Advanced Text Analysis for Business (IDS-566)

Lecture 5

Feb 23, 2018

Course Overview

- Instructor
  - Ehsan M. Ardehaly PhD, ehsan@uic.edu
  - Office hours: 4:45 - 5:45 pm F, BLC L270
  - Teacher assistant: 4:00 - 5:00 pm W, BLC L270
- Objectives:
  - Text mining
  - Applications for business decisions
  - Study of machine learning concepts
  - Design and implementation of text mining approaches

Assignments-2

- Grade: 20%
- Sentiment analysis
- Due date: 2/25/2018

- Submission:
  - Notebook (code + analysis) → PDF
  - Word document with code as an appendix → PDF

Agenda

Artificial Neural Network

- Layers, activation, SGD

Multi-Layer Perceptron:

- Word embedding

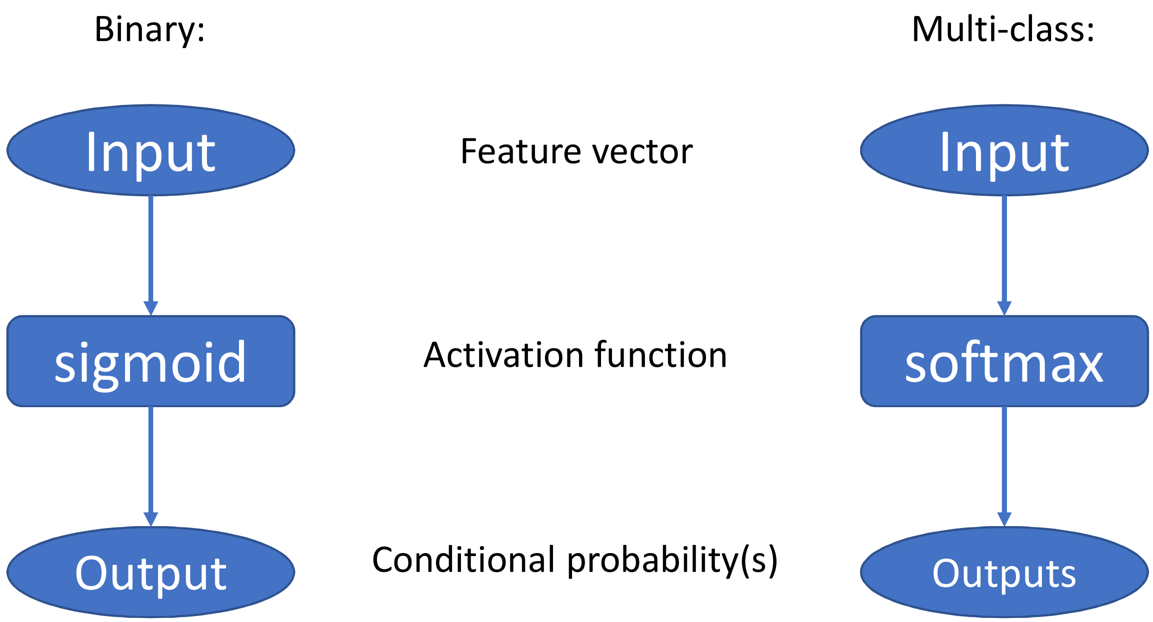
Deep learning:

- Dropout, Batch normalization

Examples:

- Keras

Binary vs. Multi-class logistic regression



Sigmoid vs. softmax

- Sigmoid (binary class):
  - $P(y = 1|x) = \frac{1}{1+e^{-x.\theta}}$

- Softmax (multi-class):
  - $P(y_i = k|x) = \frac{\exp(x.\theta^{(k)})}{\sum_{j=1}^{m} \exp(x.\theta^{(j)})}$

Training logistic regression

- Creating the cost function:
  - Negative log likelihood
  - $J(\theta) = -l(\theta)$

- Find θ which minimizes the cost function:
  - $\theta = \underset{\theta}{\operatorname{Argmin}} J(\theta)$

Binary vs multi-class cost function

- Binary:
  - Negative log likelihood → Binary cross-entropy

- Multi-class:
  - Negative log likelihood → Categorical cross-entropy

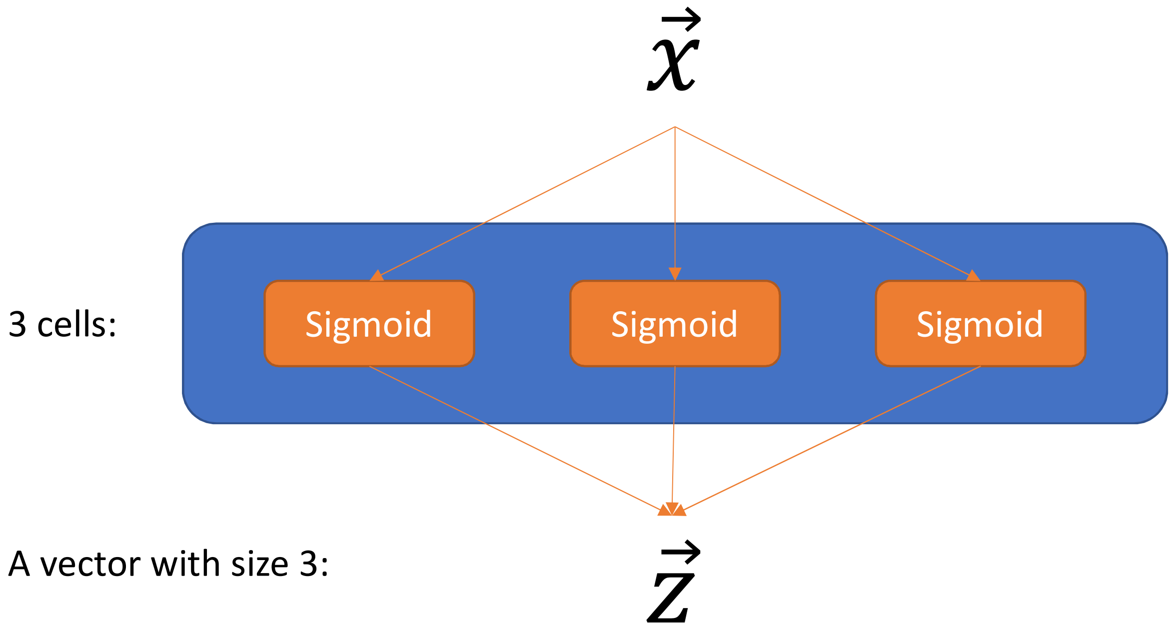
Gradient descent algorithm

- An iterative algorithm
- Finding the minimum of a cost function:
- Starts with a random initialization.
- Takes step to the negative of the gradient.

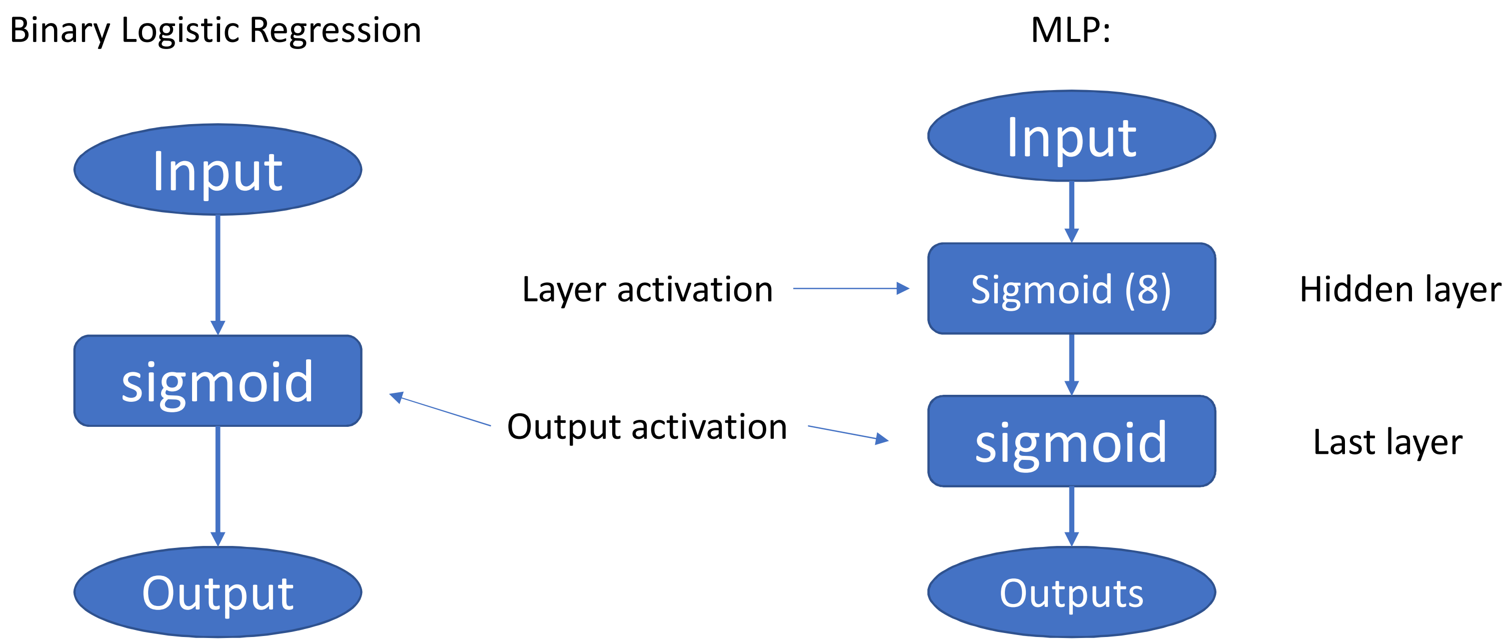
Artificial Neural Network (ANN)

- Hidden layer:
  - Input: output of last layer
  - Multiple cells
  - Apply multiple functions (per cell) → Activation function
  - Output: next layer

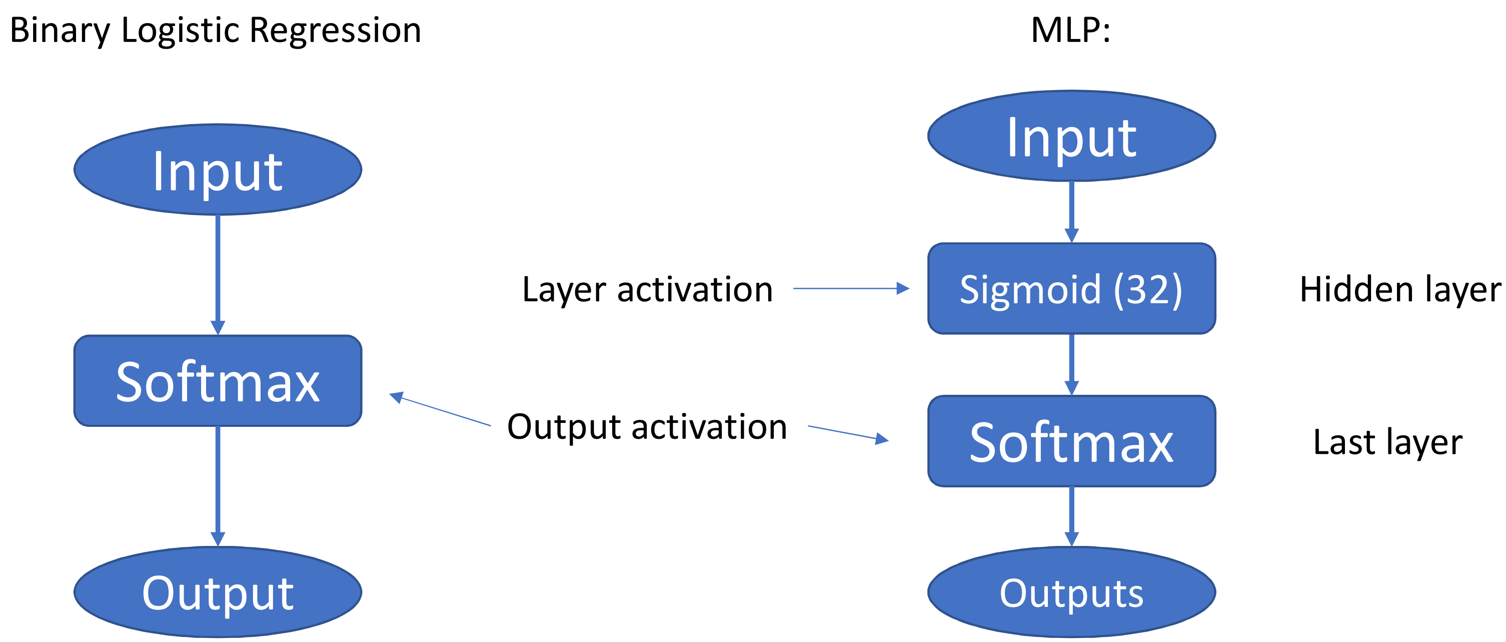
Hidden layer



2 layers MLP (binary)



2 layers MLP (multi-class)



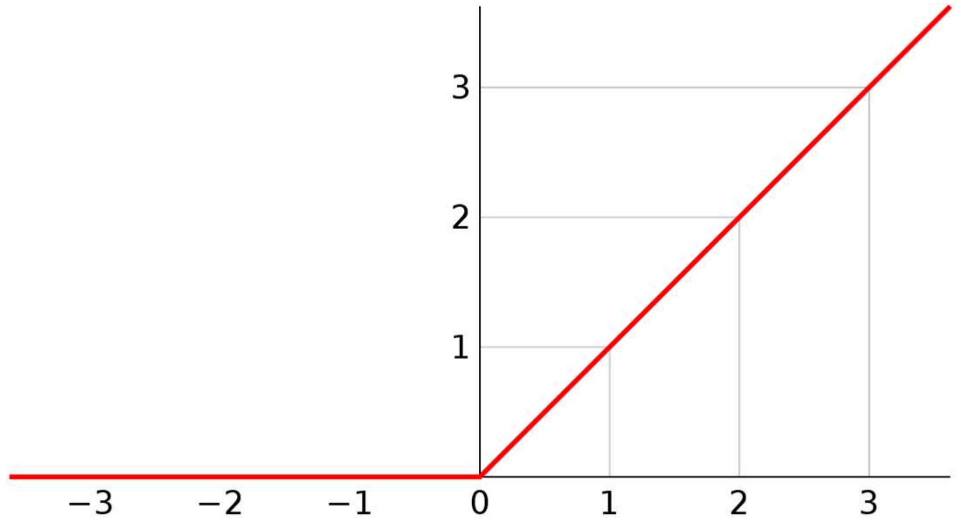
Activation functions

- Middle layers:
  - Linear (not recommended)
  - Sigmoid
  - Relu (recommended)
  - Tanh

- Output layer:
  - Sigmoid (for binary)
  - Softmax (for multi-class)

Rectifier linear unit (relu)

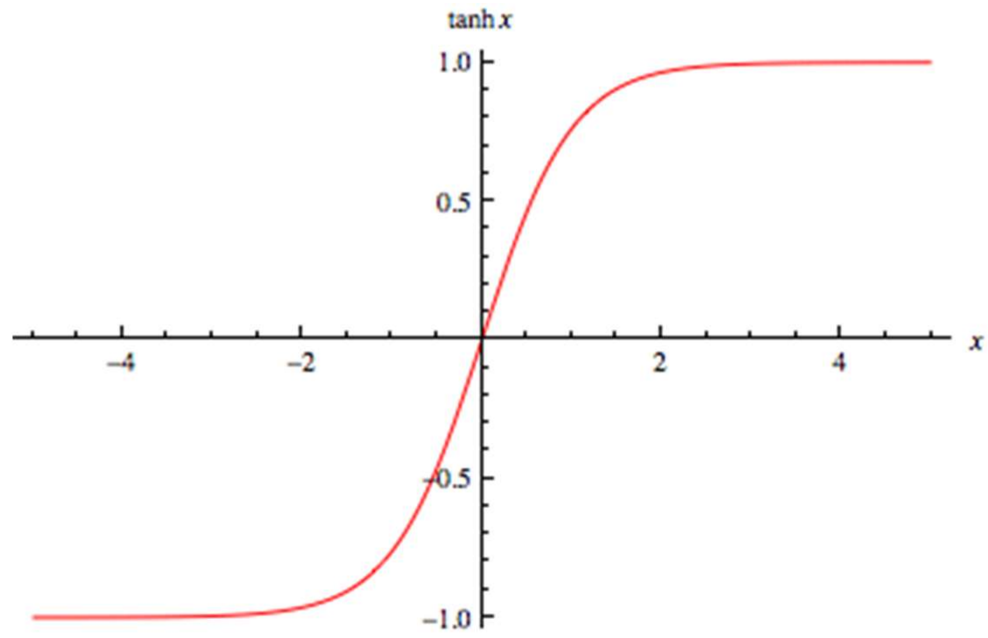
- $relu(x) = \max(0, x)$



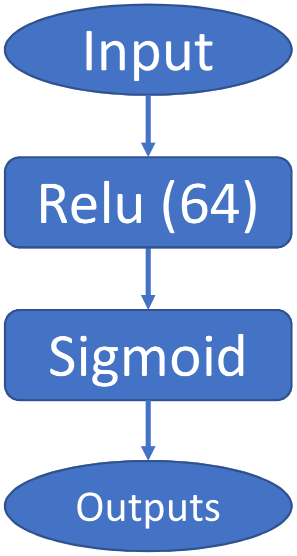
Tangent Hyperbolic

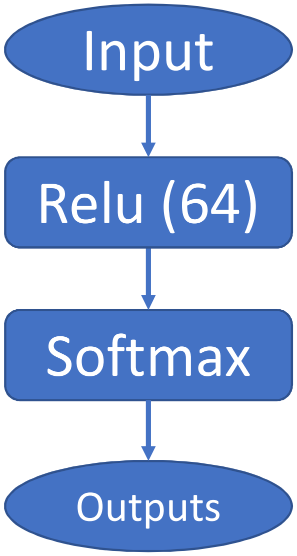
- $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

- $tanh(x) = 2\sigma(2x) - 1$

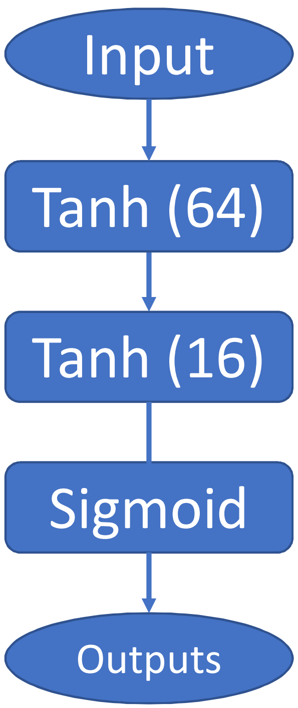


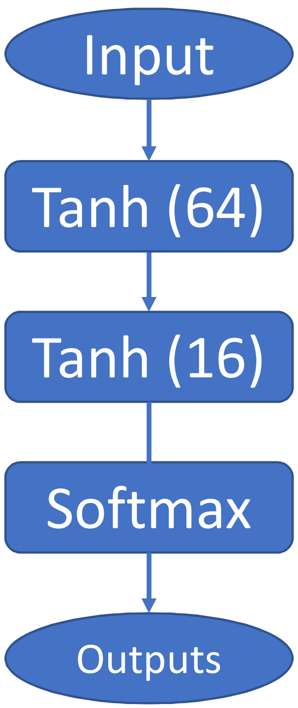
2 layer MLP (with relu)





3 layer MLP (with tanh)





Optimization

Cost function:

- Binary cross-entropy (binary)
- Categorical cross-entropy (multi-class)

Batch gradient descent:

- Stochastic Gradient Descent (SGD)

GD vs. SGD

Gradient Descent:

Updating gradients to the entire data

Good for simple models (e.g. logistic regression)

All data must be fit in the memory.

Stochastic Gradient Descent:

Dividing data to small batches

- Updating gradients for each batch.

Good for complex models (e.g. neural networks)

Only a batch need to fit in the memory.

Advanced optimizers (for batch)

- SGD with learning rate decay
- Adam
- RMSprop

Batch example - 1

- 100K instances with 2K features for 5 categories classification:
  - Shapes: X: 100000 x 2000, y: 100000

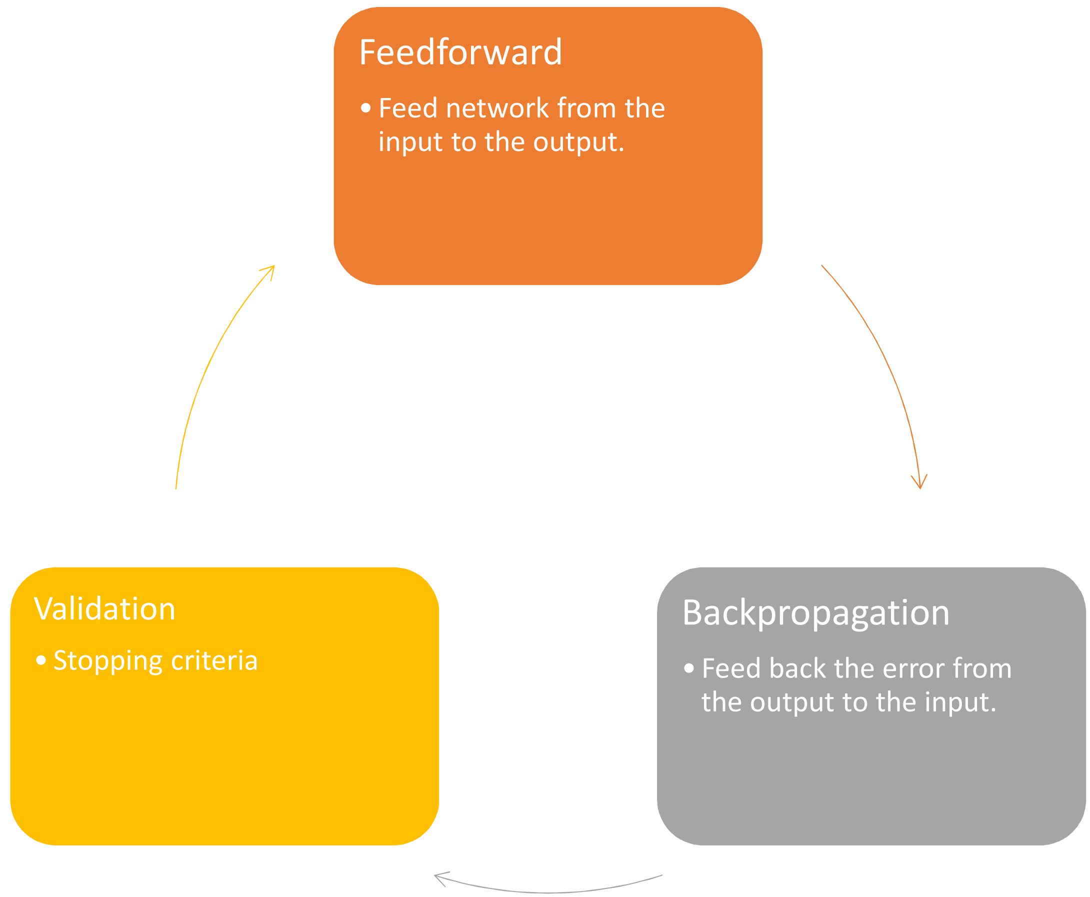
- Batch size: 1000
- Shuffle the data
- Create 100 batches:
  - Batch shapes: Shapes: X: 1000 x 2000, y: 1000

Batch example - 2

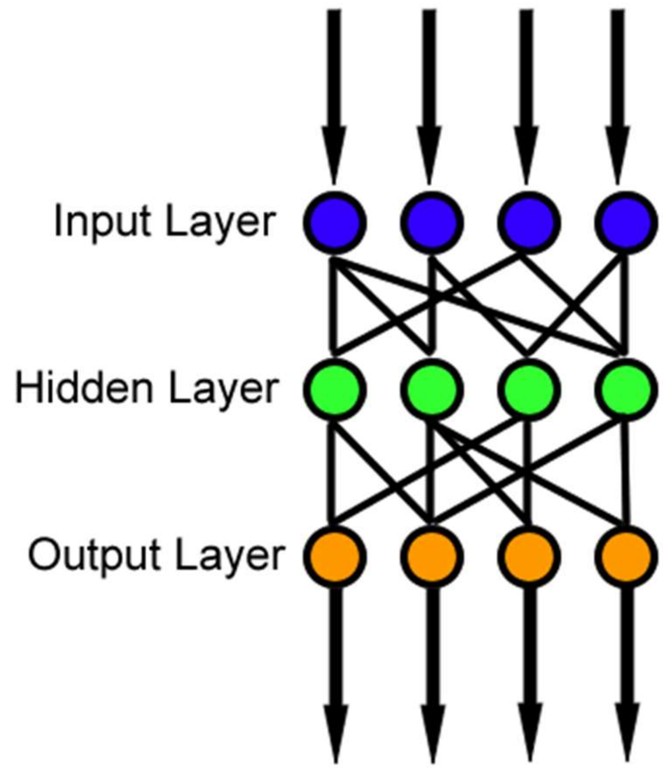
- 100K instances with 2K features for 5 categories classification:
  - Shapes: X: (100000, 2000), y: (100000,)

- Batch size: 8000
- Shuffle the data
- Create 13 batches:
  - Batch shapes: X: (8000, 2000), y: (8000,)
  - Last batch shapes: X: (4000, 2000), y: (4000,)

Training the network



Feedforward

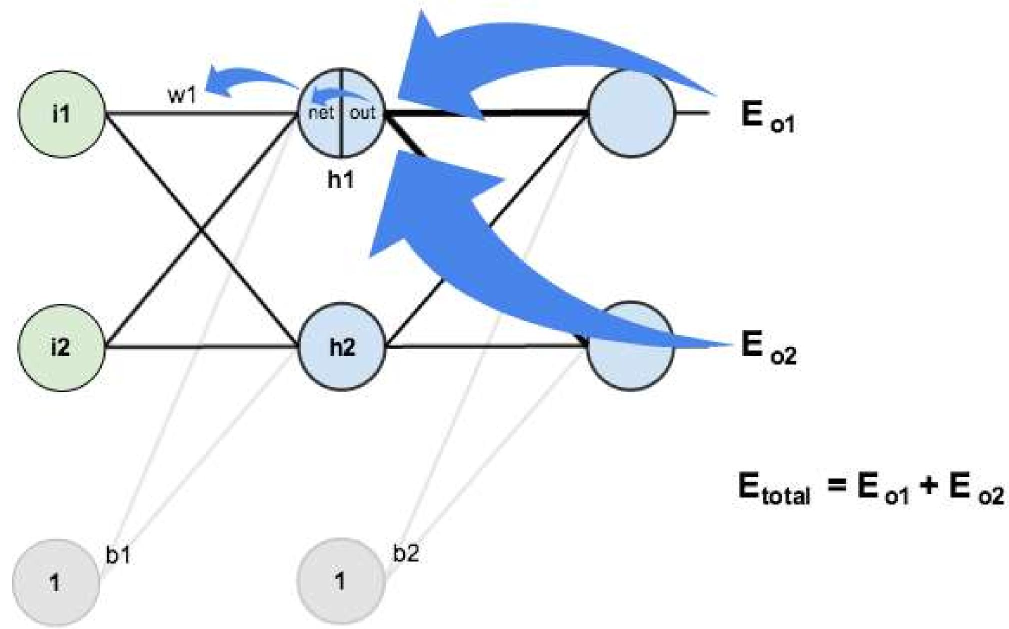


https://en.wikipedia.org/wiki/Feedforward_neural_network

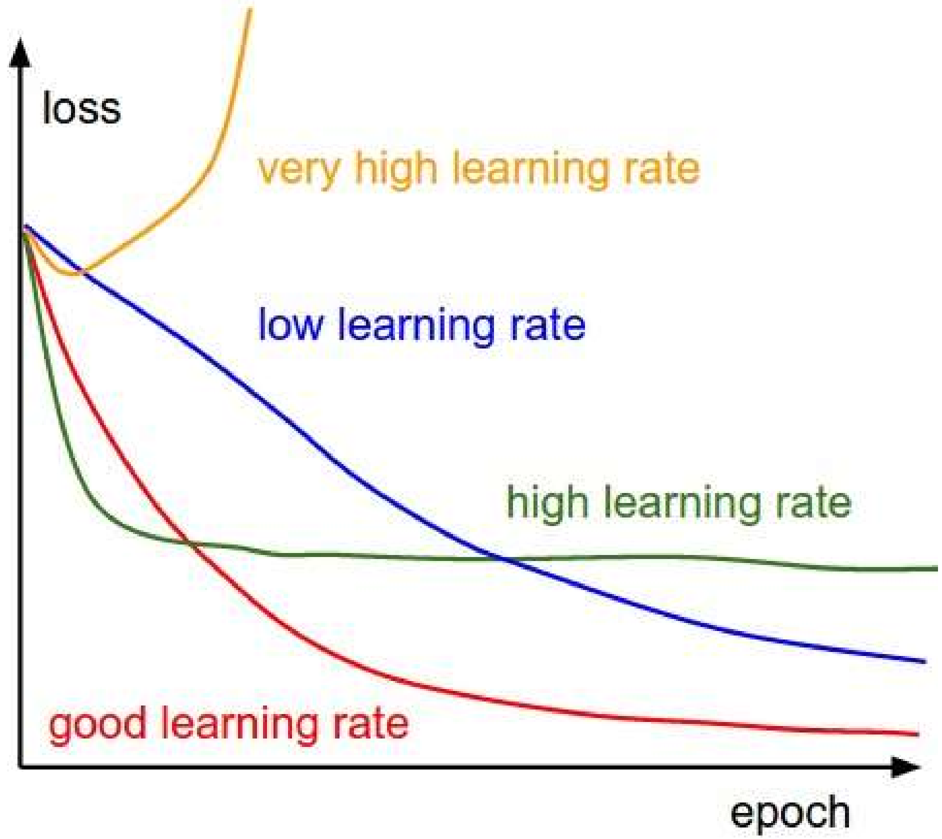
Backpropagation

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Learning rate



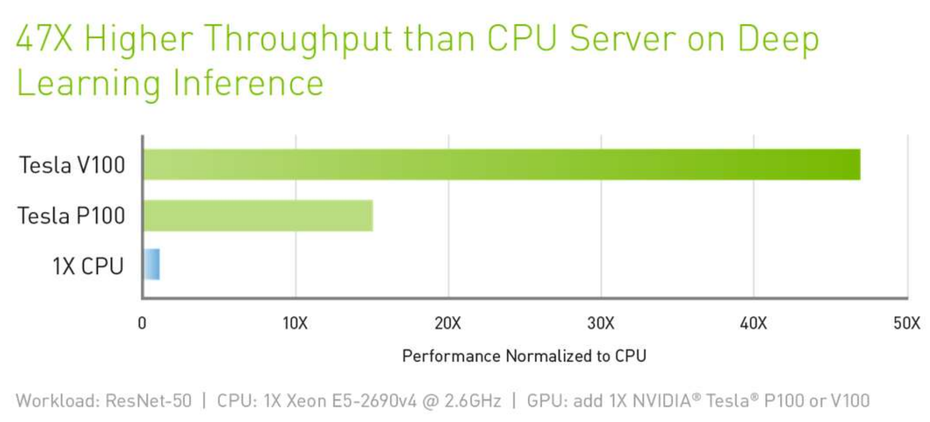
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Deep learning

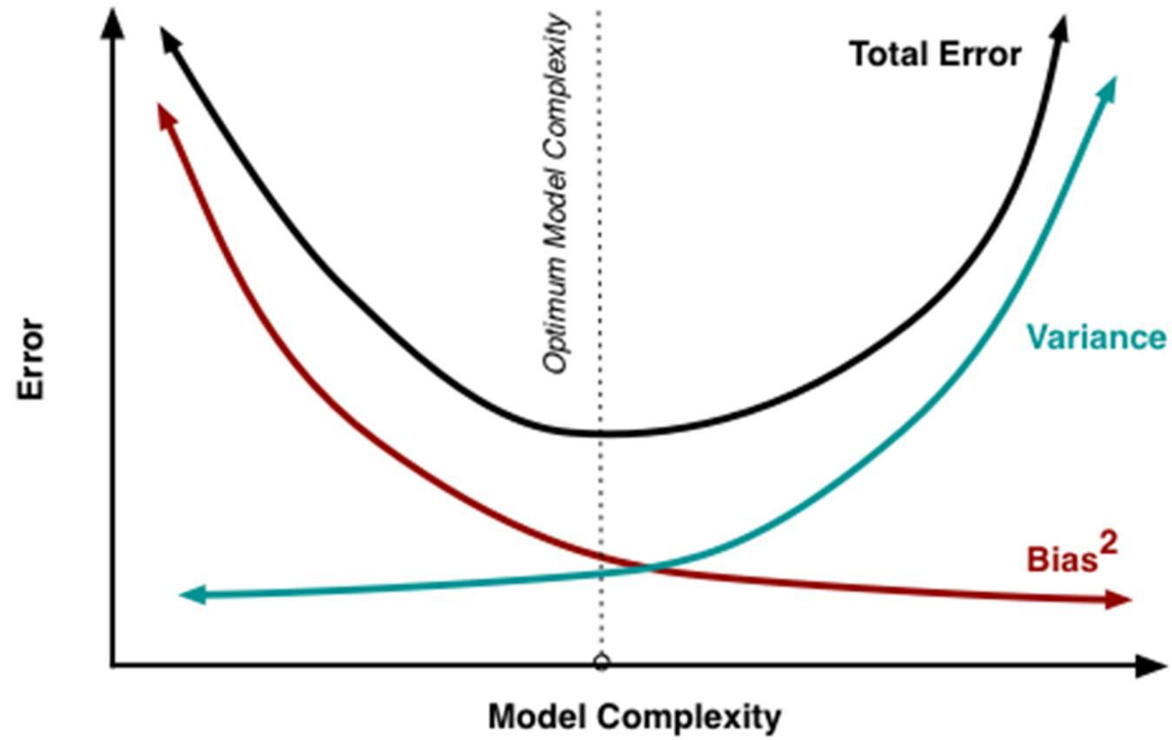
- A neural network with many layers
- Very complex networks
- Training is computation intensive

Training on Nvidia GPUs

- Using Cuda cores
- Packages:
  - Tensorflow
  - Mxnet
  - Pytorch
  - Caffe



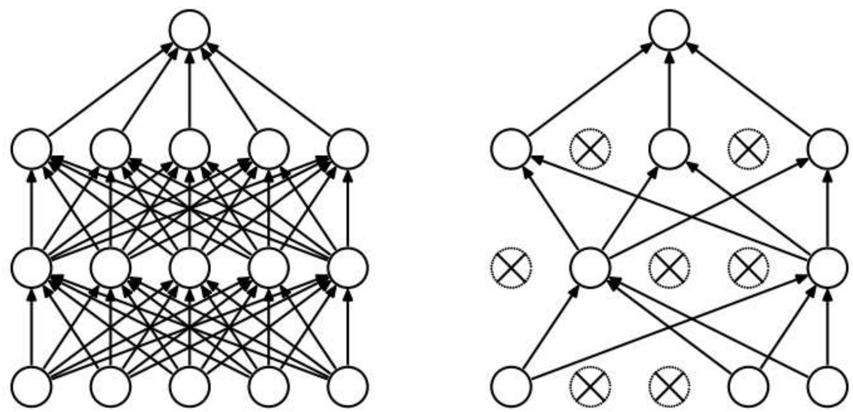
Bias vs. Variance



Regularization layers

- To reduce variance
- Different behavior in the training and the testing time
  - Dropout
  - Batch Normalization

Dropout



(a) Standard Neural Net (b) After applying 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.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Srivastava, JMLR, 2014

Batch Normalization

- Using a moving average to find the mean and average.
- Change the output of the layer to a zero mean vector with unit variance.
- Moving average update only at the training time.

Model weights

- Each layer may have multiple parameters.
- Dense layers have two parameters:
  - Coefficient vector per cell (Coefficient matrix).
  - Bias per cell (bias vector)
  - $l(z) = \tanh(z.\omega + b)$

Word embedding

- Language modeling
- Feature learning
- Words → vector (in low dimension)

Word2vect

- A MLP model
- Each word is mapped to the weights of the first layer.
  - First layer weight: # words × # cells
- We can use these vectors in different classification task.

One hot encoding

- Labels: [0, 1, 2, 1, 0]

- One hot encoding:

- $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$