Week 06 | Lecture 07
From Logistic Regression to Neural Networks

Wan Fang
Southern University of Science and Technology

- **Statistical Binary Classification**
  - Logistic Regression
  - Stochastic Gradient Descent
- **Multi-class Classification**
  - Loss Function
  - Softmax Regression
  - Summary of Linear Classifications
- **Neural Networks**
  - Multi-Layer Perceptron
  - Forward & Backward Propagation
  - Exercises

Statistical Binary Classification

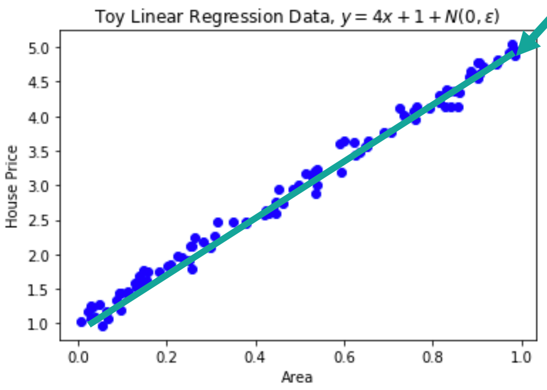
AncoraSIR.com

Statistical Binary Classification

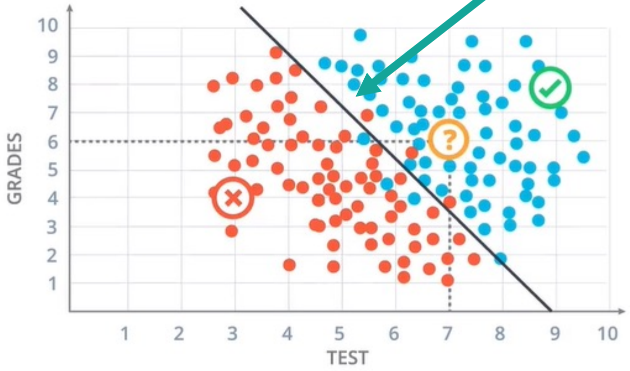
Recall on linear regression and classification

- Linear Regression
  - A basic linear model for line-fitting
  - $\hat{y} = f_{WeightedSum}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
- Linear Classification
  - Vectorized weights for two or multiple classes
  - $\mathbf{s} = f_{WeightedSum}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$

Estimate a line
to describe a relationship

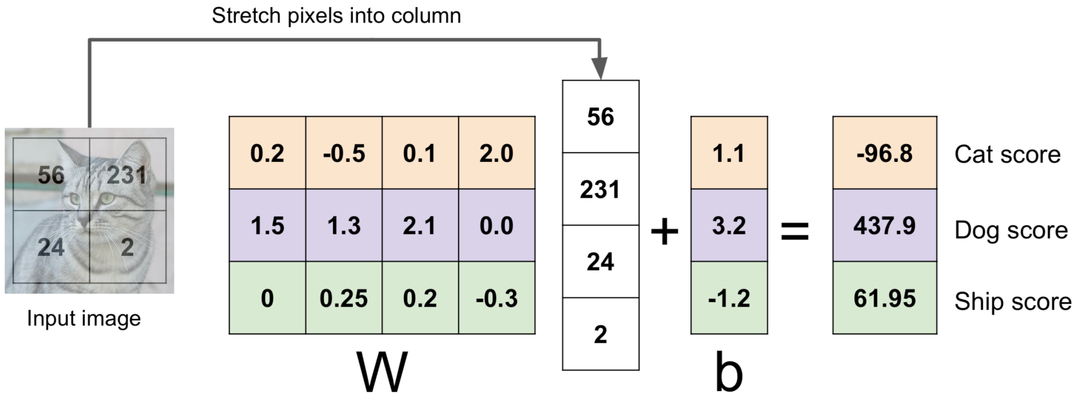


Predict University Acceptance
based on Test and Grades
(*only two categories*)



$$\hat{y} = g_{Activation}(s) = \begin{cases} 1 & if\ s \geq 0 \\ 0 & if\ s < 0 \end{cases}$$

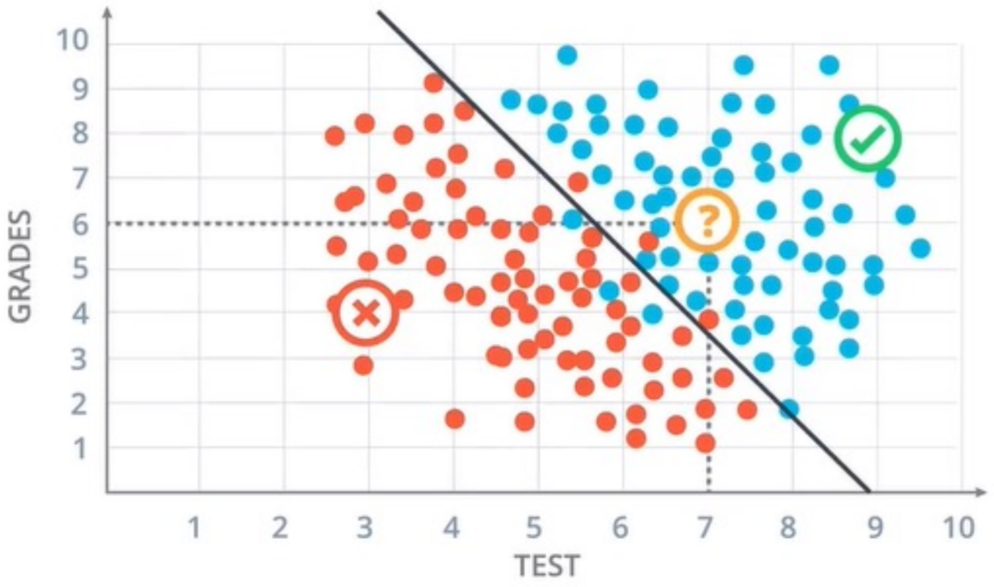
Is this picture a cat, a dog, or a ship?
(*Can we make a decision based on the results on the right?*)



AncoraSIR.com

Perceptron Algorithm

If data is separable by a large margin, then Perceptron is a good algorithm to use.



- Information lost about the distance to the cutoff value
- Uncertain about the final decision

What if the boundary line is non-linear?

- Unable to classify nonlinear scenarios

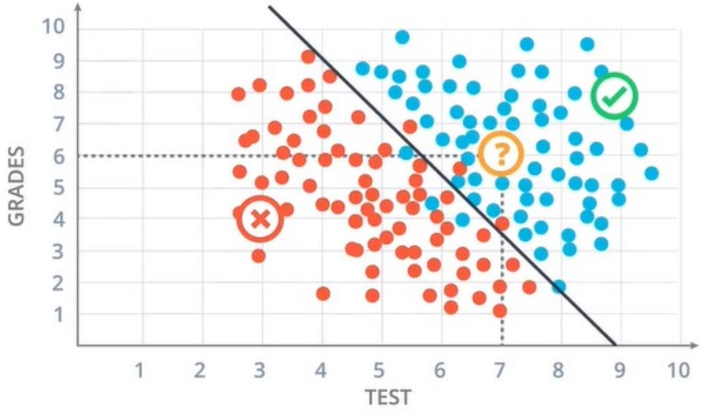
AncoraSIR.com

SUSTech
Southern University of Science and Technology

By what chances will I get accepted to a University?

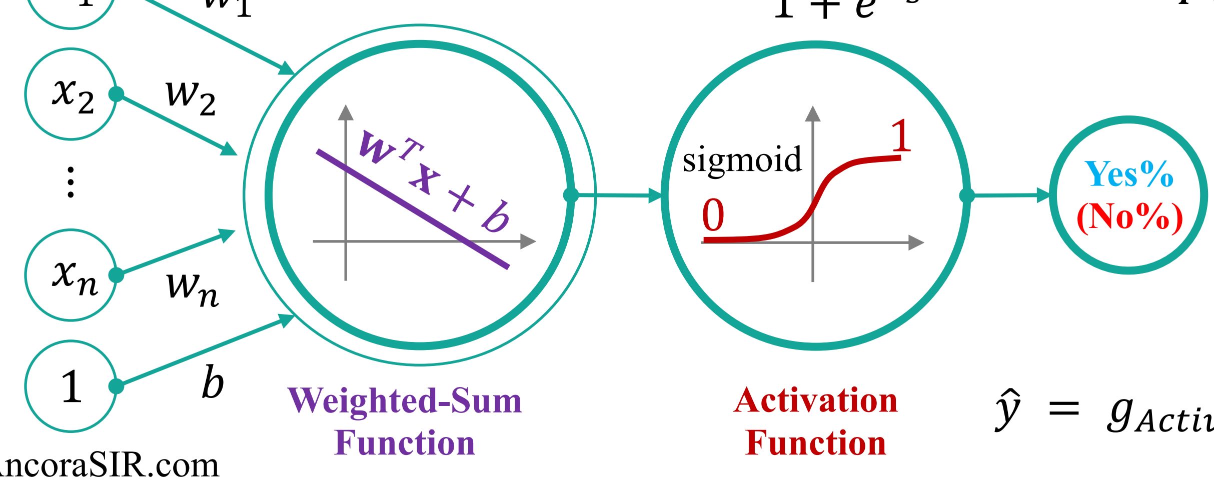
Based on my Test and Grade scores …

- **Weighted-sum node**
  - Unchanged as the input data remains the same
- **Activation node**
  - Can be changed as we want a new expression of the output as a probability of prediction
  - Sigmoid function as a natural choice that transforms the output to a value between 0 and 1 (*or within 100%*)



<u>*An example of acceptance at a University*</u>

$s = \mathbf{w}^T\mathbf{x} + b$ $\qquad \hat{y} = \dfrac{1}{1+e^{-s}}$

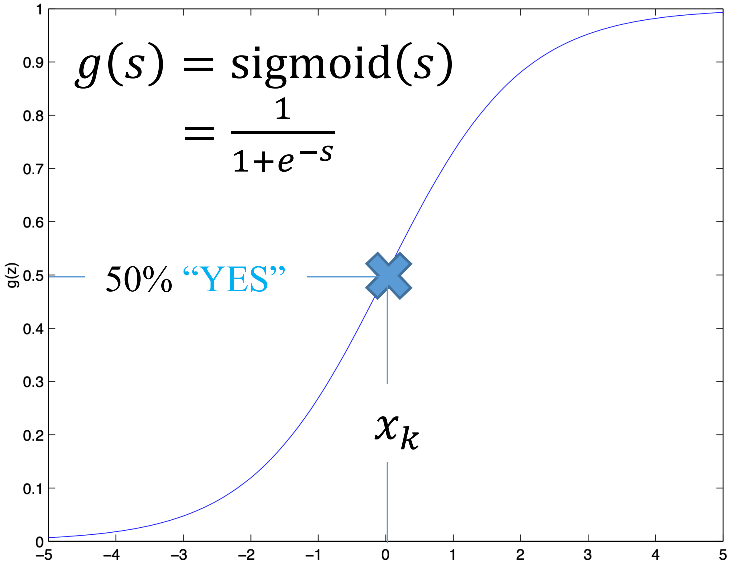


AncoraSIR.com

$$P(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

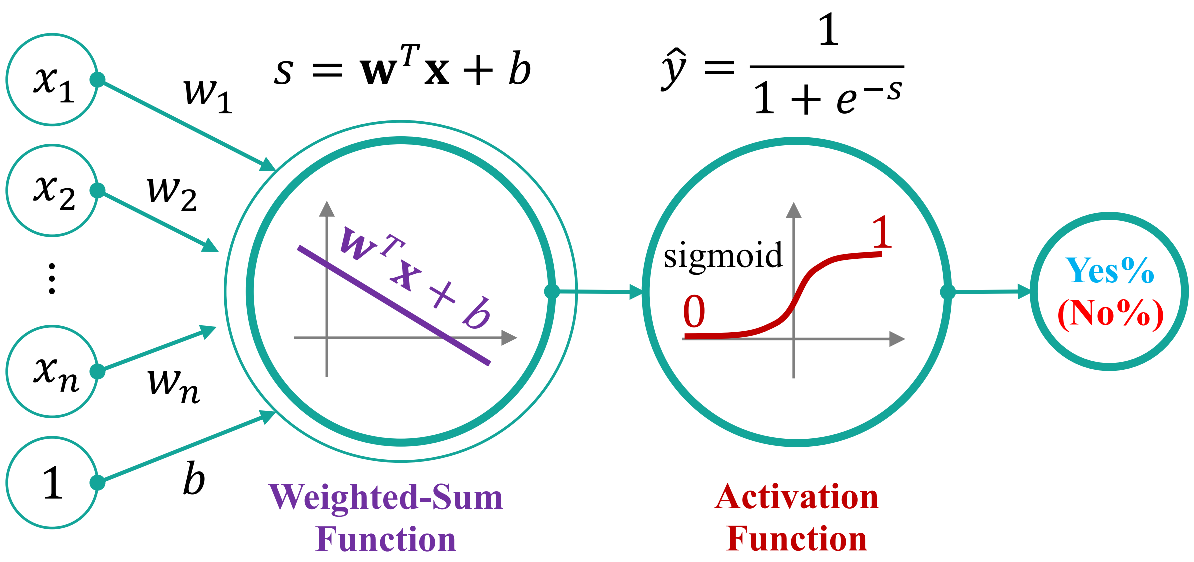
Can be viewed as a probability

$$\hat{y} = g_{Activation}(s) = \text{sigmoid}(s)$$



Logistic Regression

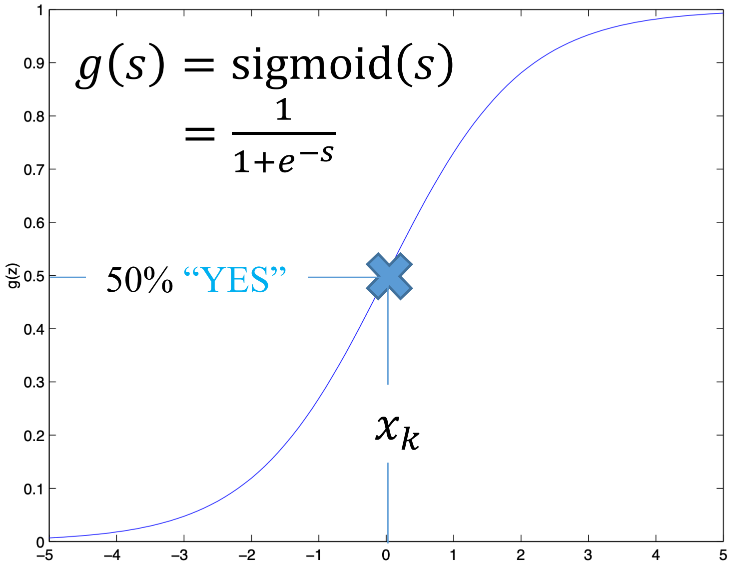
$$\hat{y} = g_{Activation}\left[f_{WeightedSum}(\mathbf{x})\right] = \text{sigmoid}(\mathbf{w}^T\mathbf{x} + b)$$



Problem statement

- Assume $\hat{y} = g_{Activation}\left[f_{WeightedSum}(\mathbf{x})\right] = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}}$
- How to minimize the **prediction error/loss** on a single training sample (with a maximum likelihood set of **w**) ?

- Hypothesis Function: $h_w(x) = \text{sigmoid}(\mathbf{w}^T\mathbf{x} + b)$
- Model output with a probability: $P(y \mid x; w) = [h_w(x)]^y[1 - h_w(x)]^{1-y}$
  - **Yes%**: $P(y = 1 \mid x; w) = h_w(x)$
  - **No%**: $P(y = 0 \mid x; w) = 1 - h_w(x)$



AncoraSIR.com

Loss Function for Logistic Regression

It measures how well you are doing on ***a single training example***

- Assume that m training examples were generated independently $h_w(x) = \text{sigmoid}(\mathbf{w}^T\mathbf{x} + b)$
- We can write the likelihood of the parameters
  - $L(w) = p(\vec{y} \mid X; w)$
    $= \prod_{i=1}^{m} p\left(y^{(i)} \mid x^{(i)}; w\right)$
    $= \prod_{i=1}^{m} \left[h_w\left(x^{(i)}\right)\right]^{y^{(i)}} \left[1 - h_w\left(x^{(i)}\right)\right]^{1-y^{(i)}}$
- Take the log expression, we have the **loss function**
  - $\ell(w) = \log L(w)$
    $= \sum_{i=1}^{m} y^{(i)} \log h_w\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_w\left(x^{(i)}\right)\right)$
  - Usually take a "—" sign to indicate loss

AncoraSIR.com

Stochastic Gradient Descent

Finding the maximum likelihood of estimation

- Rewrite the weight parameters in vectorized form
  - $w := w + \alpha \cdot \nabla_w \cdot \ell(w)$
  - $+$ sign here to **maximize** likelihood

- When working with a single training example (x, y),
  - $\frac{\partial}{\partial w_j} \ell(w) = \left(y \frac{1}{g(w^T x)} - (1-y) \frac{1}{1-g(w^T x)}\right) \frac{\partial}{\partial w_j} g(w^T x) = \left(y - h_w(x)\right) x_j$

- Therefore, we can derive the stochastic gradient ascent rule
  - $w_j := w_j + \alpha \left(y^{(i)} - h_w\left(x^{(i)}\right)\right) x_j^{(i)}$

AncoraSIR.com

SUSTech
Southern University of Science and Technology

Cost Function

It measures how well you are doing on ***an entire training set***

- We want the loss/error function to be as small as possible
  - If $y^{(i)} = 1$, then
    - $\text{LossFunc}(\hat{y}, y) = -\left[y^{(i)} \log h_w\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_w\left(x^{(i)}\right)\right)\right] = -\log h_w\left(x^{(i)}\right) = -\log \hat{y}$
    - It means that we want $\log \hat{y}$ to be as big as possible, but remember that it is bounded by 1
  - If $y^{(i)} = 0$, then
    - $\text{LossFunc}(\hat{y}, y) = -\left[y^{(i)} \log h_w\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_w\left(x^{(i)}\right)\right)\right] = -\log(1 - \hat{y})$
    - It means that we want $\log \hat{y}$ to be as small as possible, or close to 0

- Cost Function
  - The average of the loss functions of the entire training set, which is to be minimized

$$J(w, b) = -\frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right]$$

Summary

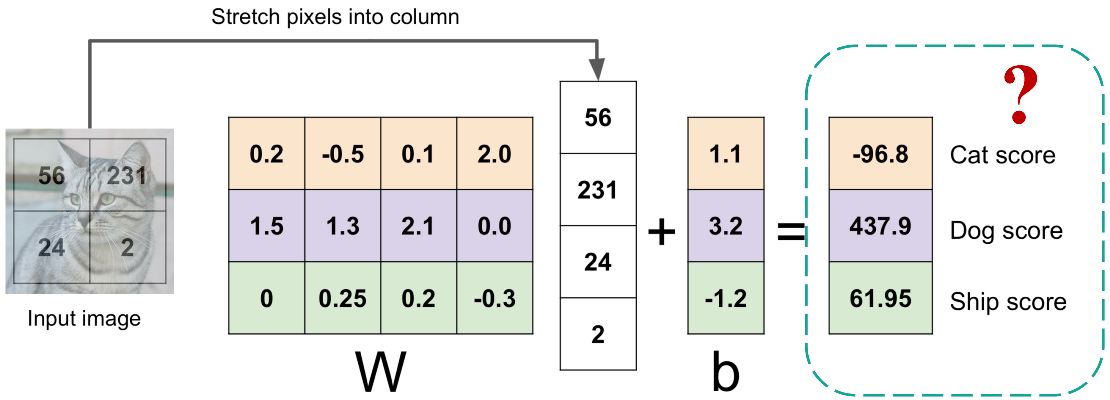
	Linear Regression	Perceptron	Logistic Regression
Problem	Value Prediction	Binary Classification with a threshold	Binary Classification with a probability
Weighted-Sum	$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$	$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$	$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$
Activation Function	NA	Step Function	Sigmoid Function
Prediction Outputs	Continuous Value	Discrete Value $\{0, 1\}$	Continuous Probability $(0, 1)$
Loss	Squared Loss	Hinge Loss	Log-Loss

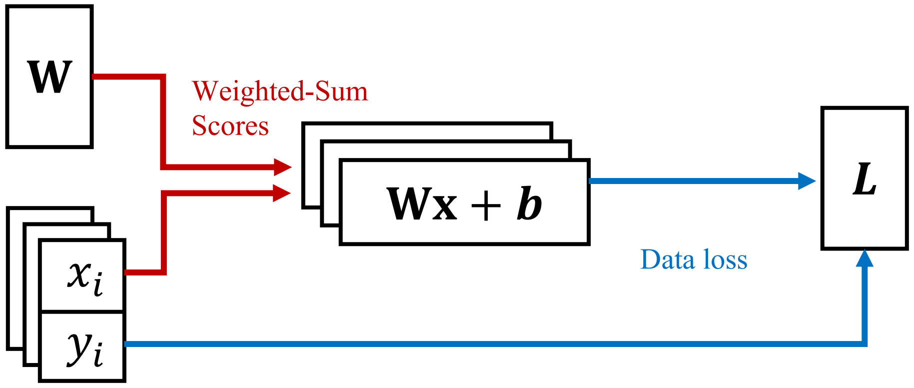
Multi-class Classification

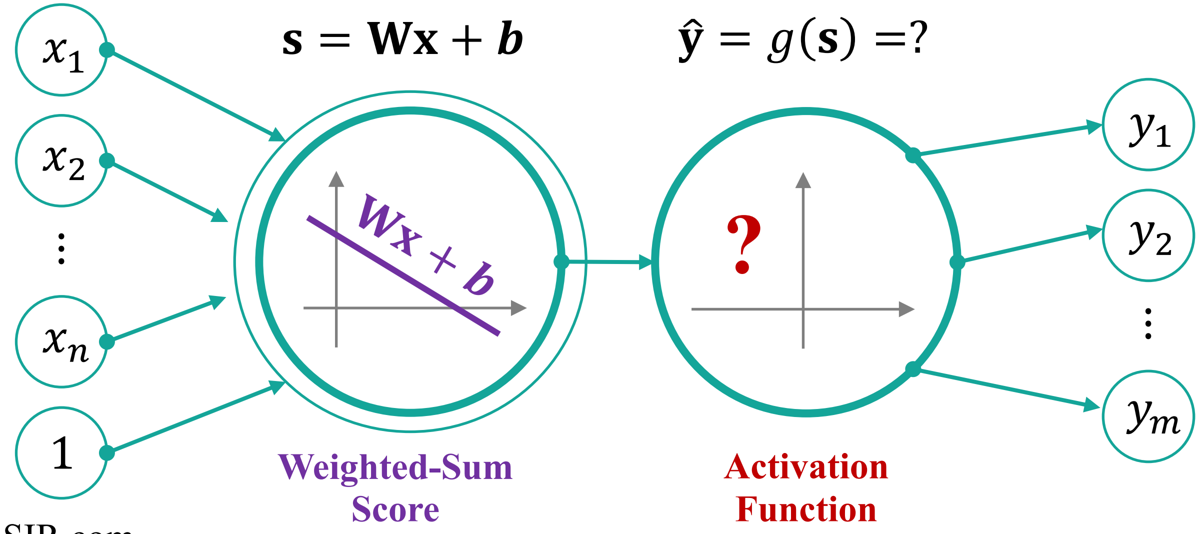
AncoraSIR.com

Multi-class Classification

$$\hat{\mathbf{y}} = g_{Activation}\left[f_{WeightedSum}(\mathbf{x})\right] = g_{Activation}(\mathbf{Wx} + \mathbf{b})$$







1. **Define a loss function** that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. (***optimization***)

AncoraSIR.com

Define a Loss Function

Quantify how good our current classifier is

3 training samples

$\{(x_i, y_i)\}_{i=1}^{3}$

Ground Truth

$\hat{y}_i$ *Labelled Prediction*

x_i image

y_i label

3 classes

Cat	**3.2**	1.3	2.2
Car	5.1	**4.9**	2.5
Frog	-1.7	2.0	**-3.1**

Loss over the dataset is a sum of loss over examples

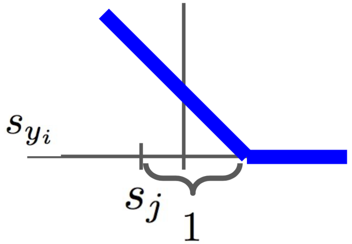
$$L = \frac{1}{N} \sum_i L_i(\hat{y}_i, y_i)$$

Denote Weighted-Sum score vector as $\boldsymbol{s} = f_{WeightedSum}(\mathbf{x})$

Let's try with the hinge loss:

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



AncoraSIR.com

SUSTech Southern University of Science and Technology

Define a Loss Function

Quantify how good our current classifier is

3 training samples

$\{(x_i, y_i)\}_{i=1}^{3}$

Ground Truth

Labelled Prediction

3 classes			
Cat	**3.2**	1.3	2.2
Car	5.1	**4.9**	2.5
Frog	-1.7	2.0	**-3.1**
Loss	2.9	0	**12.9**

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$\begin{aligned} L_1 &= \max(0,\, 5.1 - 3.2 + 1) + \max(0,\, -1.7 - 3.2 + 1) \\ &= \max(0,\, 2.9) + \max(0,\, -3.9) \\ &= 2.9 + 0 = 2.9 \end{aligned}$$

$$\begin{aligned} L_2 &= \max(0,\, 1.3 - 4.9 + 1) + \max(0,\, 2.0 - 4.9 + 1) \\ &= \max(0,\, -2.6) + \max(0,\, -1.9) \\ &= 0 + 0 = 0 \end{aligned}$$

$$\begin{aligned} L_3 &= \max(0,\, 2.2 + 3.1 - 1) + \max(0,\, 2.5 + 3.1 - 1) \\ &= \max(0,\, 6.3) + \max(0,\, 6.6) \\ &= 6.3 + 6.6 = 12.9 \end{aligned}$$

Define a Loss Function

Quantify how good our current classifier is

3 training samples

$\{(x_i, y_i)\}_{i=1}^{3}$

Ground Truth

Labelled Prediction

3 classes

Cat	**3.2**	1.3	2.2
Car	5.1	**4.9**	2.5
Frog	-1.7	2.0	**-3.1**
Loss	2.9	0	**12.9**

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

Loss over full dataset is average:

$$L = \frac{1}{N} \sum_i L_i(\hat{y}_i, y_i)$$
$$= \frac{1}{3}(2.9 + 0 + 12.9)$$
$$= 5.27$$

Recall that our goal is to find a set of **W** with minimum loss over full dataset, i.e. the cost = 0

- Suppose that we found a W such that L = 0. Is this **W** unique?
  - L is still 0 with 2**W**
- **How do we choose between W and 2W?**
  - Let's try regularization

AncoraSIR.com

Regularization

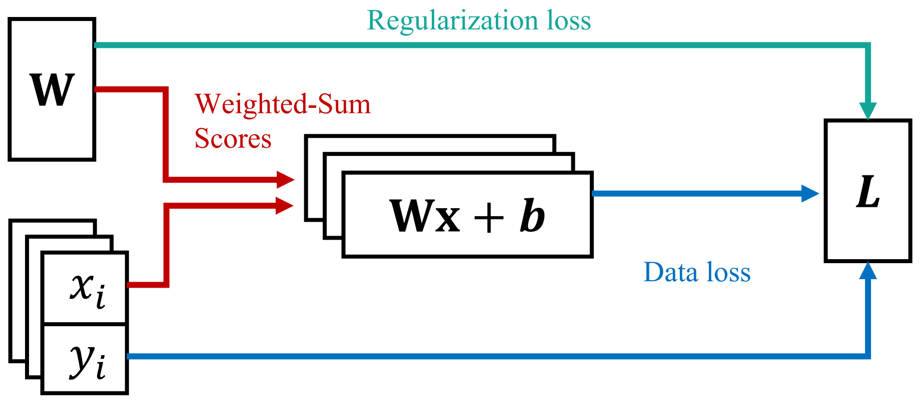
Prevent the model from doing too well on training data

$$L(W) = \boxed{\frac{1}{N}\sum_{1}^{N} L_i(\hat{y}_i, y_i)} + \boxed{\lambda R(W)}$$

λ as strength of Regularization (*hyperparameter*)

Data loss
Model predictions should match training data

Regularization
Prevent the model from doing too well on training data



Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Why regularize?

- Express preferences over weights
- Make the model simple so it works on test data
- Improve optimization by adding curvature

AncoraSIR.com

SUSTech Southern University of Science and Technology

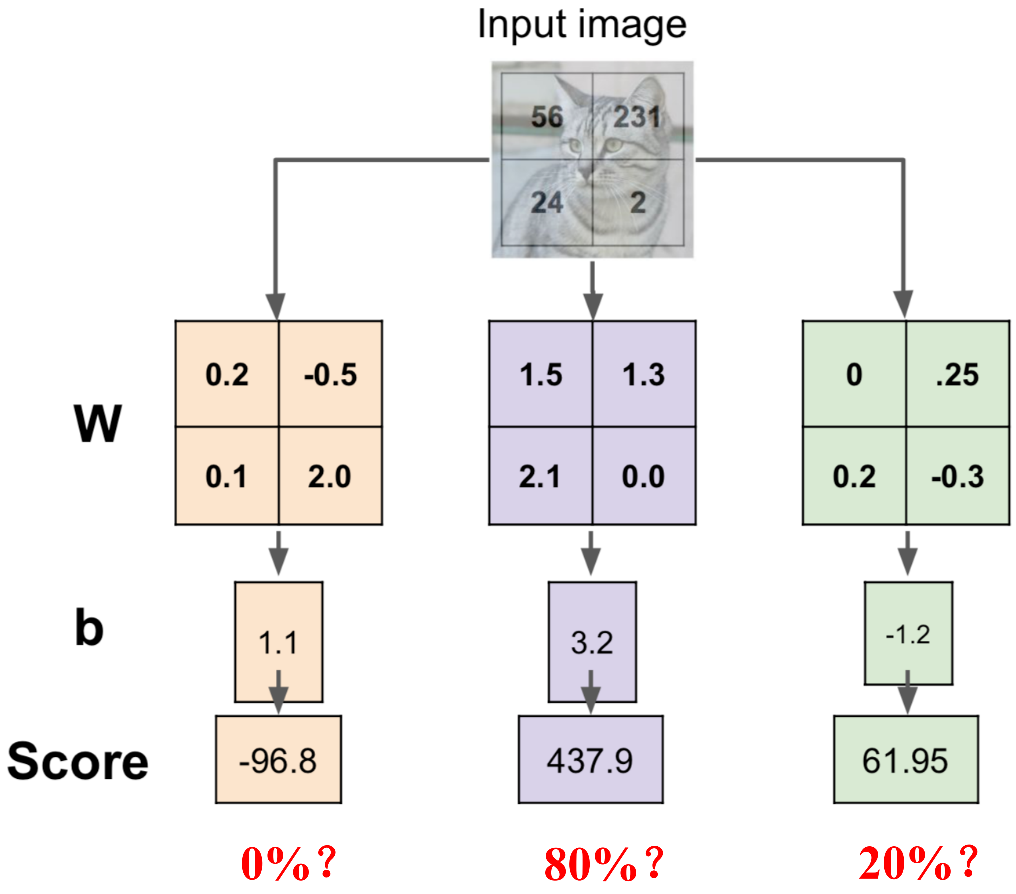
Softmax Operation

Interpret the outputs of our model as probabilities

$$\hat{y}_i = \mathrm{softmax}(o_i) = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \times 100\%$$

- One can interpret outputs $\hat{y}_i$ as the probability that a given item belongs to class i.
- Then we can choose the class with the largest output value as our prediction

  - Why using o_i directly, instead of a probability?
  - What if the sum of probability is not 100%?
  - What if when o_i becomes negative?



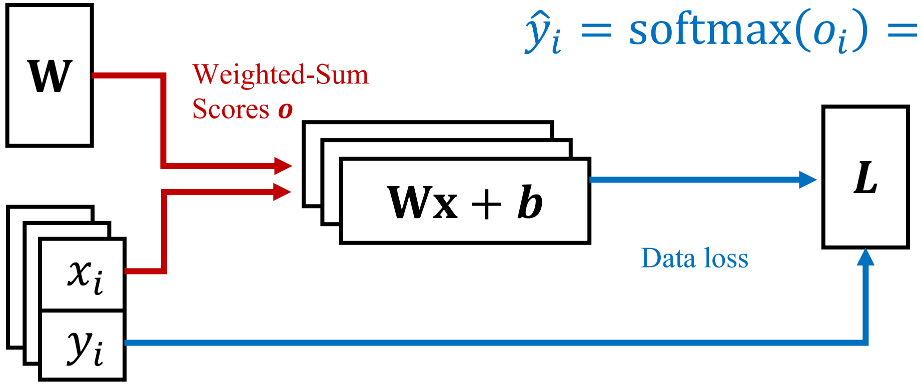
AncoraSIR.com

Softmax Classifier

*Want to interpret raw classifier scores as **probabilities***

(x_1, y_1) *Ground Truth*

Labelled Prediction



$$\hat{y}_i = \mathrm{softmax}(o_i) = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \times 100\%$$

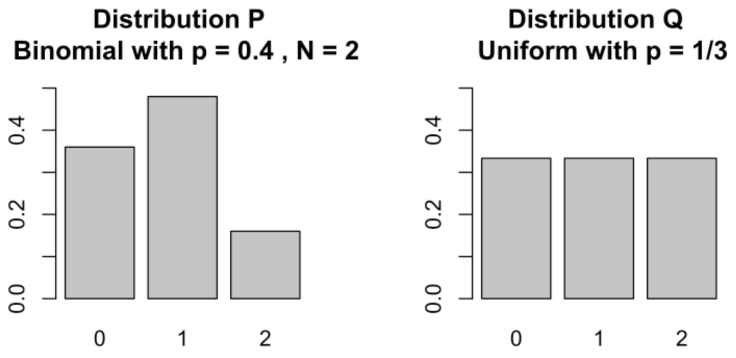
3 classes	Weighted-Sum Scores (*logits*)	$\xrightarrow{\exp(o_i)}$ Unnormalized Probability (>=0)	$\xrightarrow[(\times 100\%)]{\frac{\exp(o_i)}{\sum_j \exp(o_j)}}$ Normalized Probability Label ($\hat{y}_i$ *Sum to 1*)	Correct Probability Label
Cat	**3.2**	**24.5**	**0.13**	**1.00**
Car	5.1	164.0	0.87	0.00
Frog	-1.7	0.18	0.00	0.00

How to quantify the differences between two probability distribution?

AncoraSIR.com

Kullback–Leibler Divergence

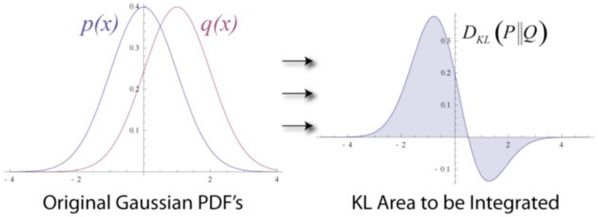
How to quantify the differences between two probability distribution?



x	0	1	2
Distribution P(x)	0.36	0.48	0.16
Distribution Q(x)	0.333	0.333	0.333

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right)$$
$$= 0.36 \ln\left(\frac{0.36}{0.333}\right) + 0.48 \ln\left(\frac{0.48}{0.333}\right) + 0.16 \ln\left(\frac{0.16}{0.333}\right)$$
$$= 0.0852996$$

AncoraSIR.com



$$D_{KL}(P \parallel Q) = \sum_{y \in \mathcal{Y}} P(y) \log \frac{P(y)}{Q(y)}$$
$$= \sum_{y \in \mathcal{Y}} P(y) \log P(y) - \sum_{y \in \mathcal{Y}} P(y) \log Q(y)$$
$$= \left[-\sum_{y \in \mathcal{Y}} P(y) \log Q(y)\right] - \left[-\sum_{y \in \mathcal{Y}} P(y) \log P(y)\right]$$
$$= \mathrm{H}(P, Q) - \mathrm{H}(P)$$

A good candidate of loss function for softmax

Can be minimized to update the weights

$$\mathrm{H}(P, Q) = -\sum_{y \in \mathcal{Y}} P(y) \log Q(y)$$

the cross-entropy of P and Q

$$\mathrm{H}(P) = -\sum_{y \in \mathcal{Y}} P(y) \log P(y)$$

the cross-entropy of P with itself (or the entropy of P)

SUSTech Southern University of Science and Technology

Loss Function

Log-Likelihood expressed in cross-entropy

- The **likelihood** of the actual classes according to our model is

$$P(Y \mid X) = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}) \quad \Rightarrow \quad -\log P(Y \mid X) = \sum_{i=1}^{n} \boxed{-\log P(y^{(i)} \mid x^{(i)})}$$

  - Maximizing the likelihood is equivalent to minimizing the log-likelihood.

- **Cross-entropy** loss for a single example (dropped superscript *i*)

$$l = -\log P(y \mid x) = -\sum_{j} y_j \log \hat{y}_j$$

  - As $\hat{y}$ is a discrete probability distribution and y is a one-hot vector, the sum over all j vanishes for all but one term.

AncoraSIR.com

Cross-Entropy Loss and its Derivative

Also called softmax loss

- Plugging **o** into the definition of the cross-entropy loss, we obtain:

$$l = -\sum_j y_j \log \hat{y}_j = \sum_j y_j \log \sum_k \exp(o_k) - \sum_j y_j o_j = \log \sum_k \exp(o_k) - \sum_j y_j o_j$$

- The derivative with respect to **o** is

$$\partial_{o_j} l = \frac{\exp(o_j)}{\sum_k \exp(o_k)} - y_j = \mathrm{softmax}(\mathbf{o})_j - y_j = P(y = j \mid x) - y_j$$

  - The gradient is $P(y = j \mid x) - y_j$
    - The difference between the probability predicted by our model $P(y = j \mid x)$ and the true label y.
  - Similar to regression where the gradient is $\hat{y} - y$
    - The difference between the true label y and the estimation $\hat{y}$

AncoraSIR.com

SUSTech Southern University of Science and Technology

Vectorization for Minibatches

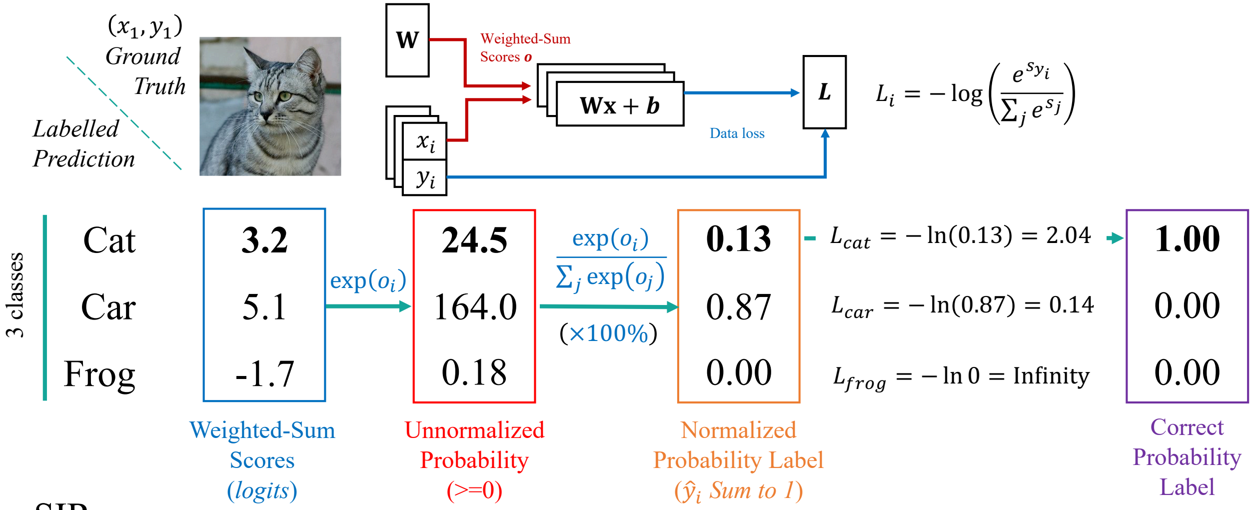
We typically carry out vector calculations for minibatches of data for efficiency

$$\hat{y}_i = \mathrm{softmax}(o_i) = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \times 100\%$$

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \ \ where \ \ \hat{y}_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \times 100\%$$

minibatch features $\mathbf{X}$ are in $\mathbb{R}^{n \times d}$,
weights $\mathbf{W} \in \mathbb{R}^{d \times q}$, and
the bias satisfies $\mathbf{b} \in \mathbb{R}^{q}$.

$$\mathbf{O} = \mathbf{XW} + \mathbf{b},$$
$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{O})$$



A minibatch $\mathbf{X}$ of examples
- dimensionality d and batch size n

Assume that we have q categories (outputs)

More efficient matrix-matrix computation $\mathbf{XW}$
Exponentiating all entries in $\mathbf{O}$ then sum

AncoraSIR.com

Understanding of Softmax Regression

- When there are two classes, softmax regression reduces to logistic regression.

Softmax	Binary Classes →	Logistic
$\hat{y}_j = \frac{\exp(o_j)}{\sum_j \exp(o_j)}$	*Activation*	$\hat{y} = \frac{\exp(o)}{\exp(o) + 1}$
$-\frac{1}{n}\sum_{i=1}^{n} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)$	*Loss*	$-\sum_{i=1}^{n}\sum_j y_j^{(i)} \log \hat{y}_j^{(i)}$

- Softmax when j=2

$$\hat{y}_0 = \frac{\exp(o_0)}{\exp(o_0) + \exp(o_1)}$$

$$= \frac{\exp(o_0 - o_1)}{\exp(o_0 - o_1) + 1}$$

- The cross-entropy classification can be thought in two ways
  1. As maximizing the likelihood of the observed data.
  2. As minimizing out surprise required to communicate the labels.

AncoraSIR.com

Summary & Comparison

Linear Neural Network

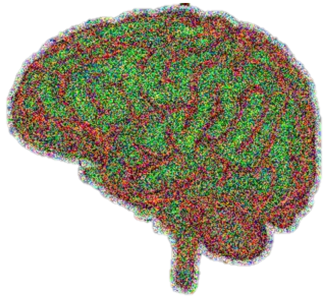
	Linear Regression	Perceptron	Logistic Regression	Softmax Regression
Problem	Value Prediction	Binary Classification	Binary Classification	Multi-Class Classification
Weights	$\boldsymbol{wx} + b$	$\boldsymbol{wx} + b$	$\boldsymbol{wx} + b$	$W\boldsymbol{x} + B$
Activation Function	NA	Step Function	Sigmoid Function	Softmax
Prediction Outputs	Continuous Value	Discrete Value 0, 1	Continuous Probability in (0,1)	A vector of Continuous Probabilities
Loss	Squared Loss	Hinge Loss	Log Loss (Binary cross entropy)	Cross Entropy
Decision Boundary	Linear			

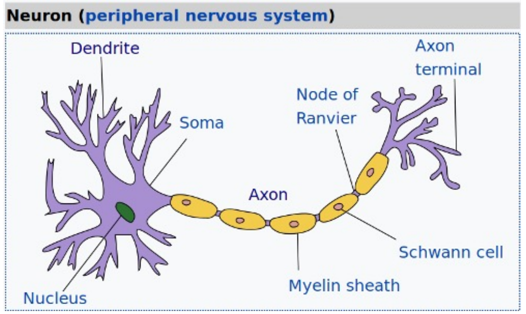
Neural Network

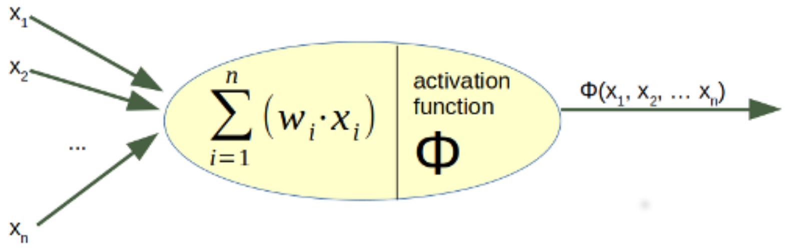
AncoraSIR.com

What is a Neural Network?

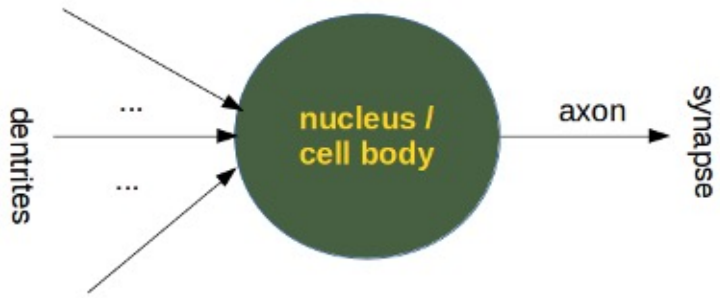
From biological inspiration to mathematical modeling

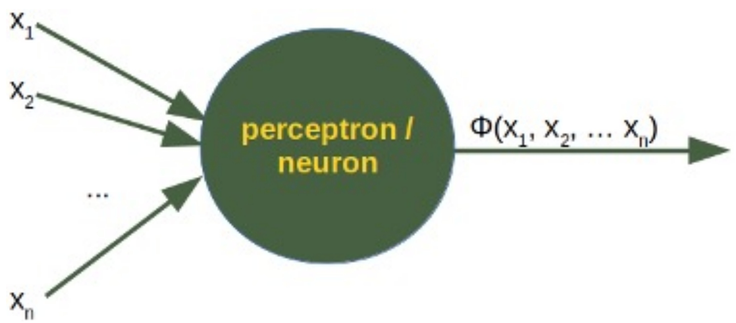






Neural Network Abstraction





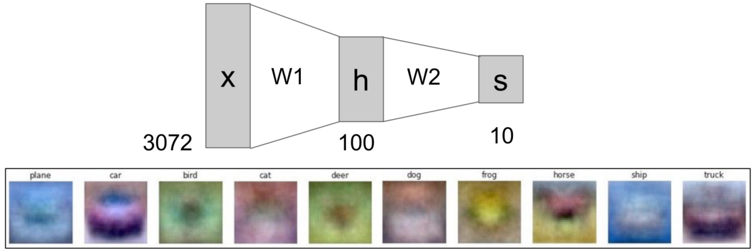
AncoraSIR.com

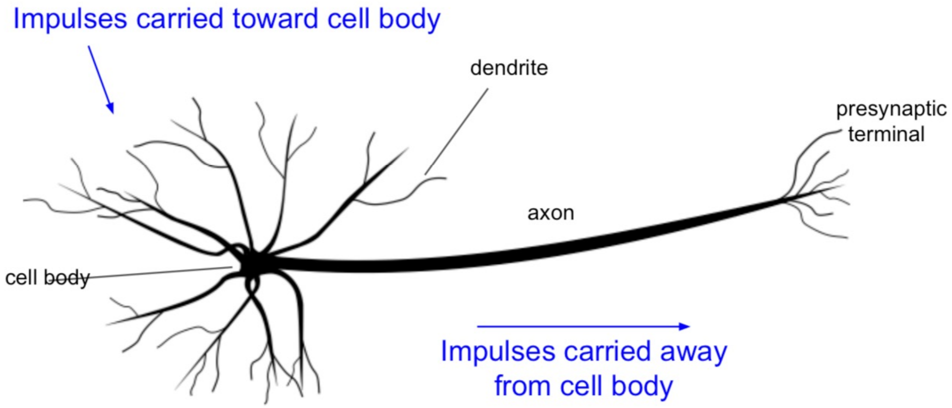
A Perceptron as an Artificial Neuron

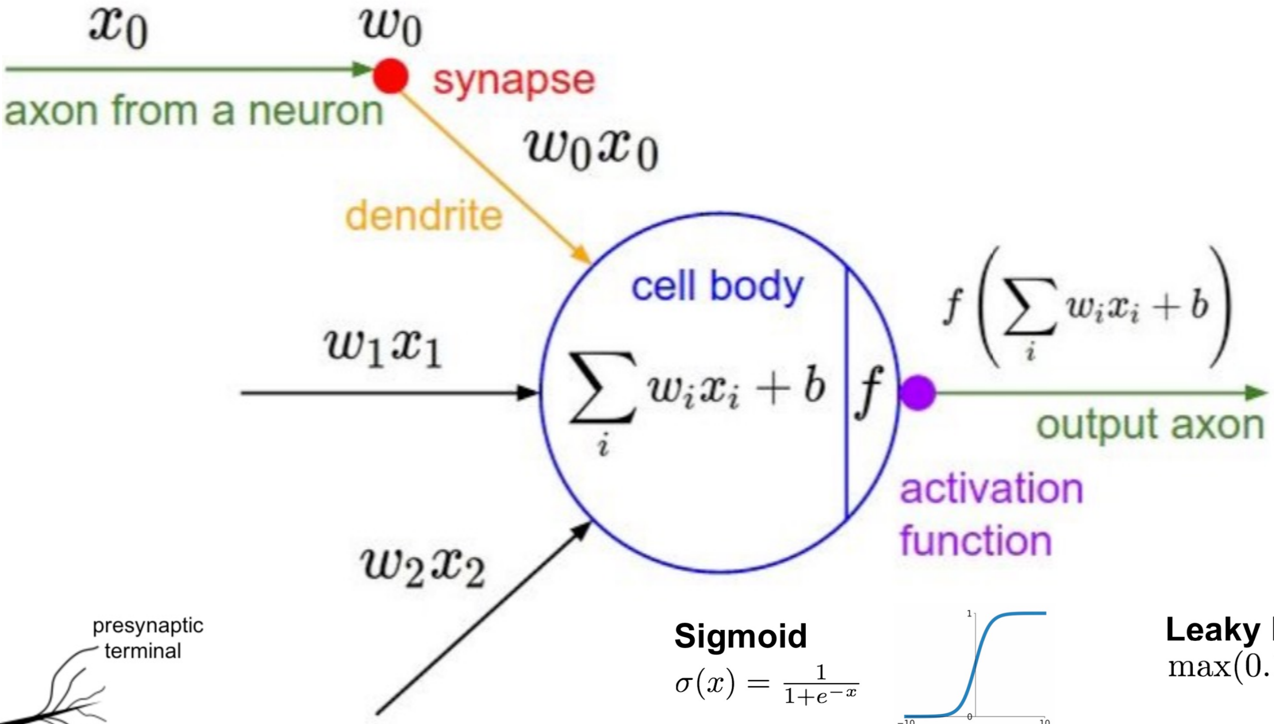
Biological Inspiration

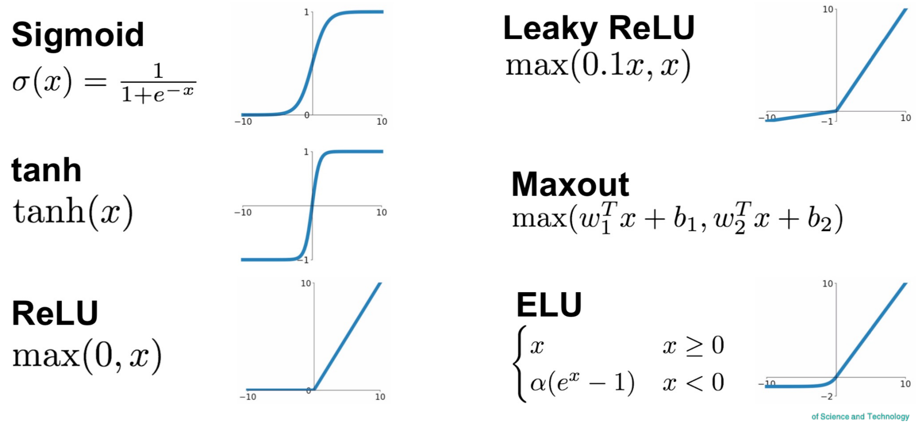
Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$



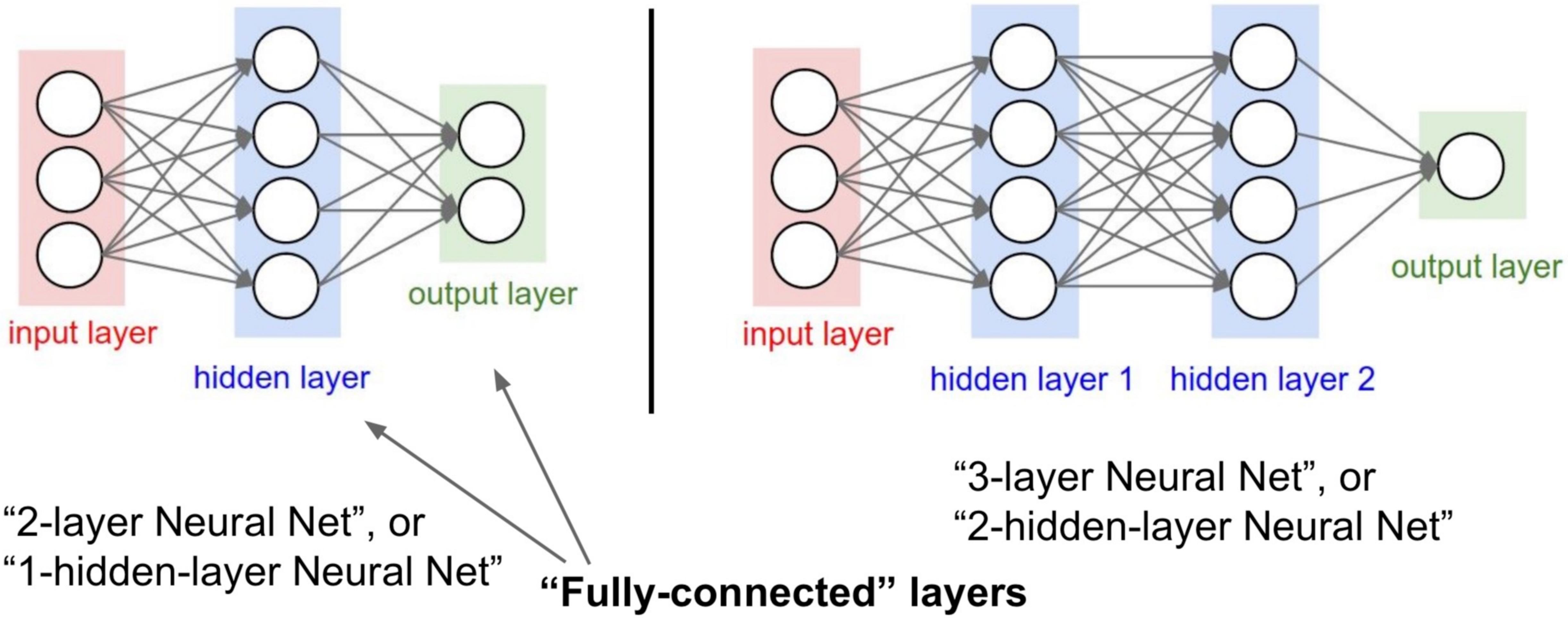






Multi-Layer Perceptrons

Artificial Neural Networks

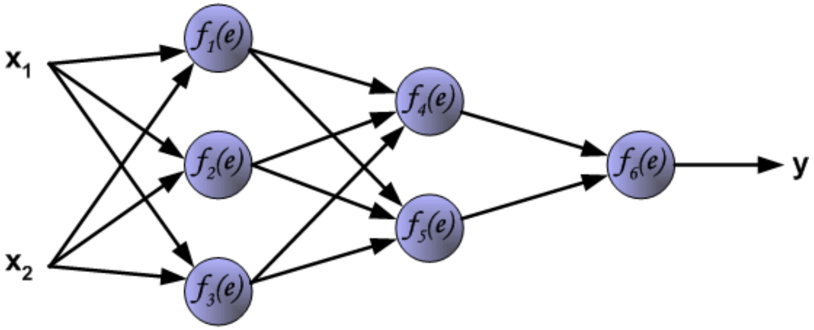


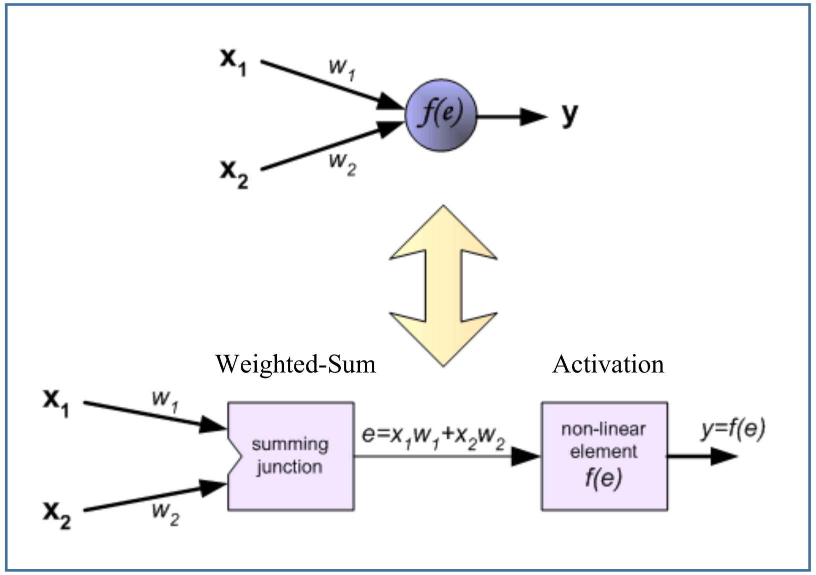
AncoraSIR.com

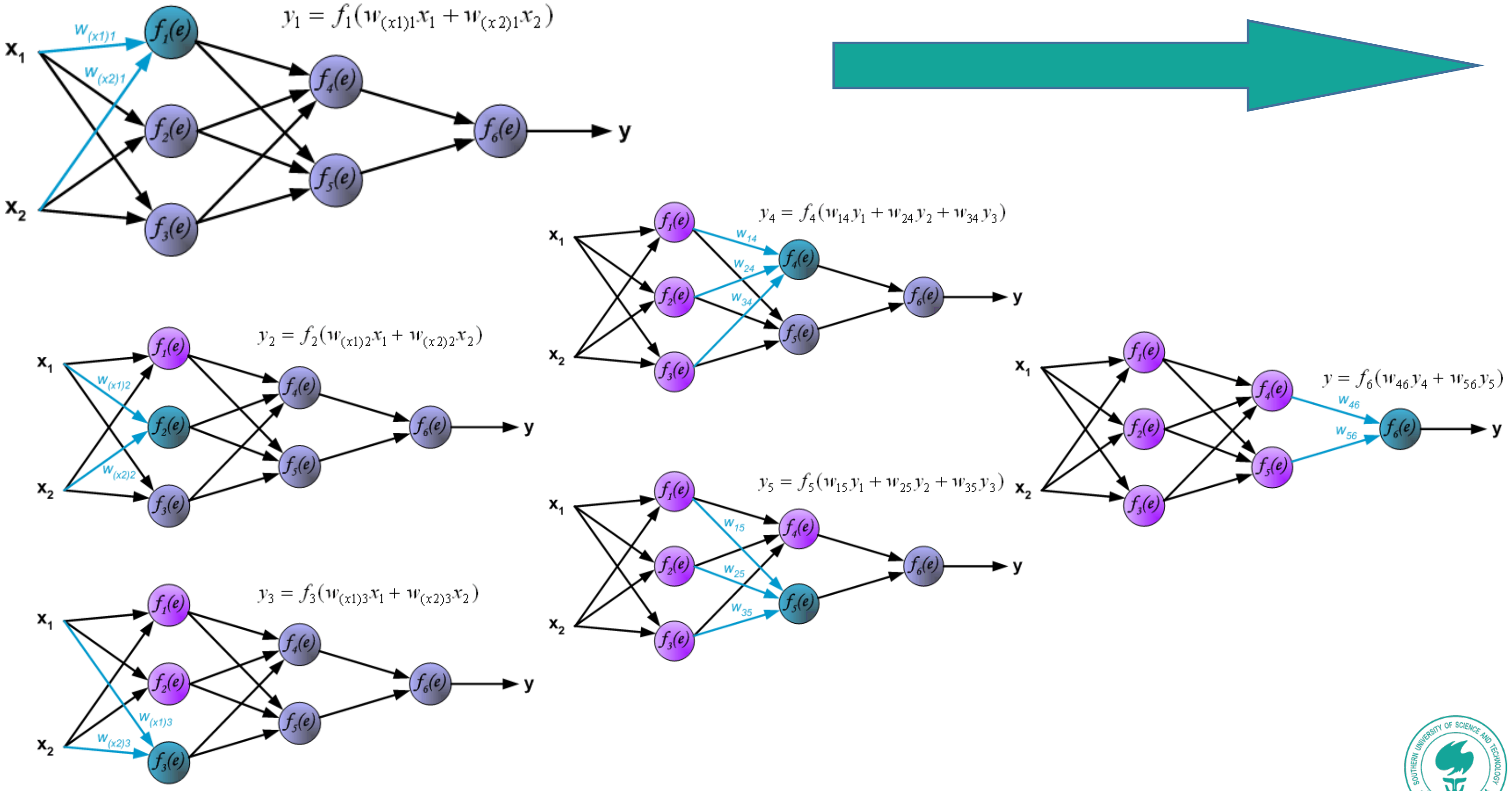
SUSTech Southern University of Science and Technology

Forward Propogation

Accept inputs to train a Multi-layer Neural Network





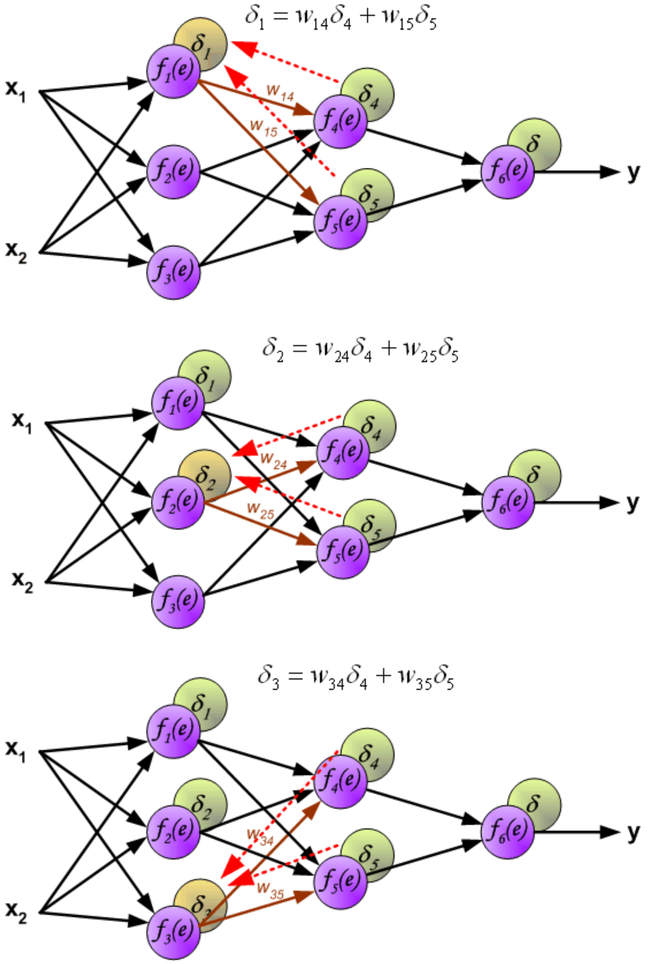


http://galaxy.agh.edu.pl/%7Evlsi/AI/backp_t_en/backprop.html

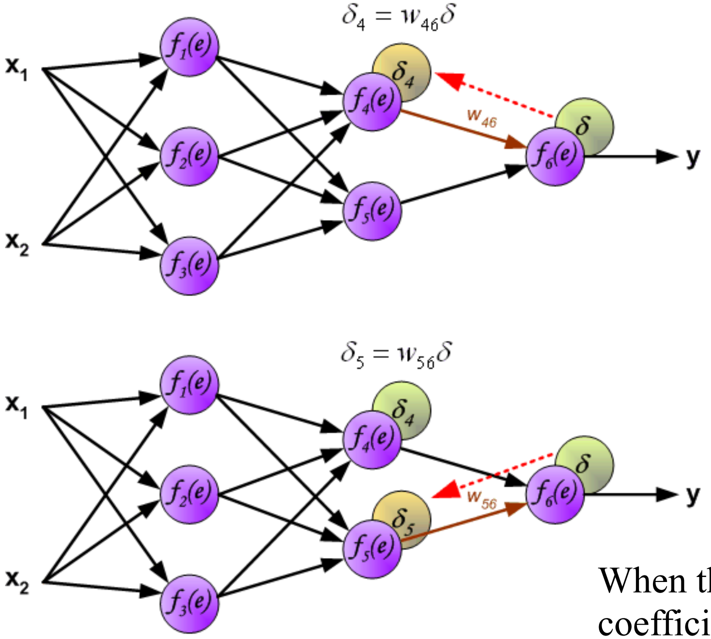
AncoraSIR.com

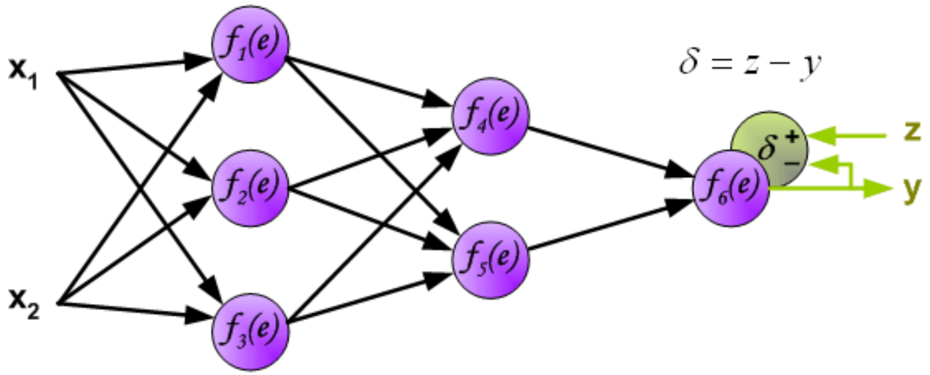
Backward Propogation

Calculate the prediction error node-by-node



The idea is to propagate error signal *d* (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.



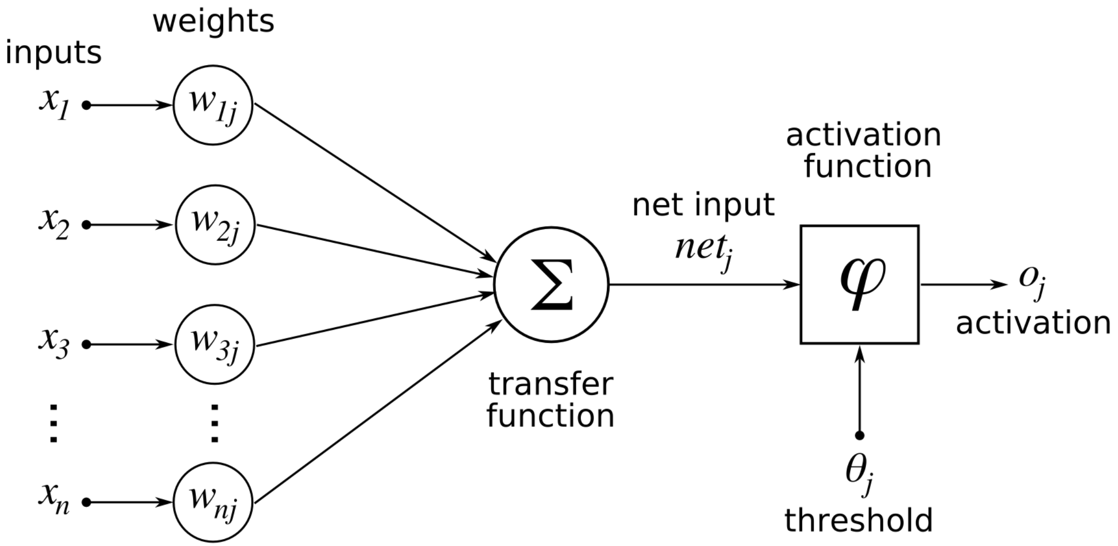


When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified.

In formulas on the right, *df(e)/de* represents derivative of neuron activation function (which weights are modified).

AncoraSIR.com

Backward Propogation



$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j$$

with

$$\delta_j = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \mathrm{net}_j} = \begin{cases} \frac{\partial L(o_j,t)}{\partial o_j} \frac{d\varphi(\mathrm{net}_j)}{d\mathrm{net}_j} & \text{if } j \text{ is an output neuron,} \\ \left(\sum_{\ell \in L} w_{j\ell} \delta_\ell\right) \frac{d\varphi(\mathrm{net}_j)}{d\mathrm{net}_j} & \text{if } j \text{ is an inner neuron.} \end{cases}$$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta o_i \delta_j$$

AncoraSIR.com

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

Exercise I

Neural Networks Playground from Google

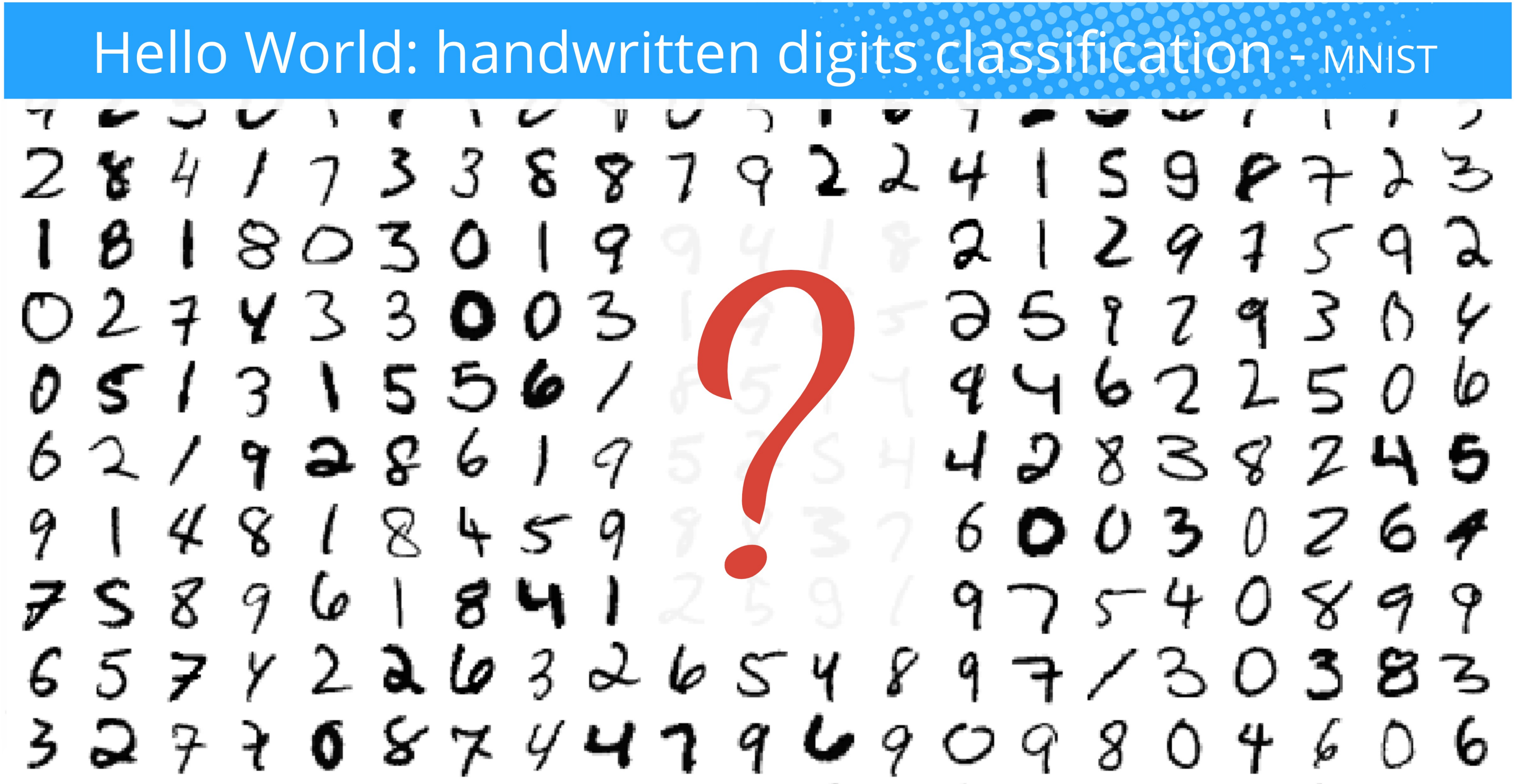
- Build your first neural networks on the website
  - https://playground.tensorflow.org/

- Play with different data types, features, network structures. Can Neural networks separate nonlinear features?

- Try different hyperparameters.

Exercise II

"Hello, MNIST!"

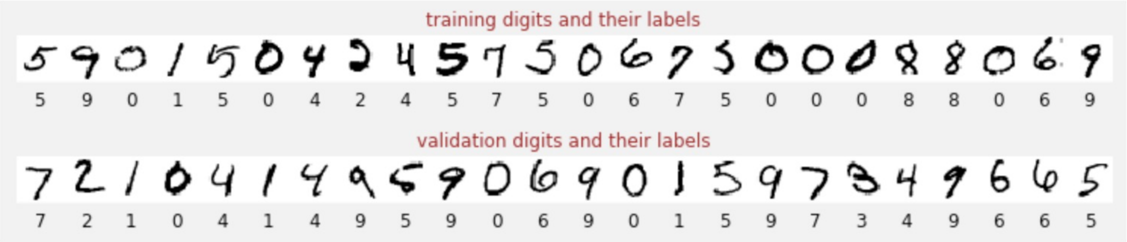


MNIST = Mixed National Institute of Standards and Technology - Download the dataset at http://yann.lecun.com/exdb/mnist/

AncoraSIR.com

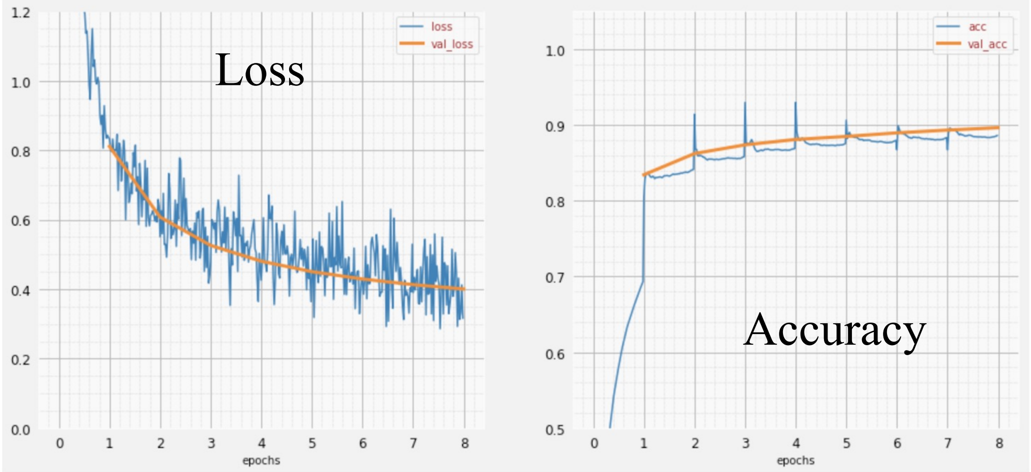
A Toy Example of Training a Neural Network

Training => Validation => Regularization => Testing => Optimization



Tensors as matrices storage of data

- [28, 28, 1]: A 28x28 pixel grayscale image (Gray)
- [128, 28, 28, 3]: A batch of 128 color images (RGB)



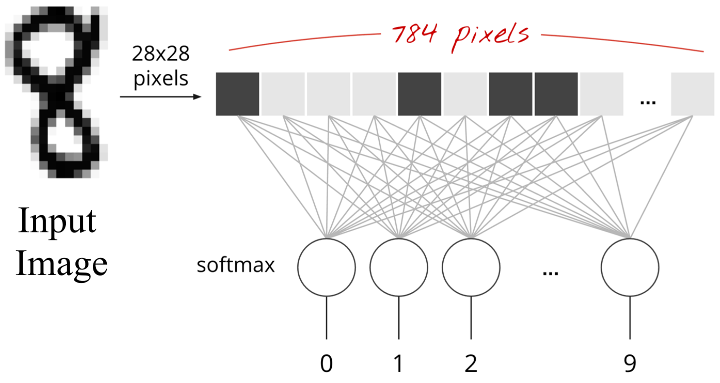

```
468/468 [==============================] - 2s 5ms/step - loss: 0.4290 - acc: 0.8862 - val_loss: 0.4003 - val_acc: 0.8963
Epoch 9/10
378/468 [=======================>......] - ETA: 0s - loss: 0.4230 - acc: 0.8866
```

60,000 training images
10,000 testing images

A Single-layer Network of Image Classification

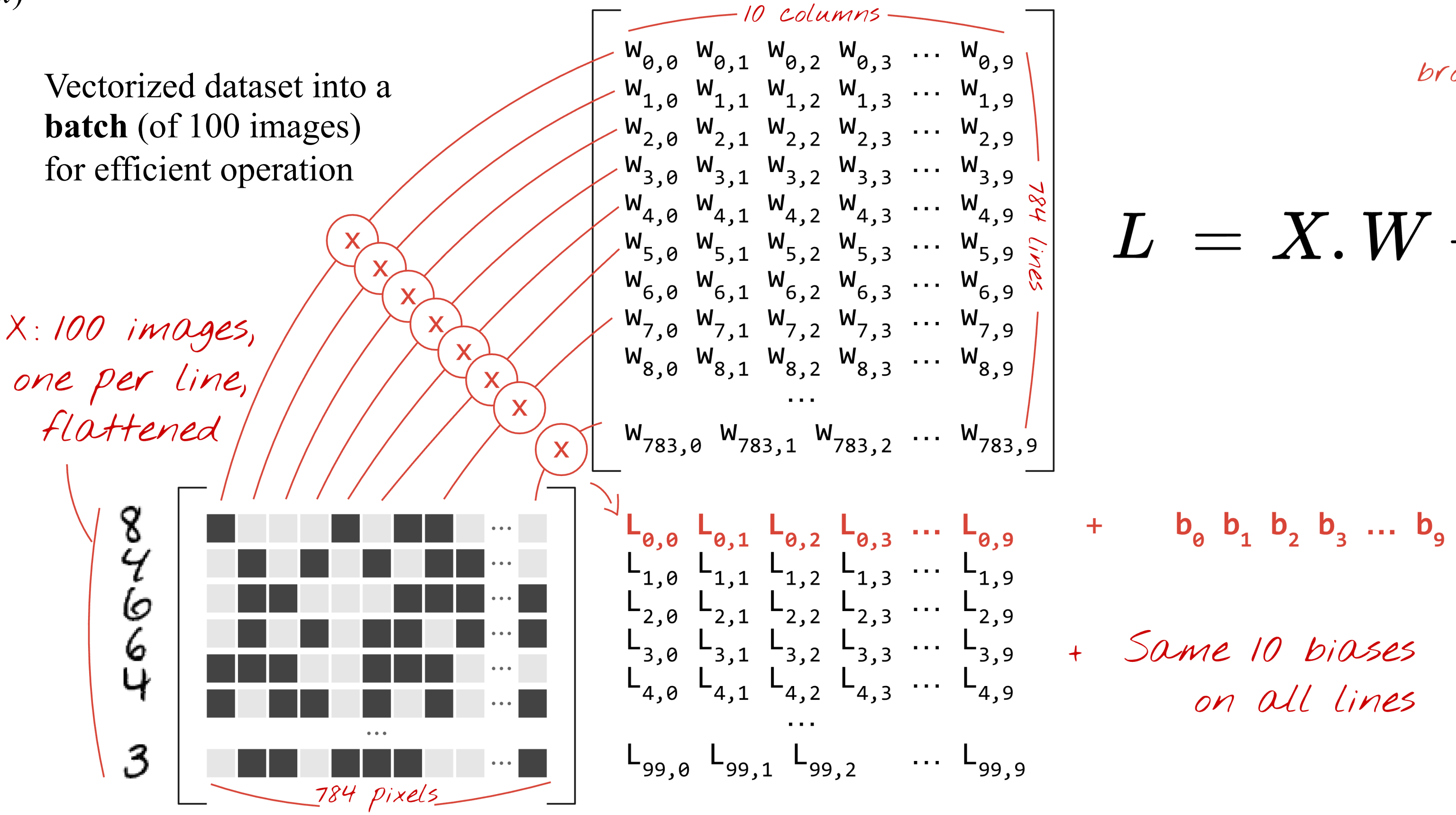
Modified National Institute of Standards and Technology database

Straightened into an array of features (x)



Through Softmax operation to predict 10 labels (y) with a normalized distrution

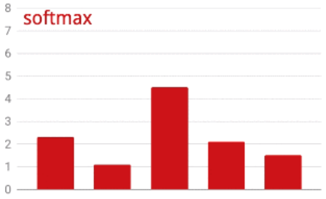
Vectorized dataset into a **batch** (of 100 images) for efficient operation



broadcast

$$L = X.W + b$$

weighted sum of all pixels + bias



$$softmax(L_n) = \frac{e^{L_n}}{||e^L||}$$

neuron outputs

AncoraSIR.com

SUSTech Southern University of Science and Technology

Softmax on a Batch of Images

Output with a distribution of computed probabilities

Predictions $Y[100, 10]$ · Images $X[100, 784]$ · Weights $W[784,10]$ · Biases $b[10]$

$$Y = softmax(X.W + b)$$

applied line by line · matrix multiply · broadcast on all lines

tensor shapes in [ ]

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

actual probabilities, "one-hot" encoded

$$-\sum Y_i'.log(Y_i)$$ Cross entropy

this is a "6"

computed probabilities

0	1	2	3	4	5	6	7	8	9
.01	.03	.00	.04	.03	.05	0.8	.02	.01	.01

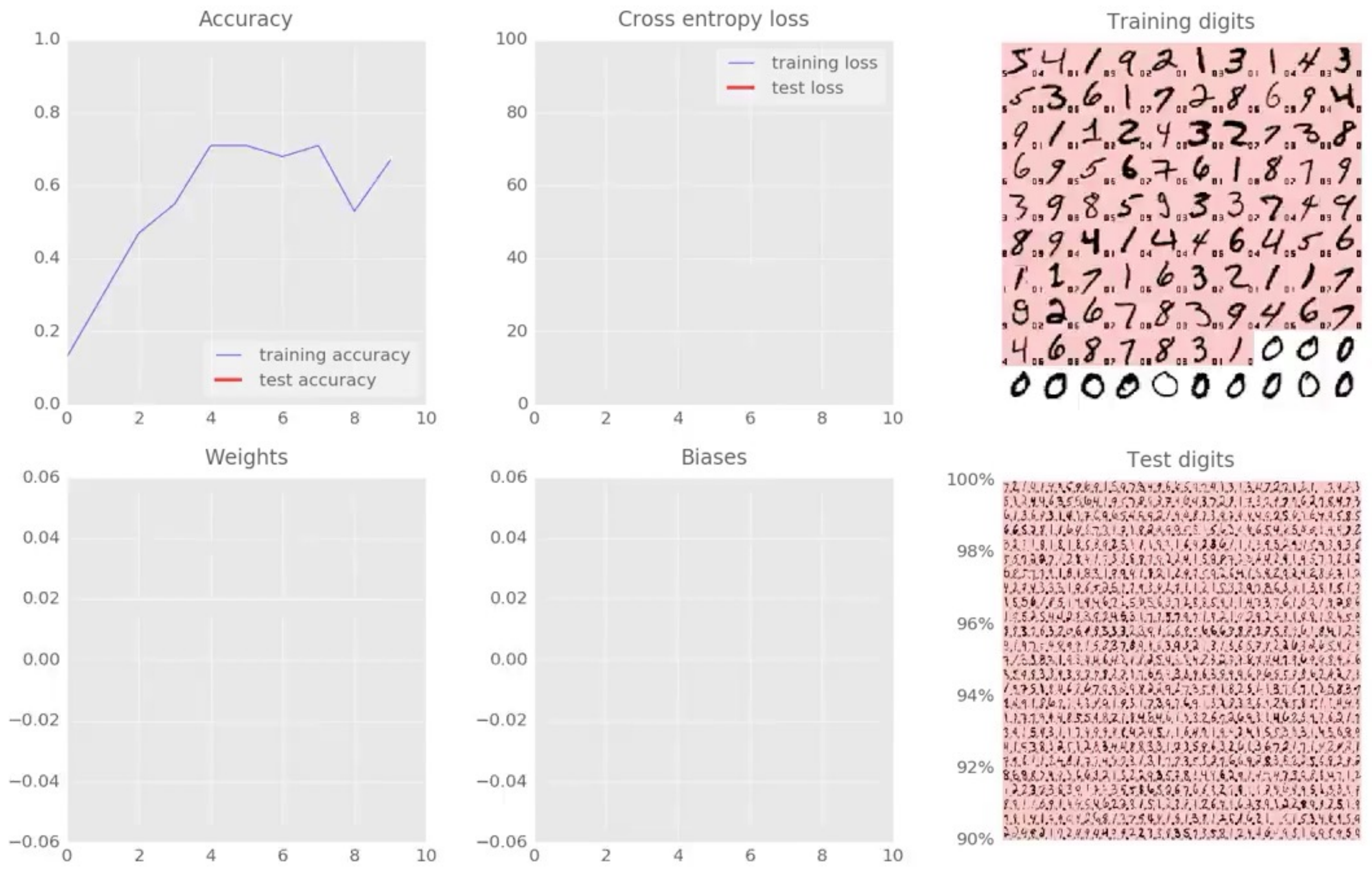
tensor shapes: $X[100, 784]$ $W[748,10]$ $b[10]$

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

matrix multiply · broadcast on all lines

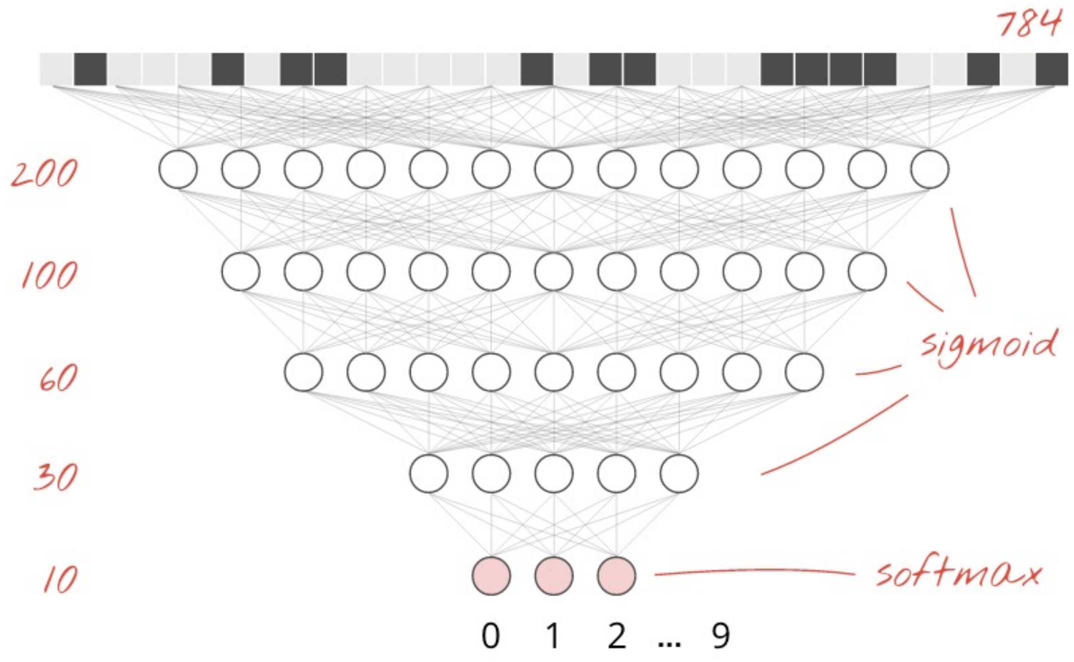
https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/blob/master/tensorflow-mnist-tutorial/keras_01_mnist.ipynb

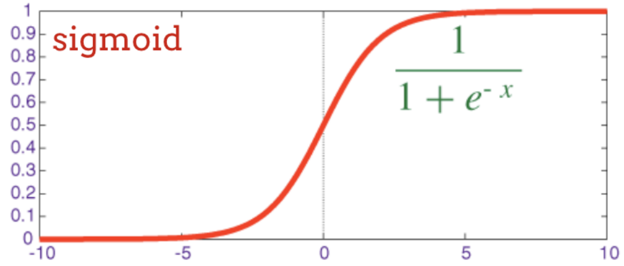
SUSTech Southern University of Science and Technology


Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits
100%
98%
96%
94%
92%
90%

Adding Layers

Simply adding more layers with sigmoid activations does not give us the expected results …

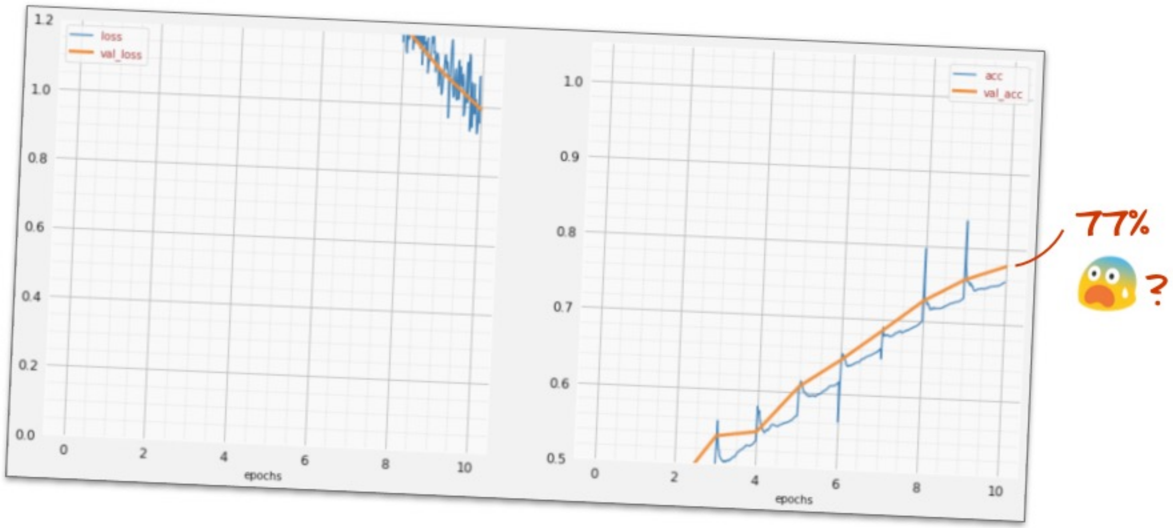




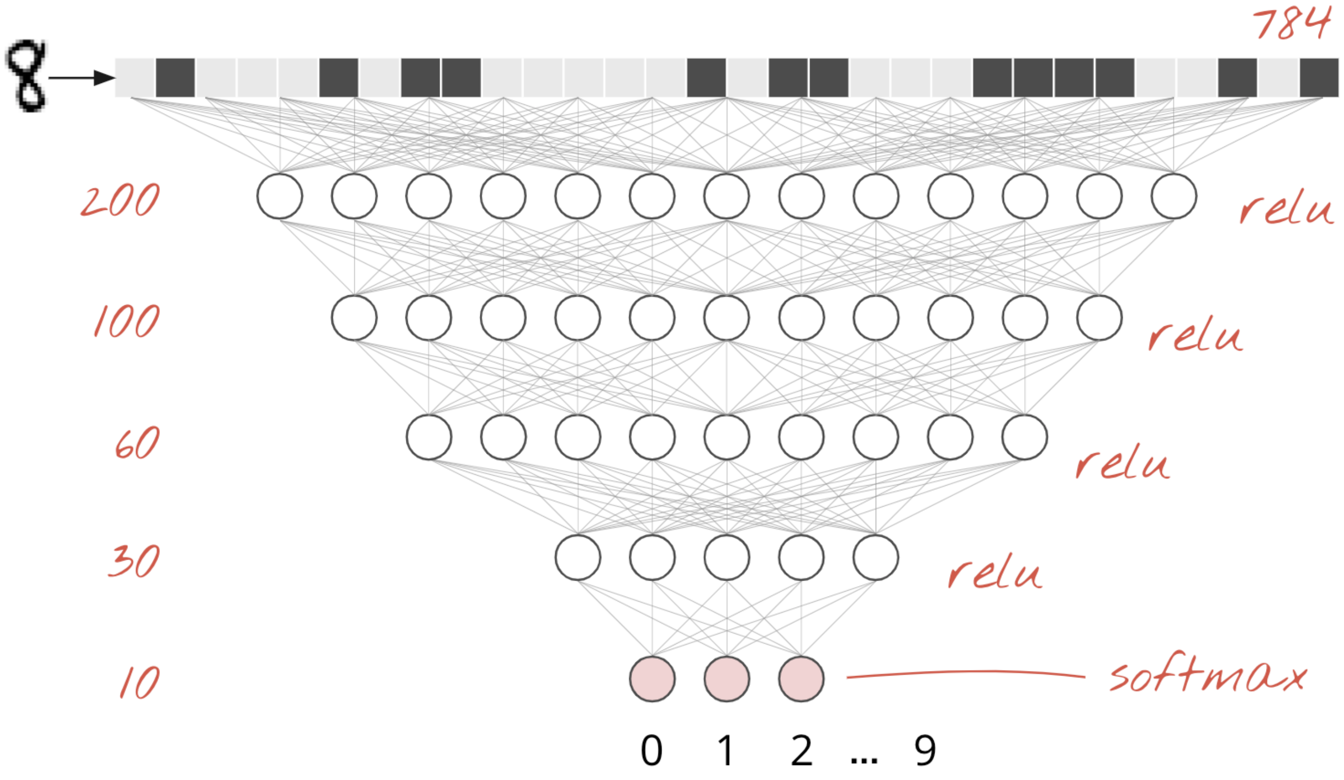
AncoraSIR.com

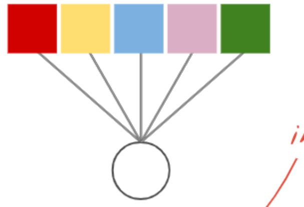
Getting flat

- The gradient can become very small and training get slower and slower.



Special Care for Deep Networks

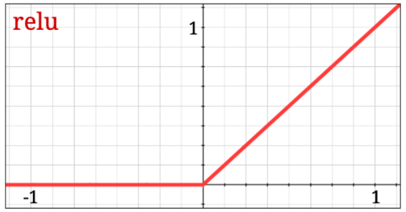


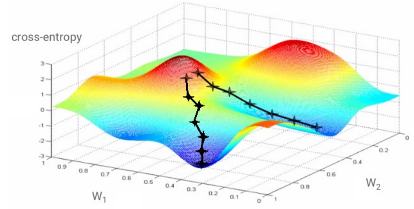


inputs

$$Y = \mathbf{relu}\left(\sum_i \mathbf{W_i} X_i + \mathbf{b}\right)$$

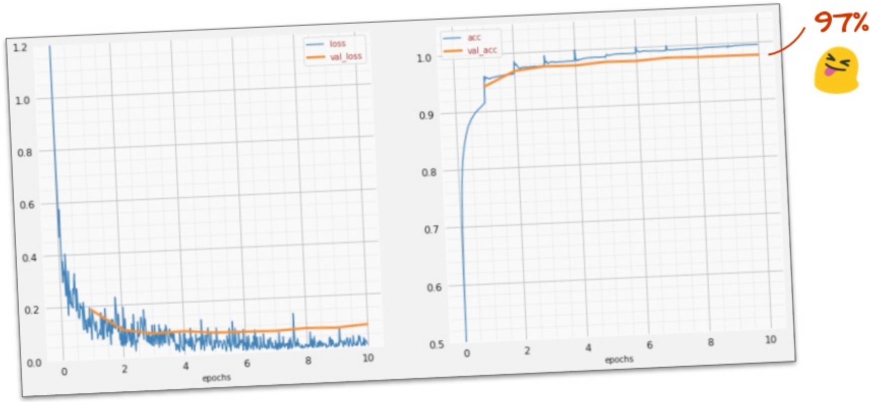
activation weights bias





HANDS ON:

Replace the `'sgd'` optimizer with a better one, for example `'adam'` and train again.



HANDS ON:

Replace all `activation='sigmoid'` with `activation='relu'` in your layers and train again.

AncoraSIR.com

https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/blob/master/tensorflow-mnist-tutorial/keras_02_mnist_dense.ipynb

SUSTech Southern University of Science and Technology

Thank you~

Wan Fang
Southern University of Science and Technology