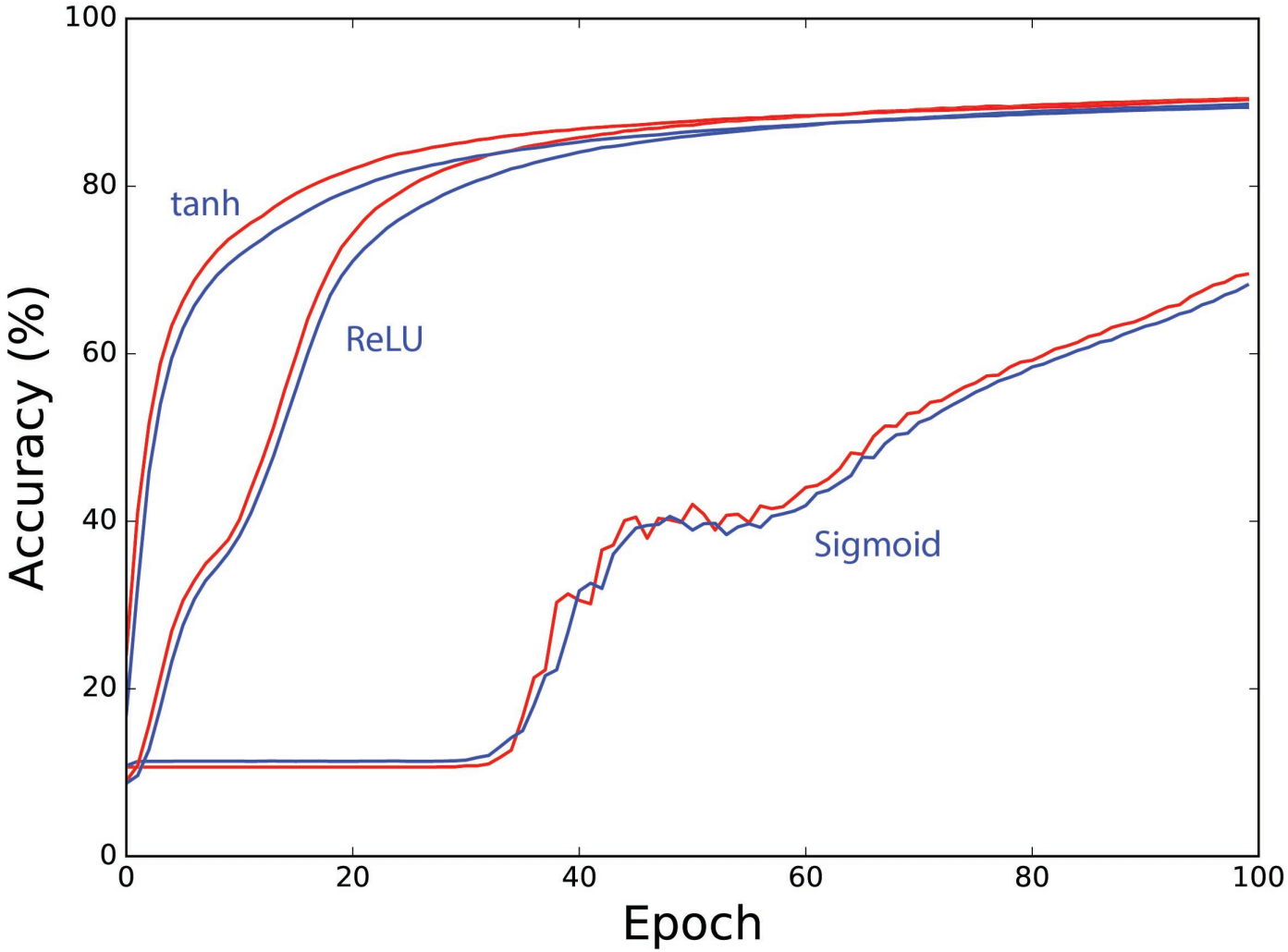
ReLU



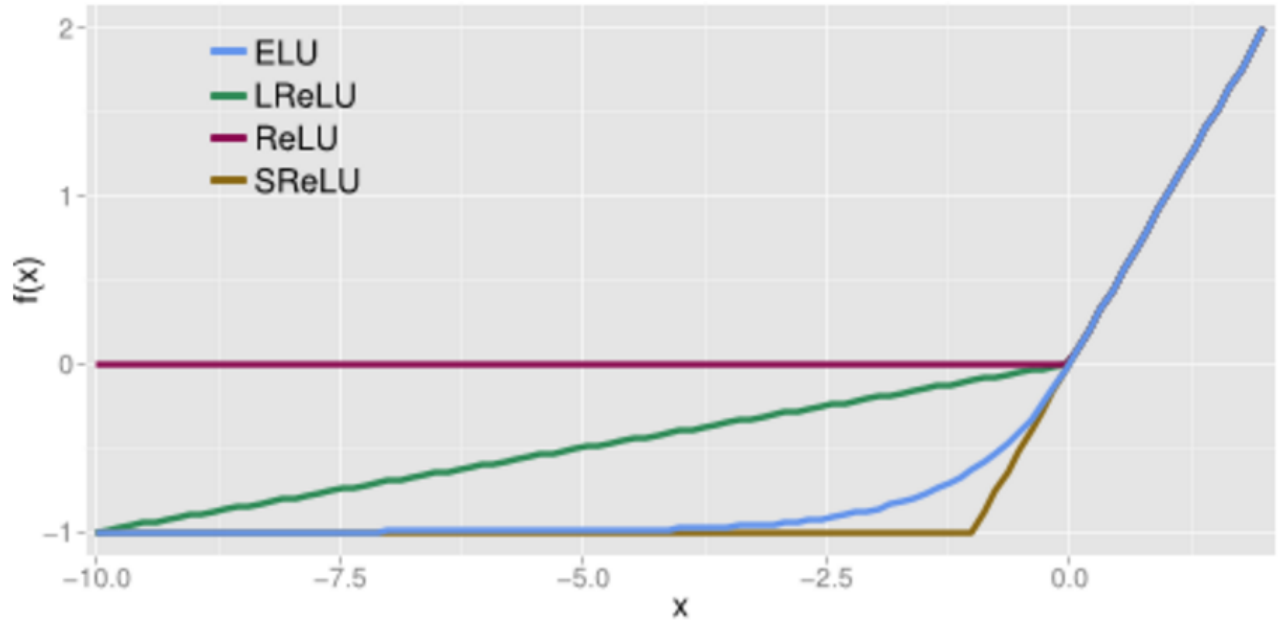
$$\text{ReLU}(z) = \max\{0, z\}$$

Pros	Cons
Does not saturate at large z	gradient and value is 0 at negative z
Perform well in image processing	
More computationally efficient	

Comparison


100
80
60
40
20
0
Accuracy (%)
tanh
ReLU
Sigmoid
0
20
40
60
80
100
Epoch

Even More Activations



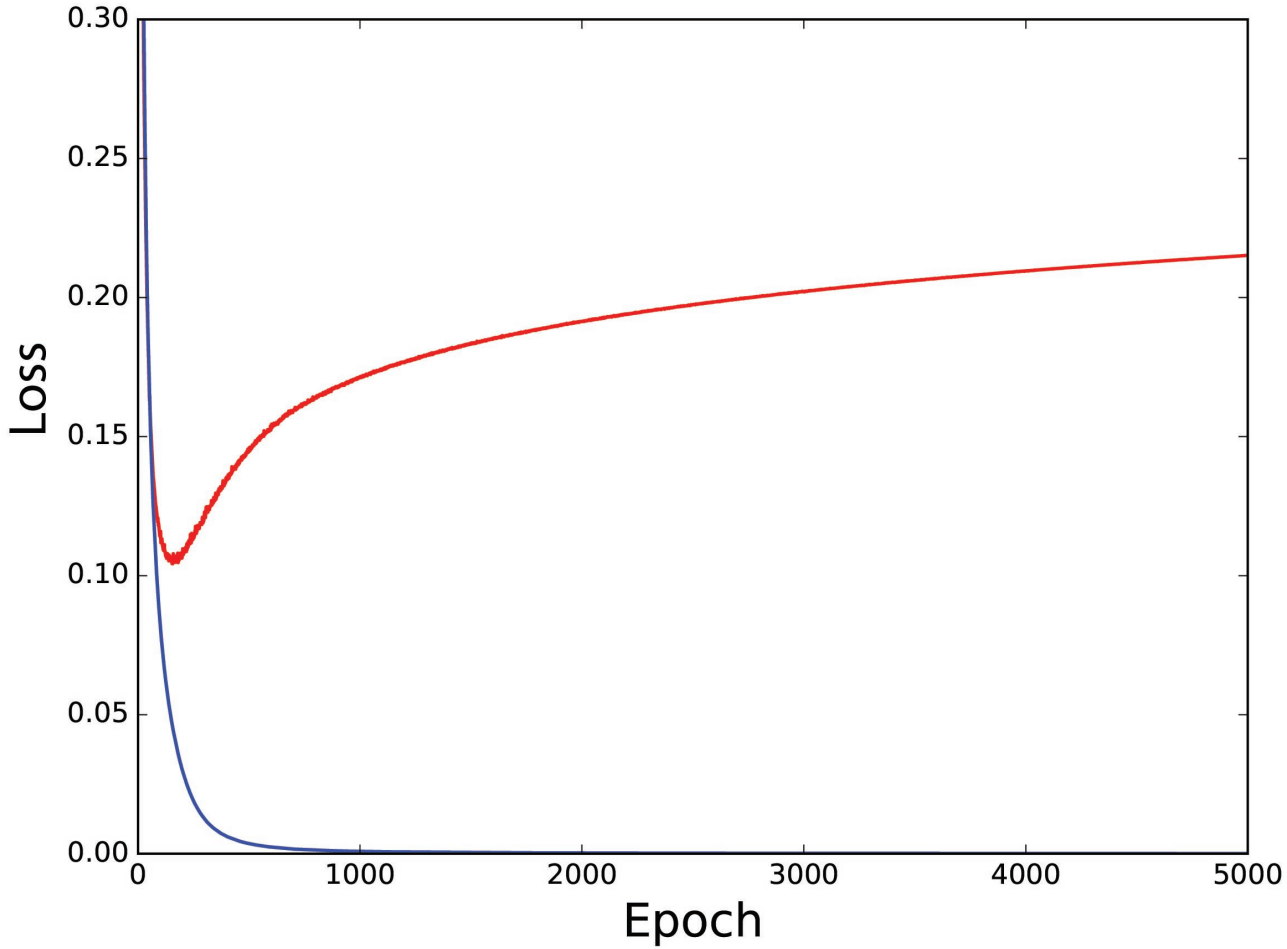
$$\text{LReLu}(z) = \max(\alpha z, z)$$

$$\text{ELU}(z) = \max(\alpha(e^z - 1), z)$$

Outline

Overfitting

- Too many parameters $\Longrightarrow$ Overfitting
- Regularize Weights



Outline

Outline

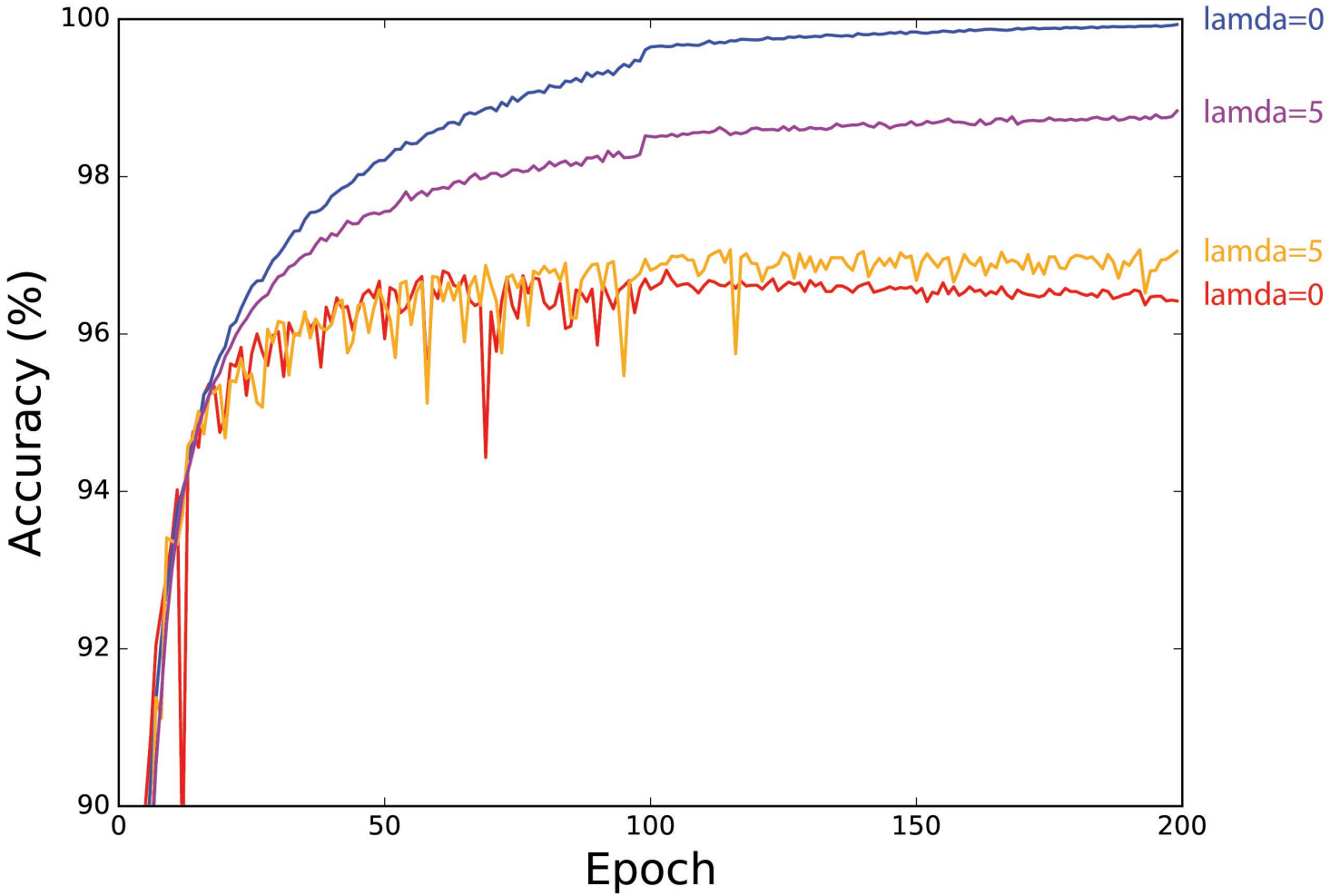
L2 Regularization

$$C = C_0 + \frac{1}{n}\frac{\lambda}{2}||w||_2^2 \tag{4}$$

$$\begin{aligned} \frac{\partial C}{\partial w} &= \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} w \\ \frac{\partial C}{\partial b} &= \frac{\partial C_0}{\partial b} \end{aligned} \tag{5}$$

$$\begin{aligned} w_{k+1} &= \left(1 - \frac{\eta\lambda}{n}\right) w_k - \frac{\eta}{m}\frac{\partial C_0}{\partial w} \\ b_{k+1} &= b_k - \frac{\eta}{m}\frac{\partial C_0}{\partial b} \end{aligned} \tag{6}$$

L2 Regularization



Outline

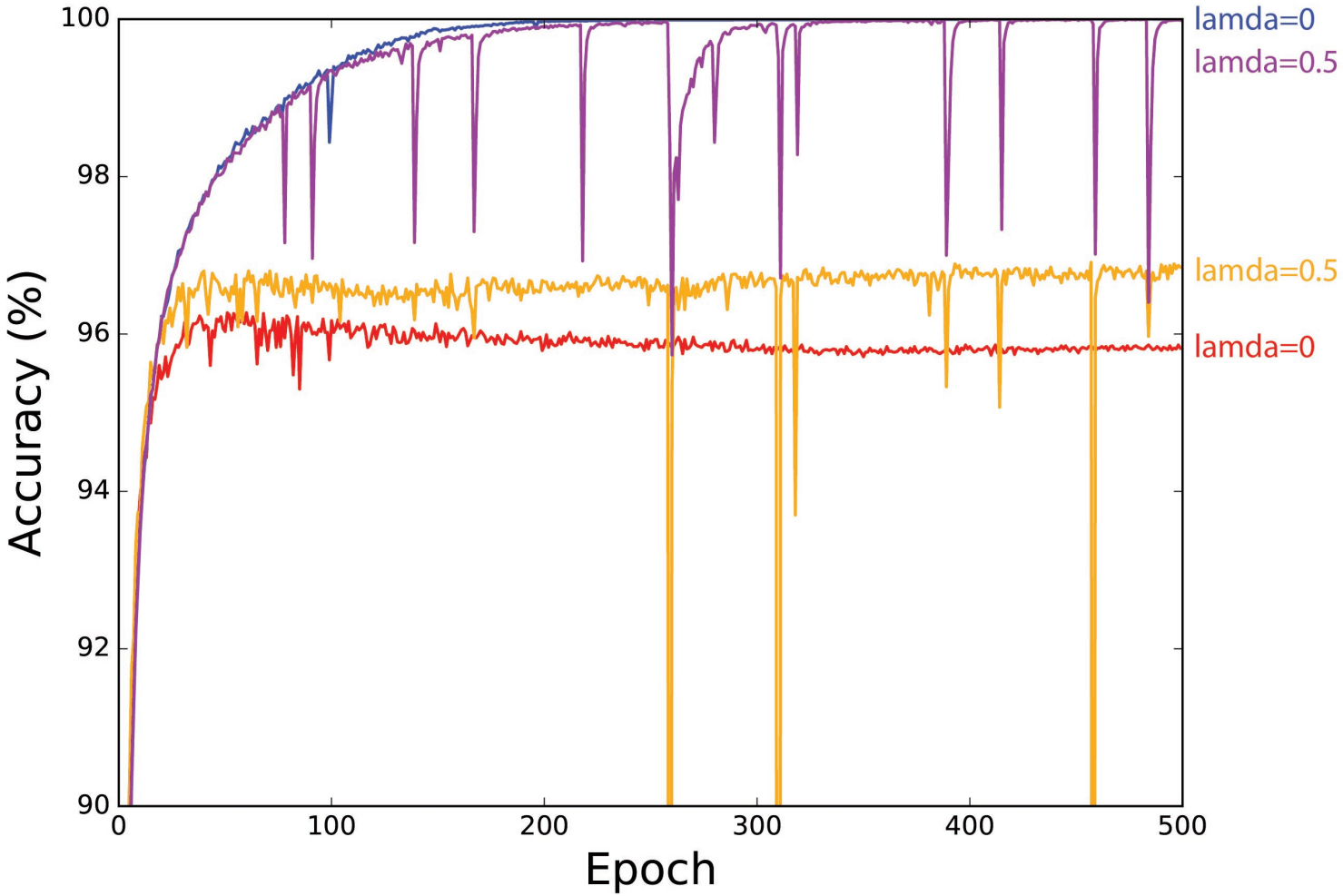
L1 Regularization

$$C = C_0 + \frac{\lambda}{n}||w||_1 \tag{7}$$

$$\begin{aligned} \frac{\partial C}{\partial w} &= \frac{\partial C_0}{\partial w} + \frac{\lambda}{n}\text{sgn}(w) \\ \frac{\partial C}{\partial b} &= \frac{\partial C_0}{\partial b} \end{aligned} \tag{8}$$

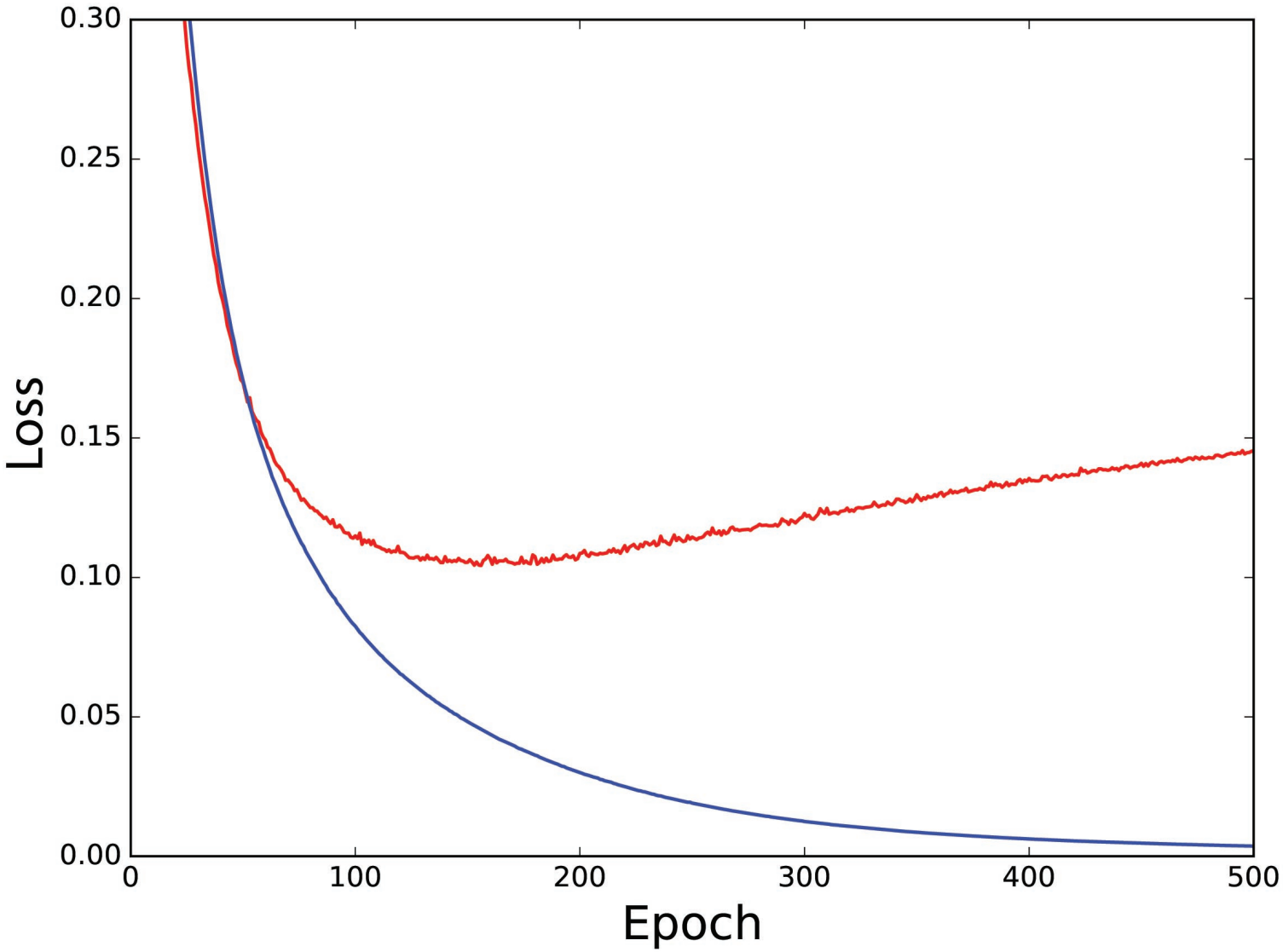
$$\begin{aligned} w_{k+1} &= \left(w_k - \frac{\eta\lambda}{n}\text{sgn}(w_k)\right) - \frac{\eta}{m}\frac{\partial C_0}{\partial w} \\ b_{k+1} &= b_k - \frac{\eta}{m}\frac{\partial C_0}{\partial b} \end{aligned} \tag{9}$$

L1 Regularization



Outline

Early Stopping



Thank You

References


www.cs.toronto.edu/∼ranzato


neuralnetworksanddeeplearning.com/


Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). *"Fast and accurate deep network learning by exponential linear units (ELUs)."* arXiv preprint arXiv:1511.07289.


Nair, V. and Hinton, G.E., 2010. *"Rectified linear units improve restricted boltzmann machines."*In Proceedings of the 27th International Conference on Machine Learning (ICML-10) (pp. 807-814).


Glorot, X., Bordes, A. and Bengio, Y., 2011, April. *"Deep Sparse Rectifier Neural Networks."*In Aistats (Vol. 15, No. 106, p. 275).

References

LeCun, Y.A., Bottou, L., Orr, G.B. and Müller, K.R., 2012. *"Efficient backprop. "*In Neural networks: Tricks of the trade (pp. 9-48). Springer Berlin Heidelberg.

Bengio, Y., 2009. *"Learning deep architectures for AI. "*Foundations and trends® in Machine Learning, 2(1), pp.1-127.