Linear Networks

Lecture 3

Subir Varma

Today's Lecture

Linear Classification Systems – Logistic Regression

- Supervised Learning
- Loss Functions
- Classification with Two Classes
- Classification with K Classes

Recap of Lecture 2

The job of the neural network is to compute P(Y|X)

X

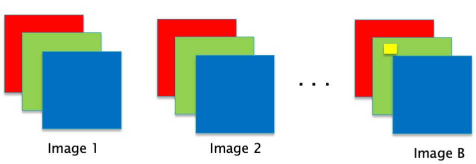
Neural Network

$P(Y = cat|X) = 0.8$

$P(Y = dog|X) = 0.1$ → Sampling → cat

$P(Y = car|X) = 0.1$

Input: 4D Tensor



Output: Probability Density

Example of a 4D tensor:
A sample of L color images
(sample #, channels, height, width)

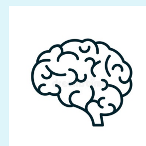
Supervised Learning: The Classification Problem

Labels

"cat"

Input Tensor

X → Ground Truth or Label → Label $= T$



Categorical
Label = 001

Sparse Categorical
Label = 2

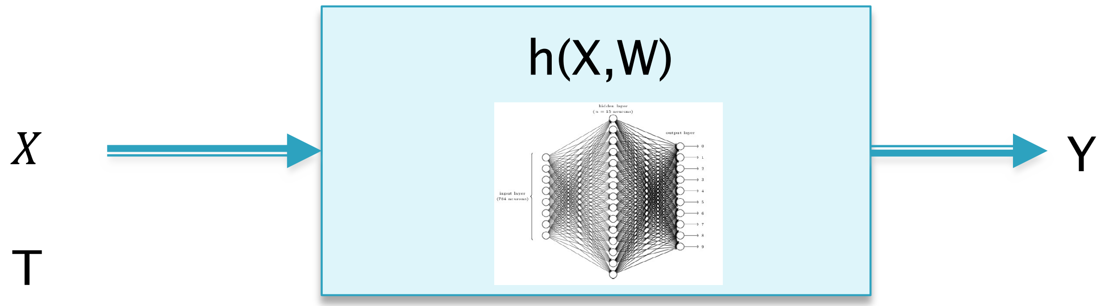
Application of the input tensor X results in the label T

$\{1, 2, \ldots, K\}$ → Sparse Categorical Labels (Integers)

$(t_1, t_2, \ldots, t_K)$ → Categorical Labels (1-Hot Encoded Labels)

The K categories correspond to the K unit vectors
$(1, 0, 0, \ldots, 0)$ to $(0, 0, 0, \ldots, 1)$

The Supervised Learning Problem



X(1) → T(1)
X(2) → T(2)
..
X(M) → T(M)

Application of the input tensor X(s) results in the label T(s), and we observe M such input-output pairs

Training Set

Problem: Find a model h(X,W) for the Unknown System, such that it is able to Predict "suitably good" values for T, for new values of X.

Test Set

Solution in Two Steps

- Step 1

Come up with the structure for the classifier h(X,W) with unknown parameters W

– An educated guess!

- Step 2

Iteratively estimate the unknown parameters W from the labeled Training Data (X(s),T(s)), s = 1,..M

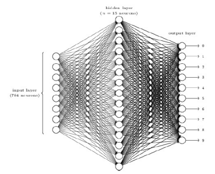
– Known as Training or Learning

Probabilistic Classification

Label $= T \in \{1,2,\ldots,K\}$

X

$h(X, W)$



$P(T=1|X) = y_1 = h_1(X, W)$

$P(T=K|X) = y_K = h_K(X, W)$

Output is a Discrete Probability Density Function

$$y_k = h_k(X, W) = \mathrm{P}(\mathrm{Y} = \mathrm{k}|\mathrm{X})$$

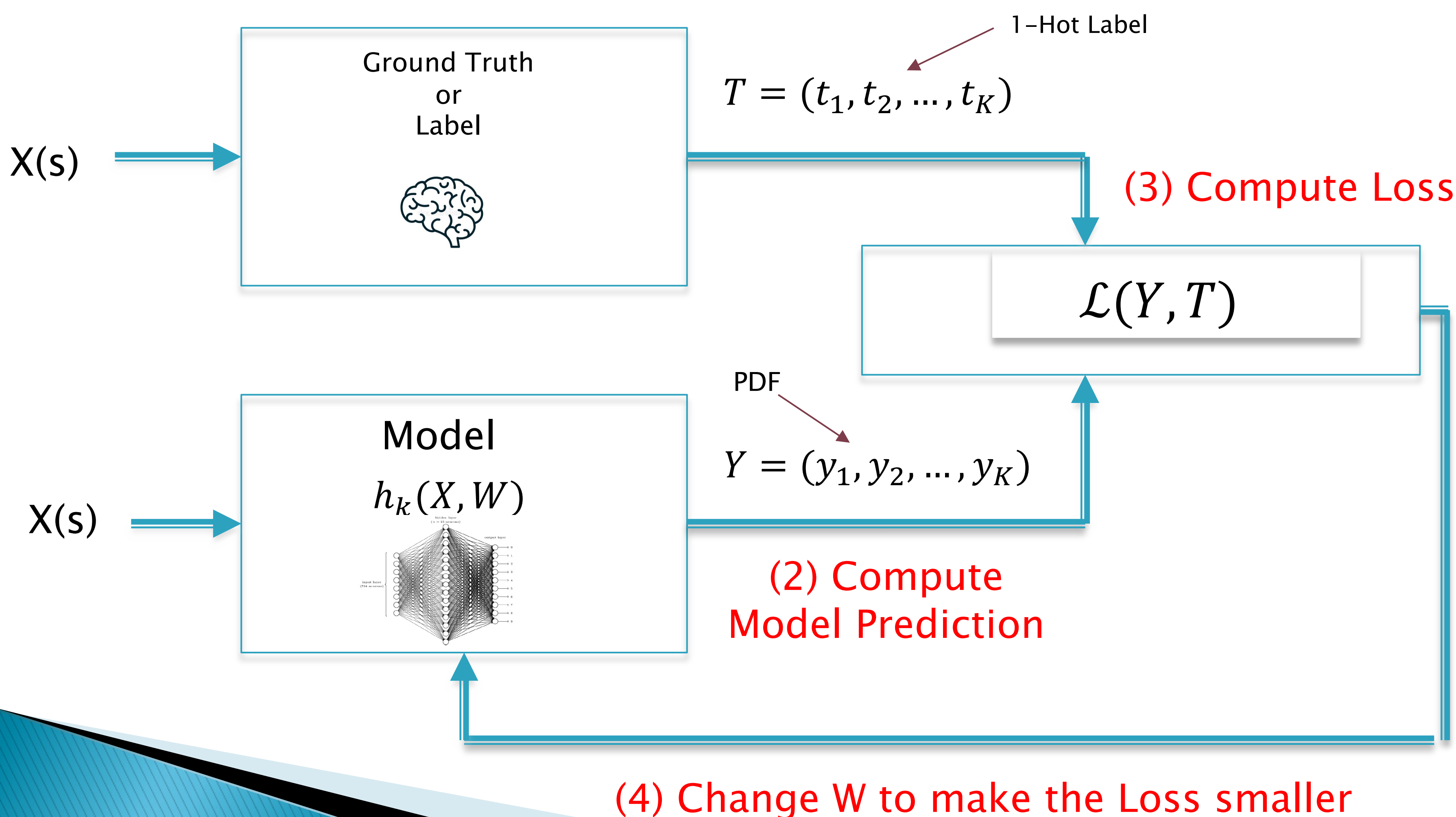
$$\sum_{k=1}^{K} y_k = 1$$

Solution Strategy

(0) Collect Labeled Data

(1) Choose Model $h_k(X,W)$

We have reduced the problem of Model Synthesis to an Optimization Problem !!



Loss Functions

Choice of Loss Functions

Loss Function for a single sample

1. Mean Square Error (MSE)

$$\mathcal{L}(s) = \frac{1}{K}\sum_{k=1}^{K}[y_k(s) - t_k(s)]^2$$

2. Mean Absolute Error (MAE)

$$\mathcal{L}(s) = \frac{1}{K}\sum_{k=1}^{K}|y_k(s) - t_k(s)|$$

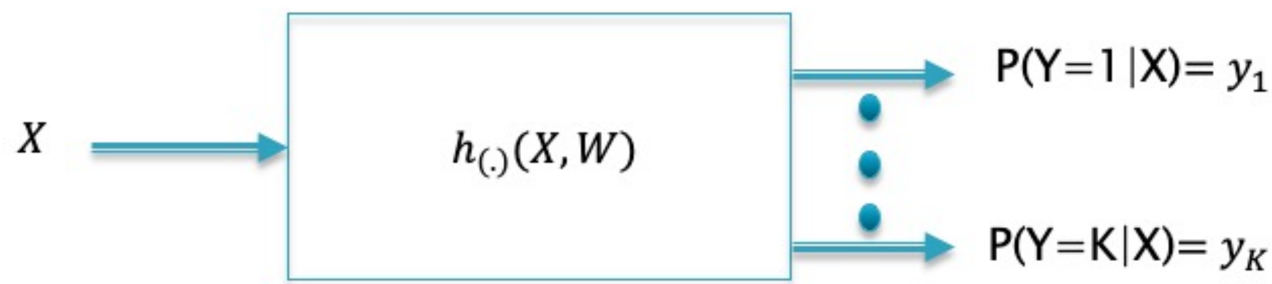
$$L = \frac{1}{M}\sum_{s=1}^{M}\mathcal{L}(s)$$

Loss for the entire Dataset

Used in Regression Problems

Cross Entropy Loss Function

Label $= T = (t_1, t_2, \ldots, t_K)$



$$\sum_{k=1}^{K} y_k = 1$$

$$\mathcal{L}(s) = -\sum_{k=1}^{K} t_k\,(s) \log y_k(s)$$

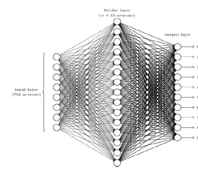
Used in Classification Problems

$$L = \frac{1}{M} \sum_{s=1}^{M} \mathcal{L}(s)$$

Formula Derived using Maximum Likelihood Estimation Theory

Example: K = 2 (Binary Cross Entropy Loss)

X → $h(X, W)$

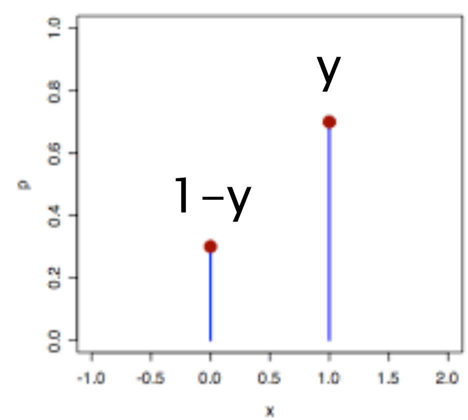


→ Single output

P(T=1|X) = y

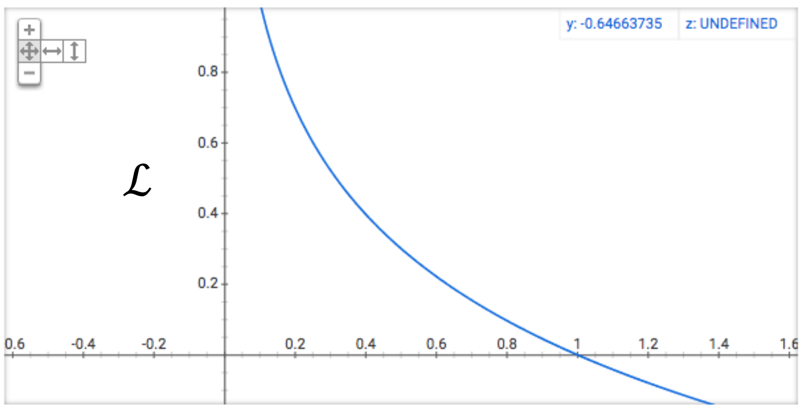
Single Label: t
(1,0) and (0,1)

$$\mathcal{L} = -[t \log y + (1-t)\log(1-y)]$$



The Cross Entropy Loss

$$\mathcal{L} = -[t \log y + (1-t)\log(1-y)]$$



y_q

$$t_q = 1$$

$$\mathcal{L} = -\log y_q,\ 0 \le y_q \le 1$$

Exact Match

$y_q = 1$

$\mathcal{L} = 0$

Complete Mismatch

$y_q = 0$

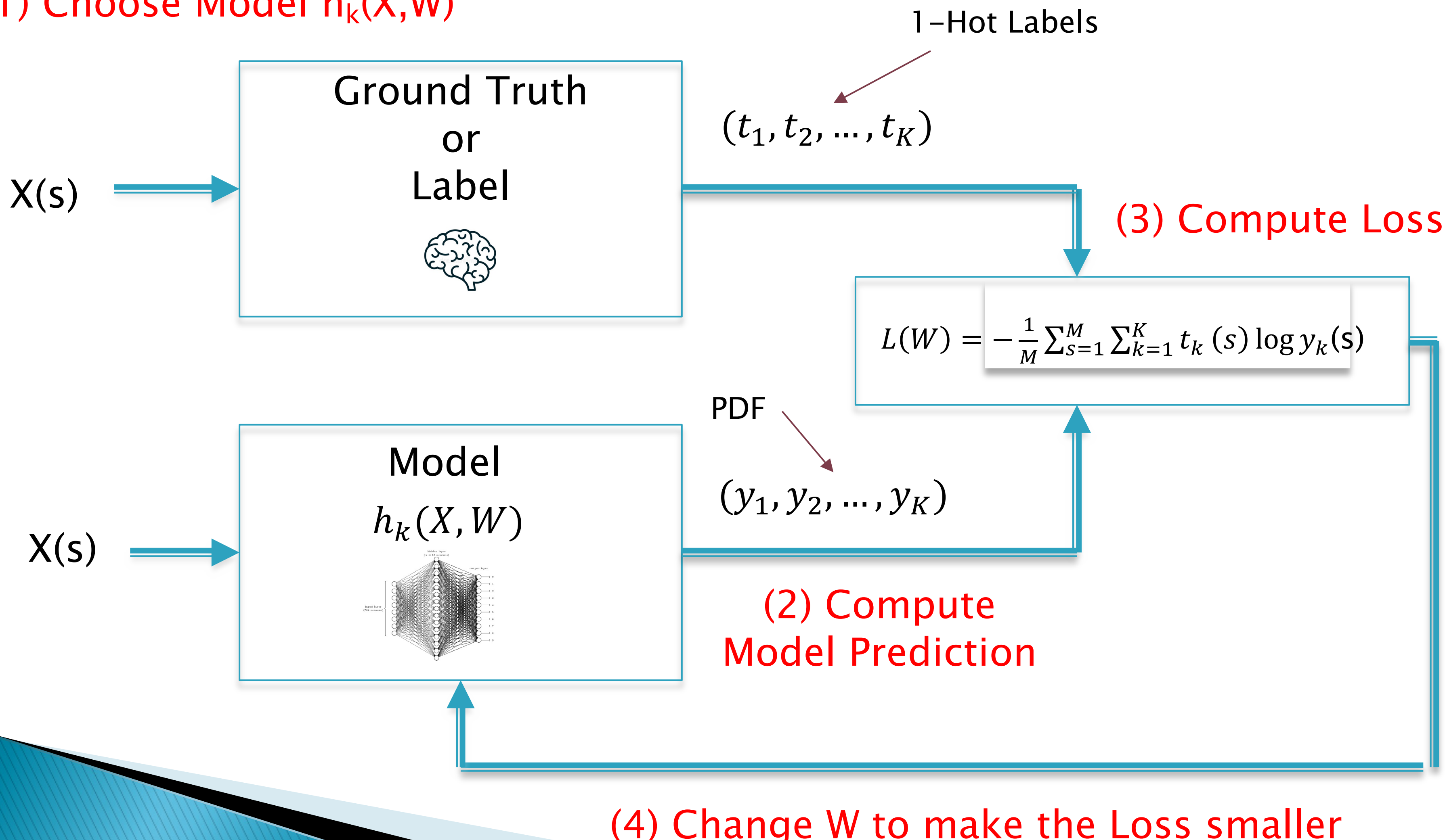
$\mathcal{L} = \infty$

Exercise: Plot the graph for t = 0

Solution to Classification Problem

(0) Collect Labeled Data (X(s),T(s))

(1) Choose Model h_k(X,W)



In Rest of Course

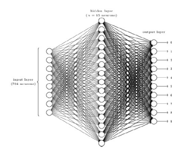
Discover increasingly sophisticated models $h_k(X,W)$

- Start with the simplest: Linear Models (Logistic Regression)
- Add Hidden Layers – Dense Feed Forward Networks
- Add Local Filtering – Convolutional Neural Networks (CNNs or ConvNets)
- Add Time Dependence – Recurrent Neural Networks (RNNs, LSTMs)
- Add Attention – Transformers

Linear Classification Models with Two Classes

Probabilistic Classification with K = 2

$h_{(.)}(X, W)$



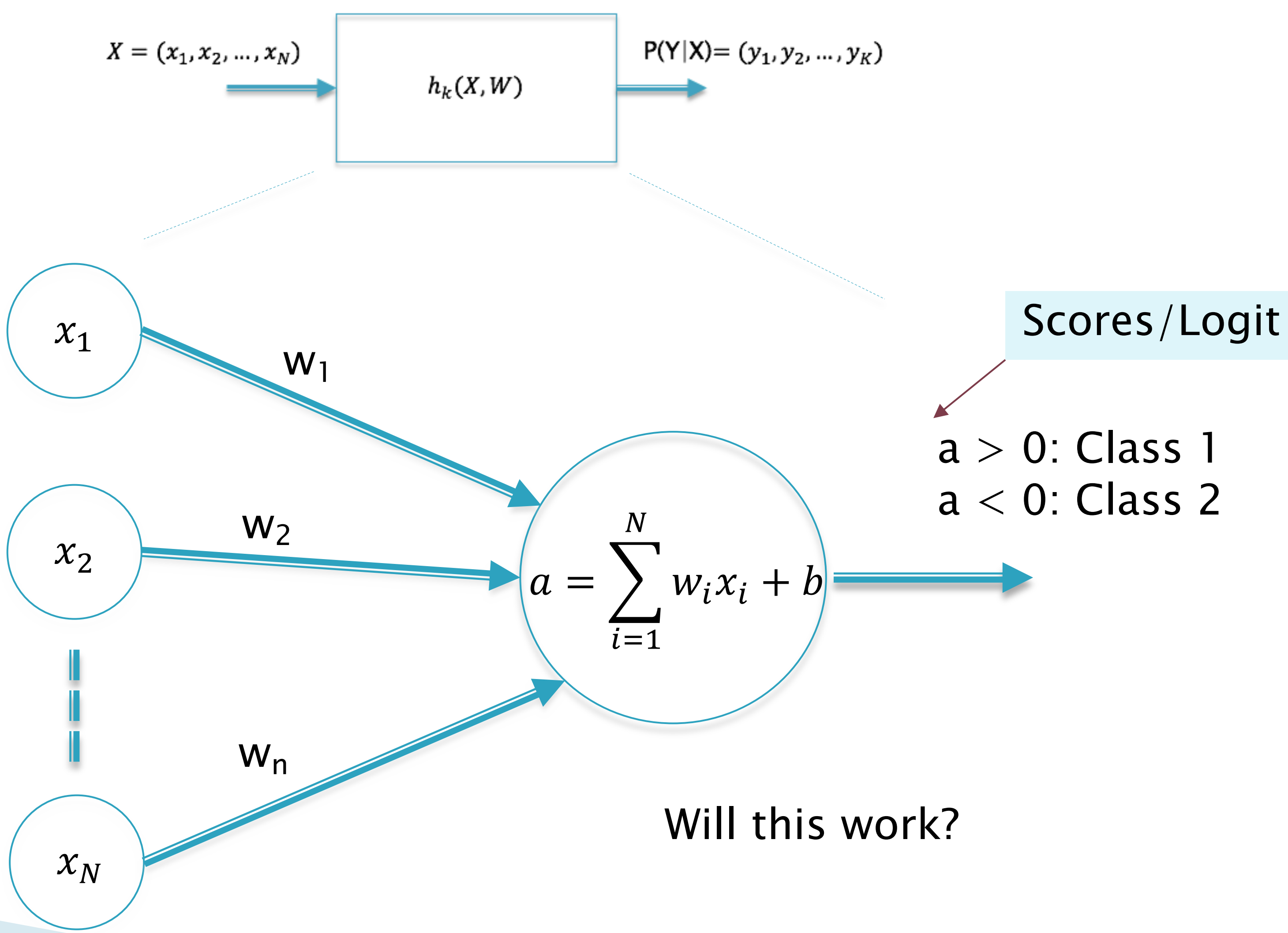
X

P(T=(1,0)|X)= y

Label $= T = (t, 1 - t)$

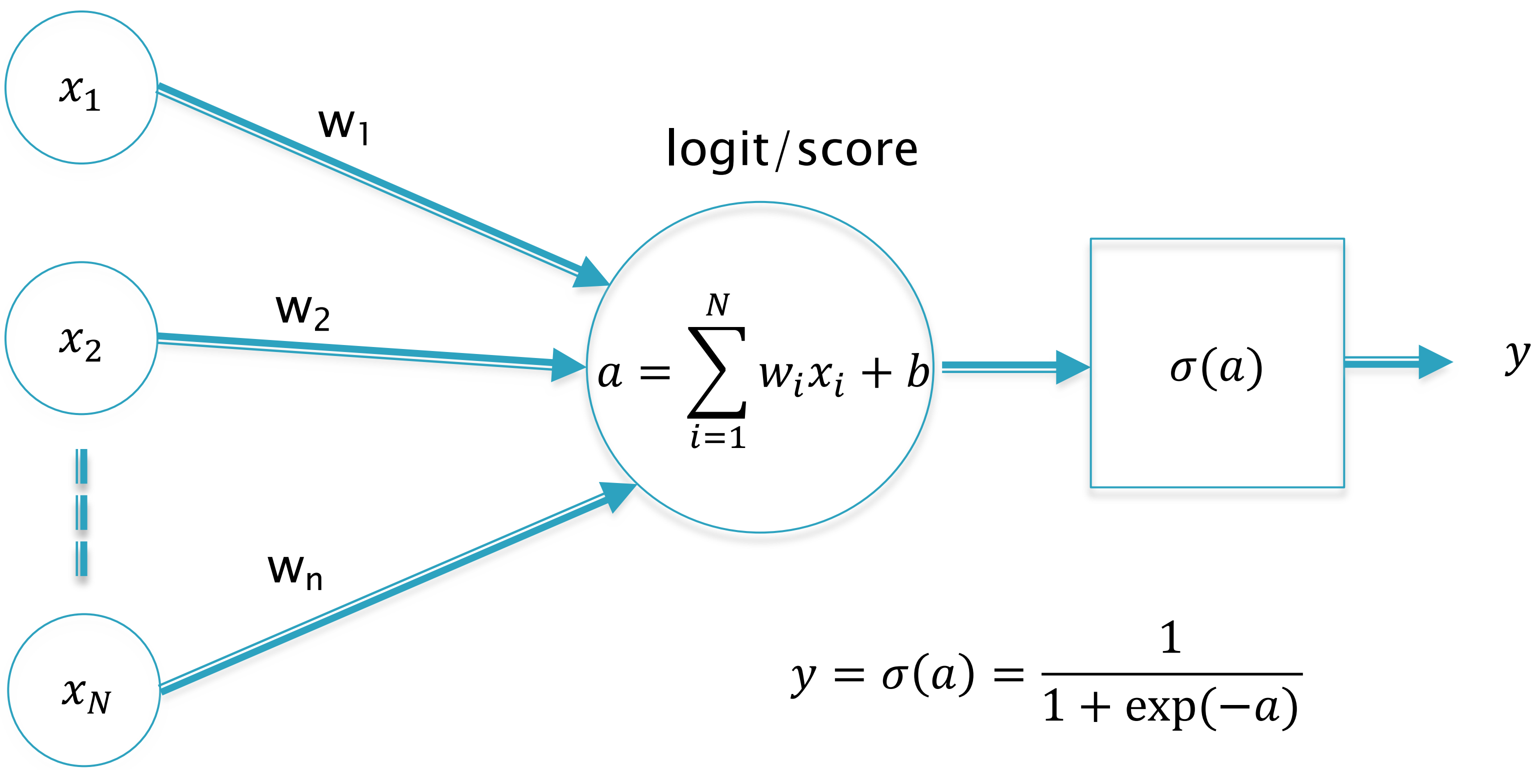
P(T=(0,1)|X)= $1 - y$

Linear Models: Logistic Regression



Will this work?

Convert Scores to Probabilities via the Logistic Sigmoid Function



$$y = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

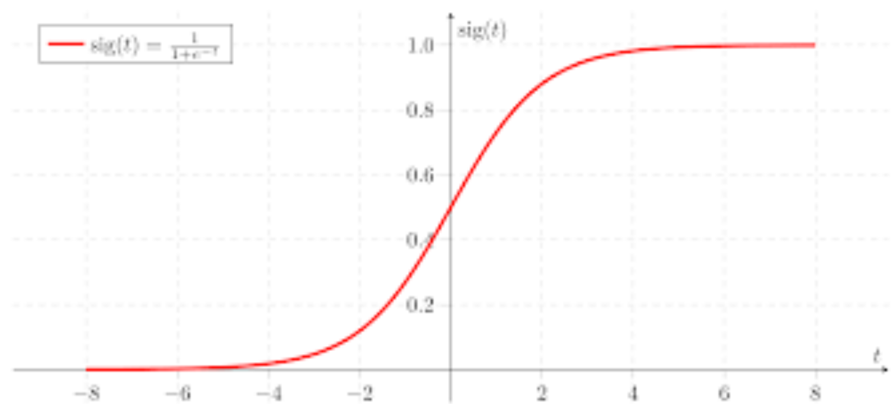
$$1 - y = 1 - \sigma(a) = \frac{\exp(-a)}{1 + \exp(-a)}$$

The Logistic Sigmoid Function

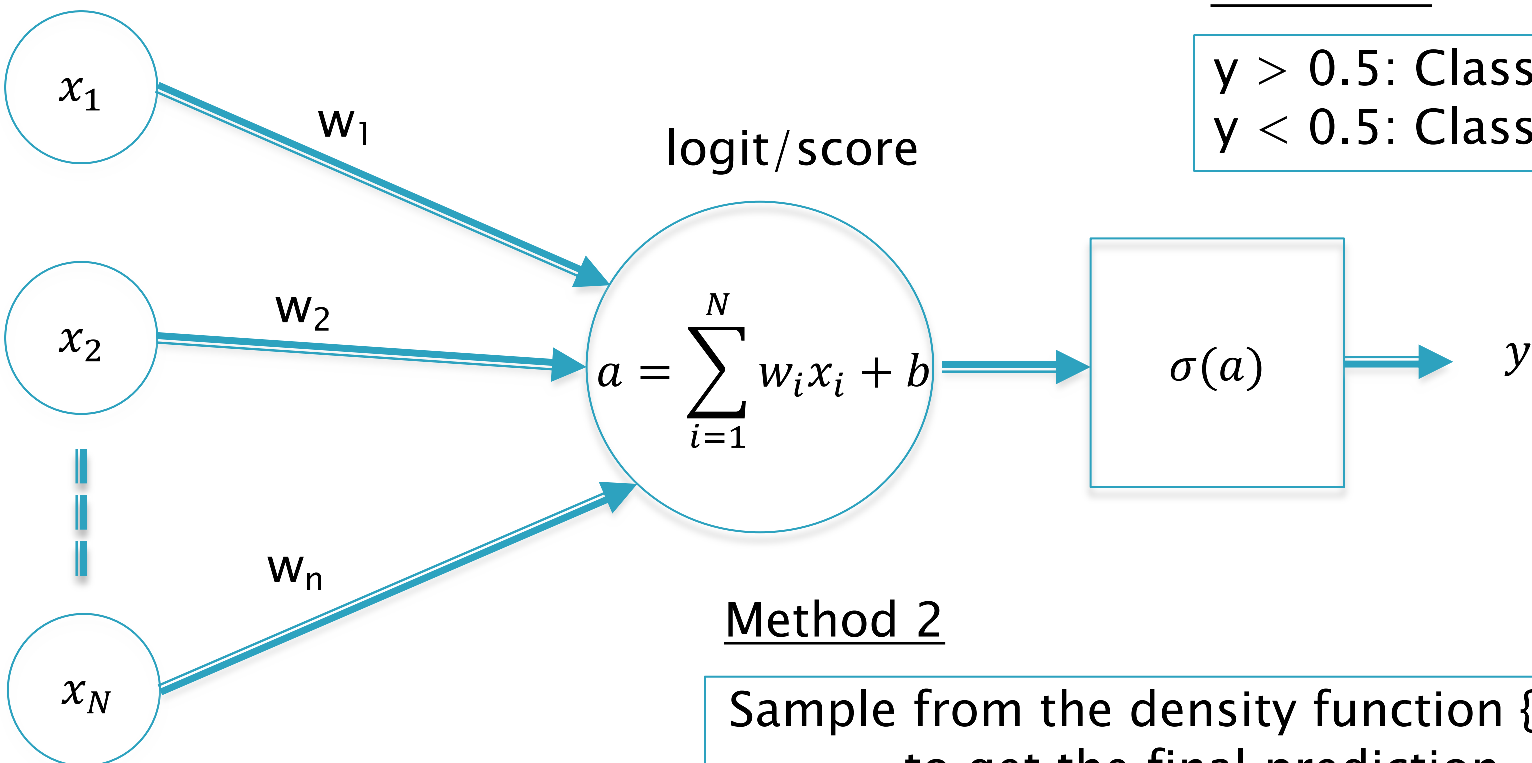
$$y = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

Takes a number and "squashes" it so that it is between 0 and 1

- Appropriate as the output function for networks with binary decision outputs.



Prediction



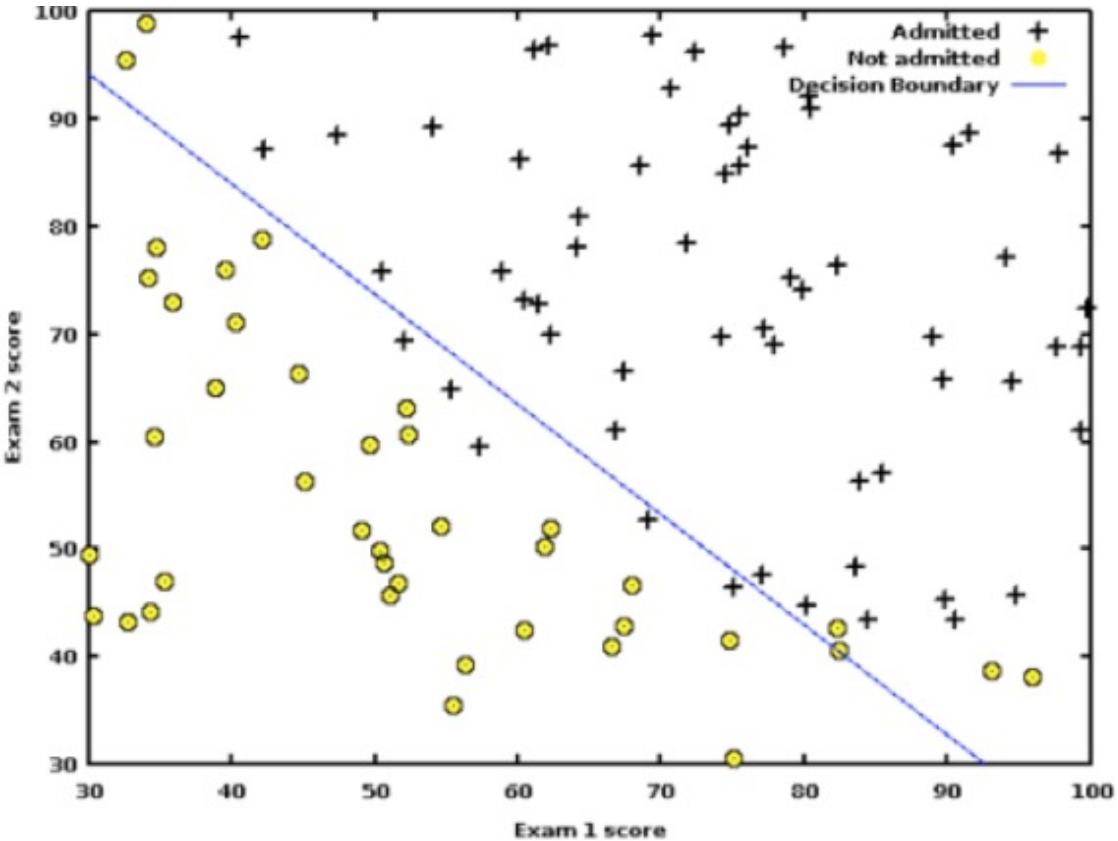
<u>Method 1</u>

y > 0.5: Class 1
y < 0.5: Class 2

<u>Method 2</u>

Sample from the density function {y, 1−y} to get the final prediction

Linear Models: Logistic Regression



How to Find the Weights?

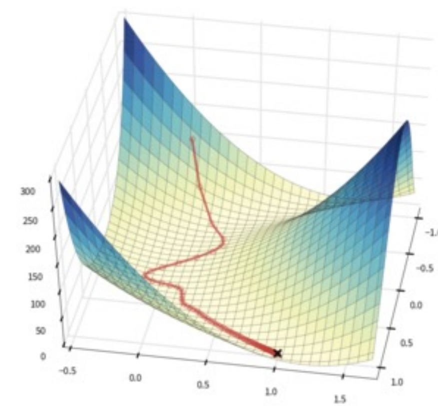
Given training samples
(X(s),T(s)), s = 1,…,M

Find weights that minimize the Loss Function

$$L(W) = -\frac{1}{M}\sum_{s=1}^{M}[t(s)\log y(s) + (1 - t(s))\log(1 - y(s))]$$

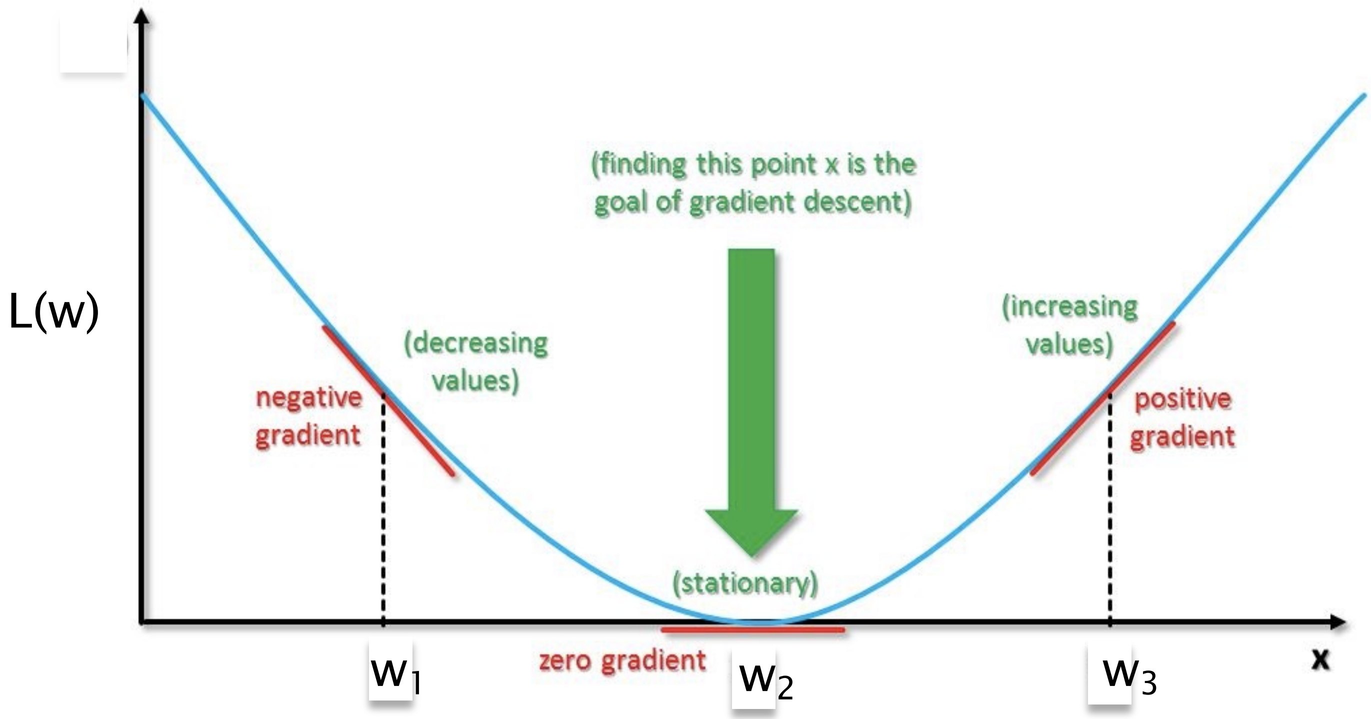
where

$$y(s) = \frac{1}{1 + \exp(-\sum_{i=1}^{N} w_i x_i(s) - b)}$$



No Closed Form solution
Will have to use Numerical Methods

Minimization using Gradient Descent



$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

Learning Rate

Gradient Computation

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

where

$$L(W) = -\frac{1}{M}\sum_{s=1}^{M}[t(s)\log y(s) + (1 - t(s))\log(1 - \mathrm{y}(\mathrm{s}))]$$

Where $y(s) = \frac{1}{1+\exp(-\sum_{i=1}^{N} w_i x_i(s) - b)}$

Gradient Computation

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

Loss for a single sample

$$L = \frac{1}{M}\sum_{s=1}^{M} \mathcal{L}(s), where \quad \mathcal{L}(s) = -[t(s)\log y(s) + (1 - t(s))\log(1 - y(s))]$$

$$\frac{\partial L}{\partial w} = \frac{1}{M}\sum_{s=1}^{M} \frac{\partial \mathcal{L}(s)}{\partial w}$$

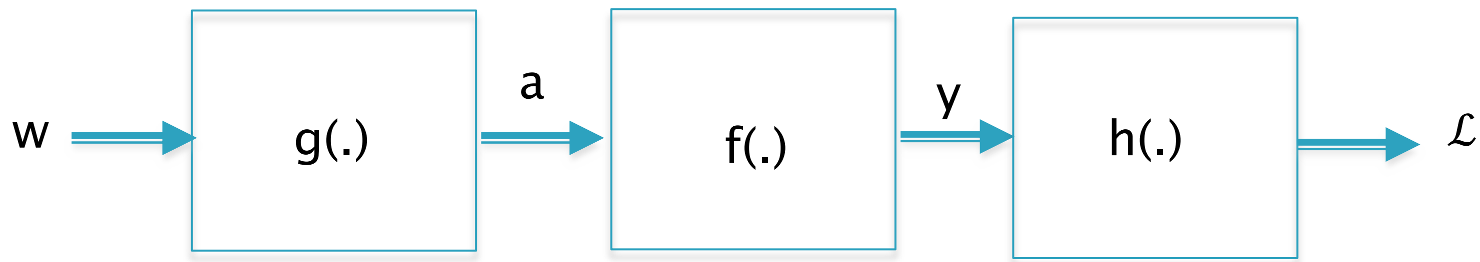
Gradient for a single sample

Gradient Computation

$a = g(w)$

$y = f(a) = f(g(w))$

$$\mathcal{L} = h(y) = h(f(a)) = h(f(g(w))) = q(w)$$



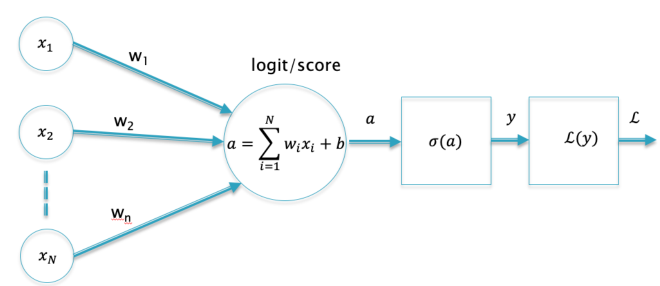
Problem: Evaluate $\frac{\partial \mathcal{L}}{\partial w}$

$$\mathcal{L}(s) = -[t(s)\log y(s) + (1 - t(s))\log(1 - y(s))]$$

$$y(s) = \frac{1}{1 + \exp(-a)}$$

$$a = -\sum_{i=1}^{N} w_i x_i\,(s) - b$$

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial a}\frac{\partial a}{\partial x} = h'(y)\ f'(a)\ g'(w)$$



Gradient Computation

$$\mathcal{L} = -[t \log y + (1-t)\log(1-\mathrm{y})]$$

$$y = \frac{1}{1+e^{-a}}, \qquad \mathrm{a} = \sum_{i=1}^{n} w_i x_i + b$$

Use Chain Rule of Derivatives: $$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial a}\frac{\partial a}{\partial w_i}$$

$$\frac{y-t}{y(1-y)} \qquad y(1-y) \qquad x_i$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = (y-t)x_i$$

Back to Gradient Descent

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

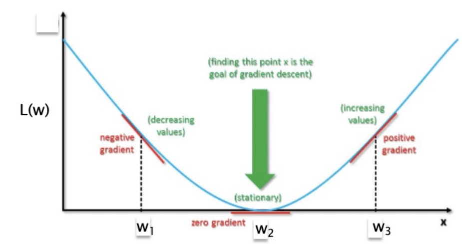
Loss for a single sample

$$L = \frac{1}{M}\sum_{s=1}^{M} \mathcal{L}(s), where \quad \mathcal{L}(s) = -[t(s)\log y(s) + (1 - t(s))\log(1 - y(s))]$$

$$\frac{\partial L}{\partial w} = \frac{1}{M}\sum_{s=1}^{M} \frac{\partial \mathcal{L}(s)}{\partial w} = \frac{1}{M}\sum_{s=1}^{M} x_i(s)[y(s) - t(s)]$$

So that ..

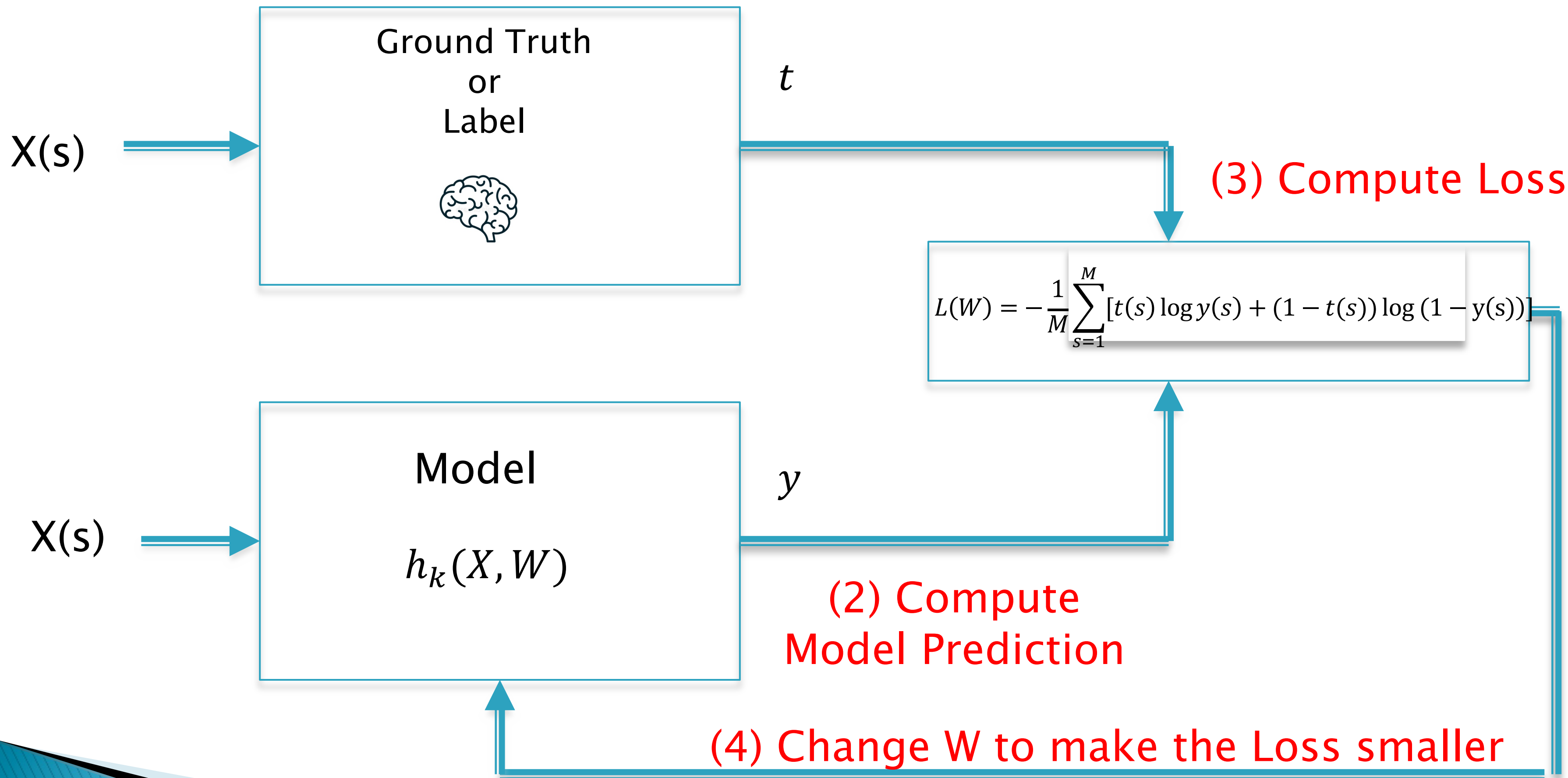
$$w \leftarrow w - \frac{\eta}{M}\sum_{s=1}^{M} x_i(s)[y(s) - t(s)]$$



Solution to Classification Problem

(0) Collect Labeled Data (X(s),T(s))

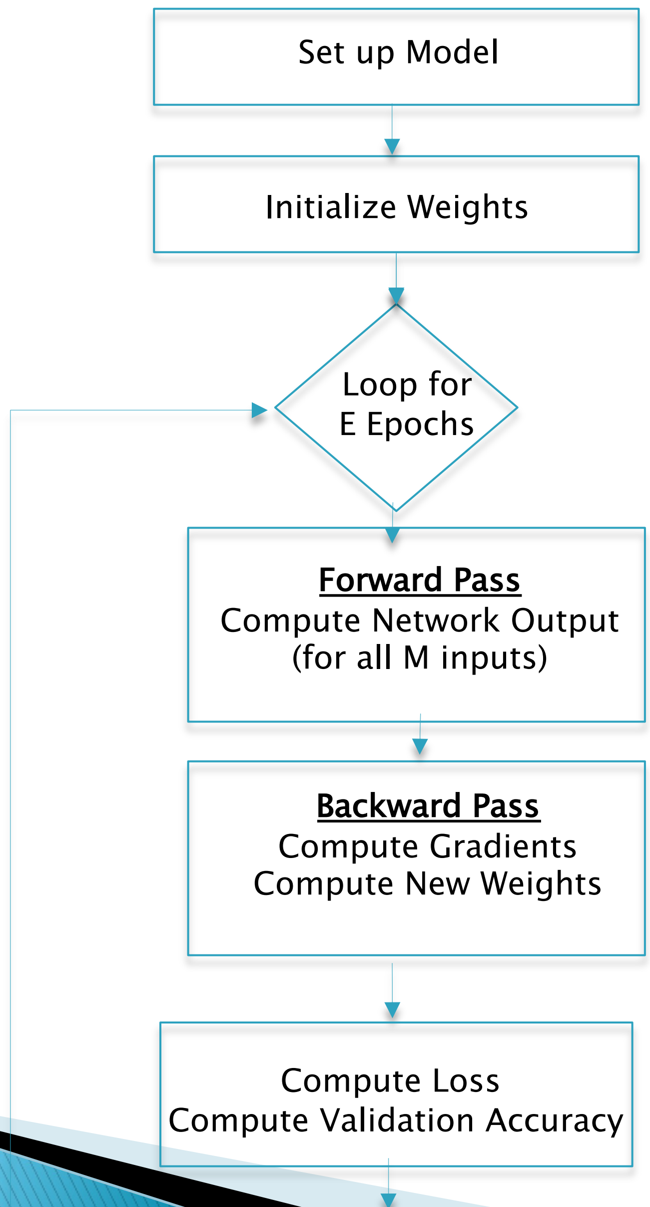
(1) Choose Model $h_k(X,W)$

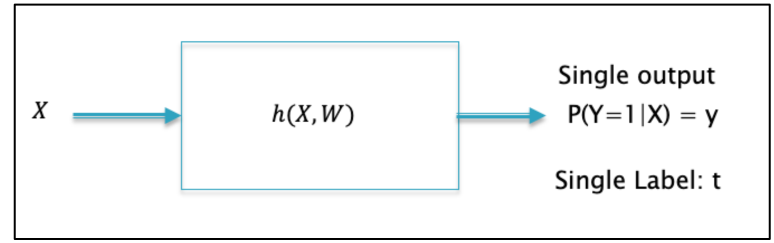


$$L(W) = -\frac{1}{M}\sum_{s=1}^{M}[t(s)\log y(s) + (1-t(s))\log(1-y(s))]$$

$$w \leftarrow w - \frac{\eta}{M}\sum_{s=1}^{M} x_i(s)[y(s)-t(s)]$$

Training Algorithm





$$y(s) = \sigma(a(s)) = \frac{1}{1 + \exp(-a)}$$

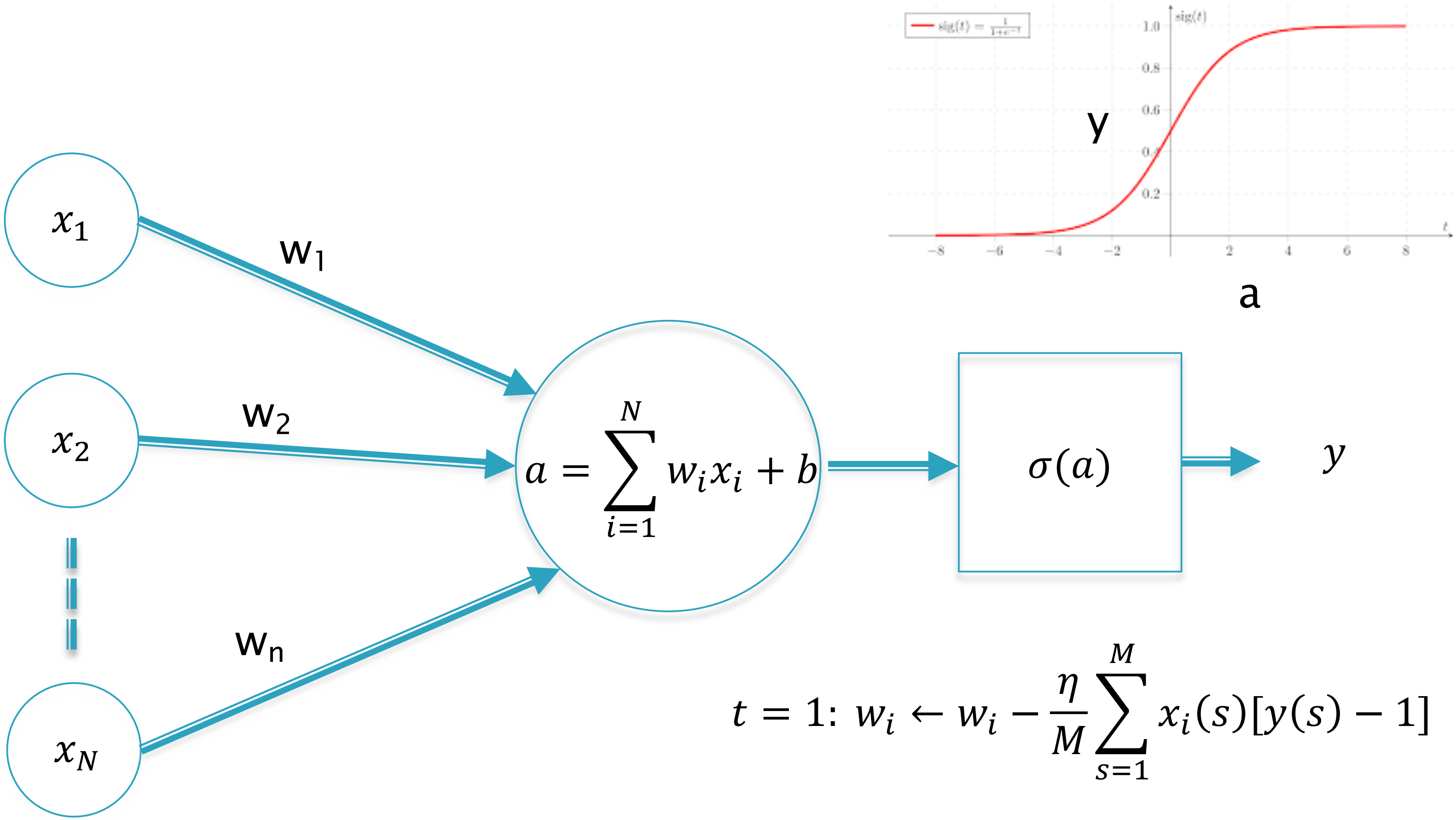
$$a(s) = \sum_{i=1}^{N} w_i x_i(s) + b$$

s = 1,…M

$$w_i \leftarrow w_i - \frac{\eta}{M} \sum_{s=1}^{M} x_i(s)[y(s) - t(s)]$$

$$L(W) = -\frac{1}{V} \sum_{s=1}^{V} [t(s) \log y(s) + (1 - t(s) \log (1 - y(s))]$$

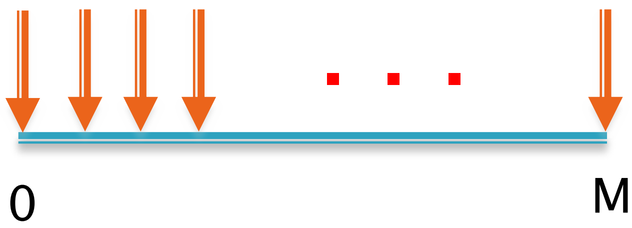
Optimization Process



$$t = 1:\ w_i \leftarrow w_i - \frac{\eta}{M} \sum_{s=1}^{M} x_i(s)[y(s) - 1]$$

$$t = 0:\ w_i \leftarrow w_i - \frac{\eta}{M} \sum_{s=1}^{M} x_i(s) y(s)$$

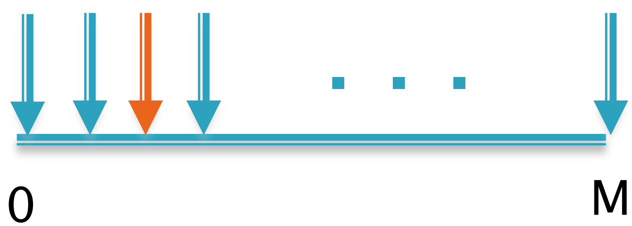
Stochastic Gradient Descent



Gradient Descent

$$w_i \leftarrow w_i - \frac{\eta}{M}\sum_{s=1}^{M} x_i(s)[y(s) - t(s)]$$

Single Weight Update per Epoch → Slow Convergence



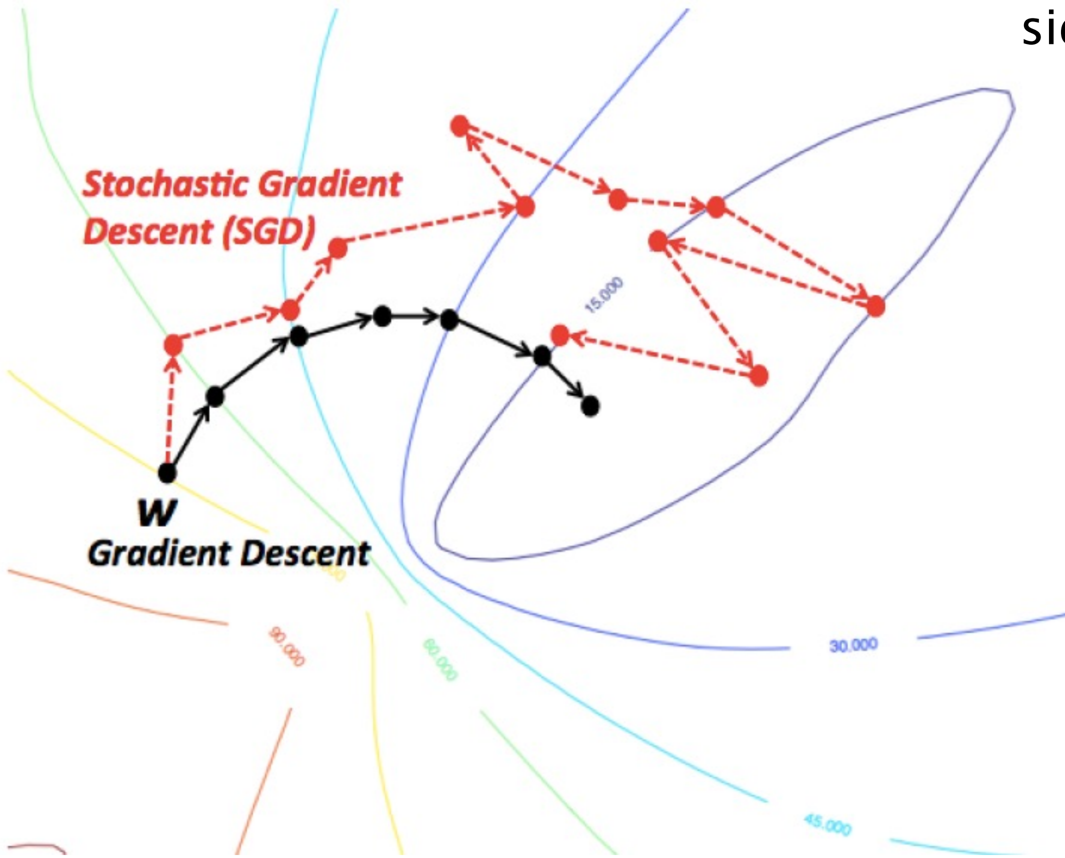
Stochastic Gradient Descent

$$w_i \leftarrow w_i - \eta x_i(s)[y(s) - t(s)]$$

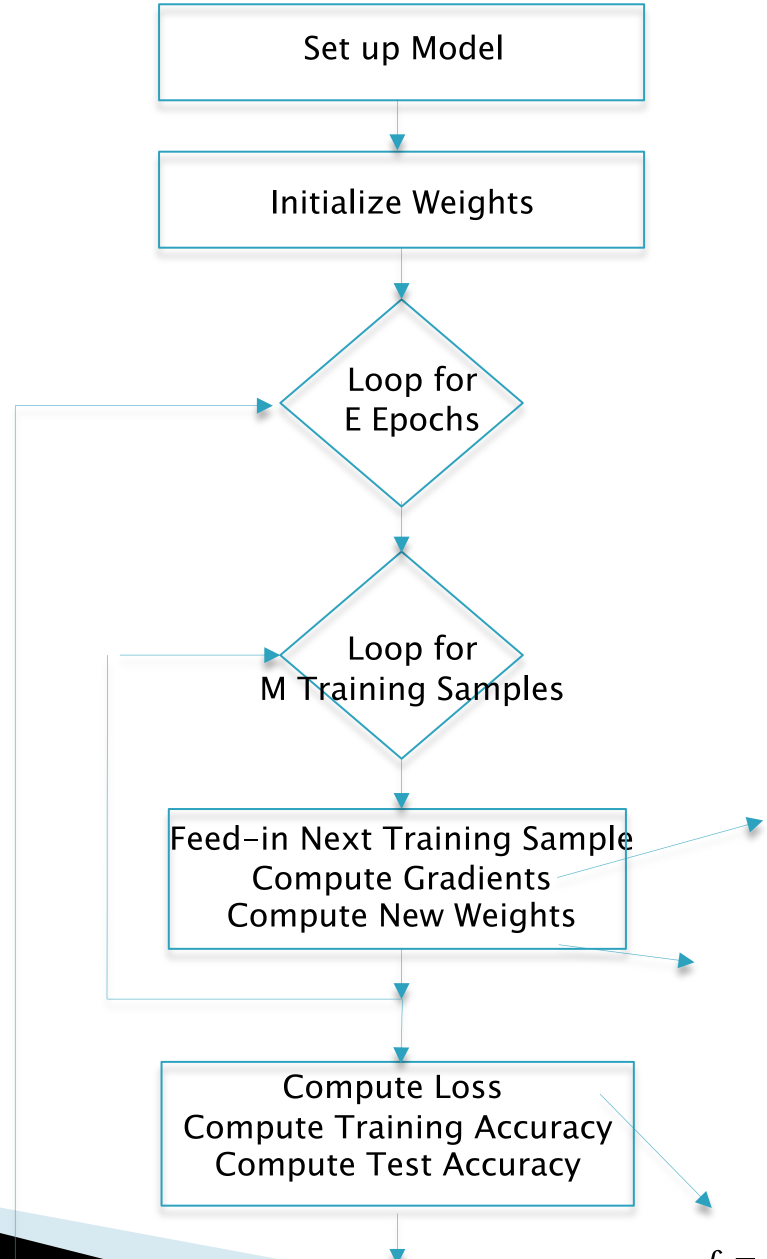
Multiple Weight Updates per Epoch → Faster Convergence, but in a noisy manner

Gradient Descent vs Stochastic Gradient Descent

The jumping around has some side benefits!



Stochastic Gradient Descent Algorithm

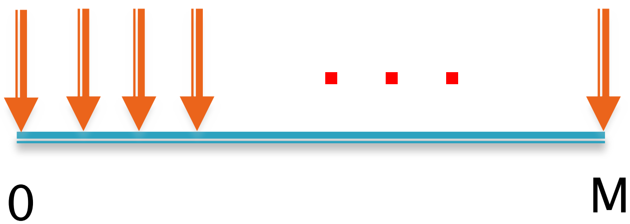


$$\frac{\partial \mathcal{L}}{\partial w_i} = x_i(s)[y\ (s) - t\ (s)]$$

$$w_i \leftarrow w_i - \eta x_i(s)[y(s) - t(s)]$$

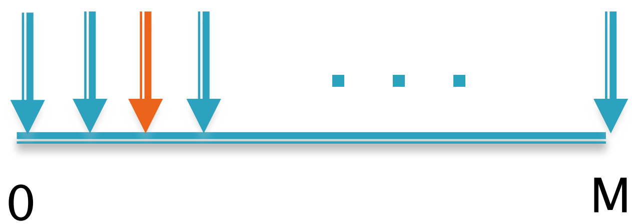
$$\mathcal{L} = -[t(s) \log y(s) + (1 - t(s) \log \left(1 - \mathrm{y}(s)\right)]$$

Gradient Descent vs Stochastic Gradient Descent vs Batch Stochastic Gradient Descent



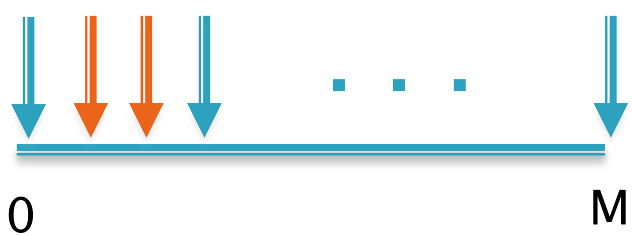
Gradient Descent

$$w_i \leftarrow w_i - \frac{\eta}{M}\sum_{s=1}^{M} x_i(s)[y(s) - t(s)]$$



Stochastic Gradient Descent

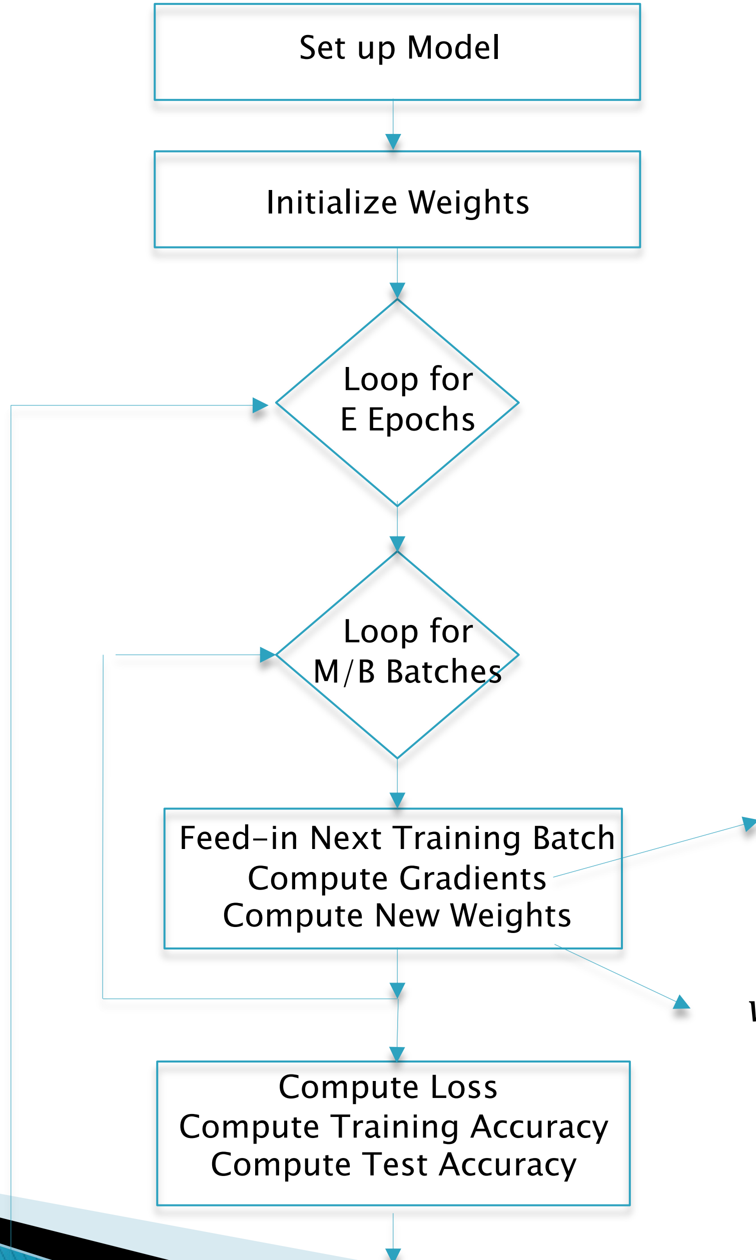
$$w_i \leftarrow w_i - \eta x_i(s)(y(s) - t(s))$$



Batch Stochastic Gradient Descent

$$w_i \leftarrow w_i - \frac{\eta}{B}\sum_{s=1}^{B} x_i(s)[y(s) - t(s)]$$

Batch Stochastic Gradient Descent Algorithm



$$\frac{\partial L}{\partial w_i} = \frac{1}{B} \sum_{s=1}^{B} x_i(s)[y\,(s) - t\,(s)]$$

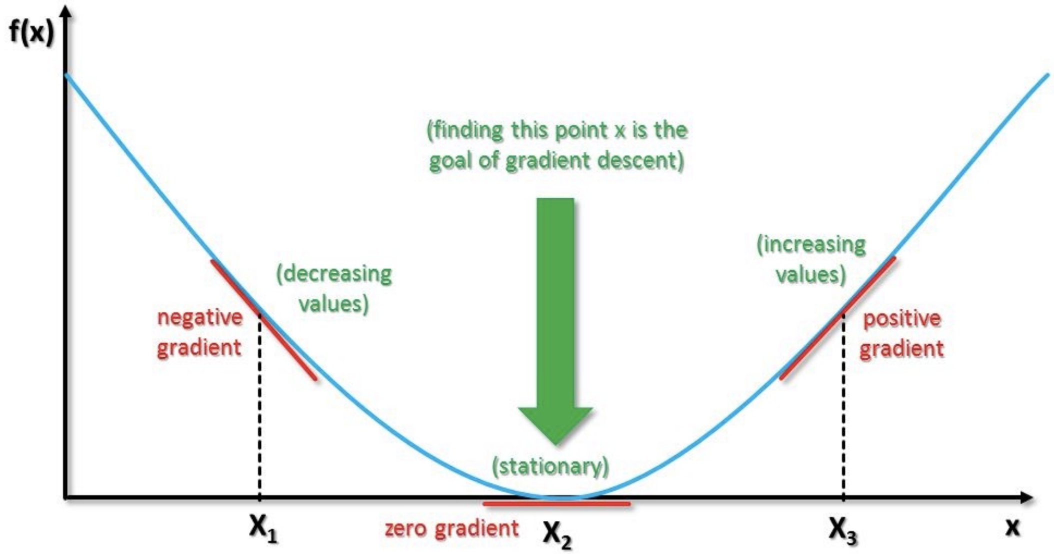
$$w_i \leftarrow w_i - \frac{\eta}{B} \sum_{s=1}^{B} x_i(s)[y(s) - t(s)]$$

Issues in Running Gradient Descent Algorithms

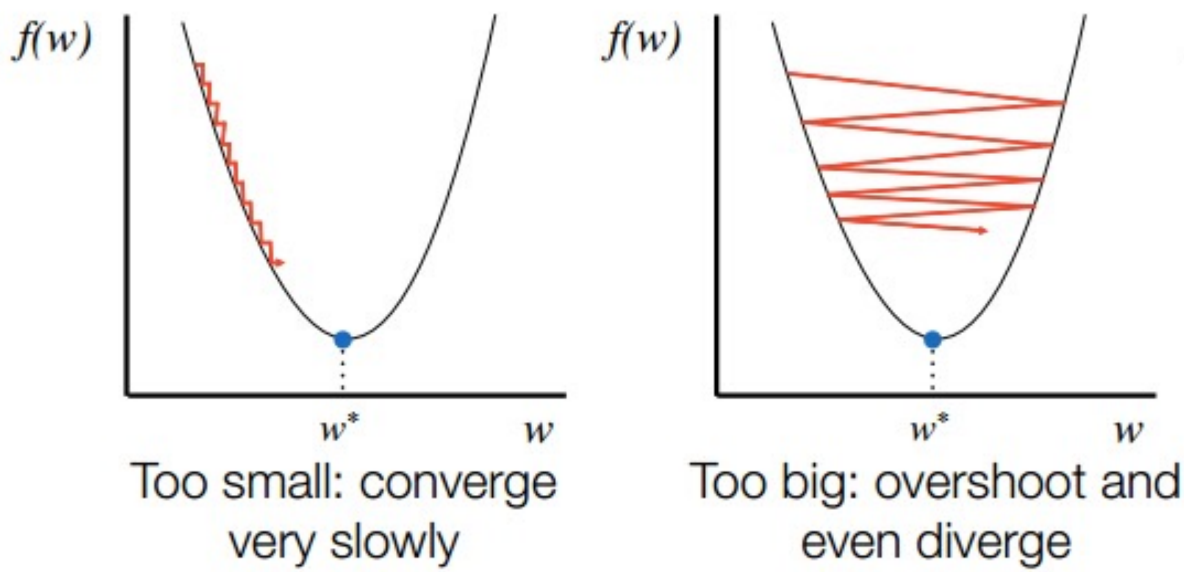
- Choosing the Learning Rate parameter η
- Weight Initialization
- Deciding when to stop the algorithm

Gradient Descent: Choice of η

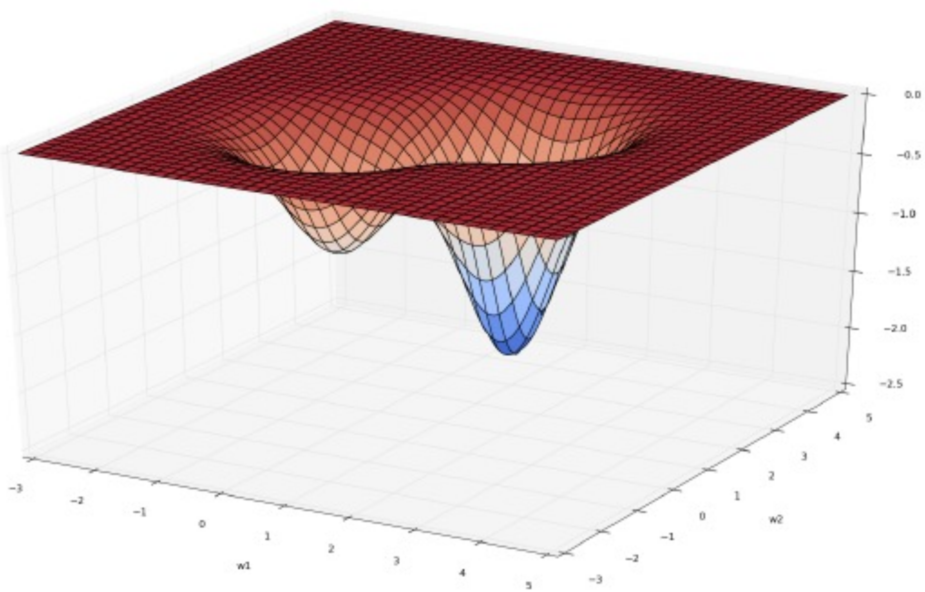
$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i}$$

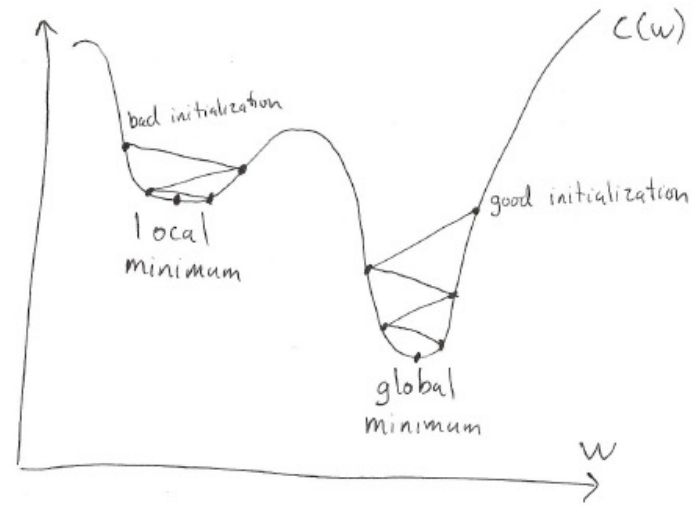


Effect of choice
Of η

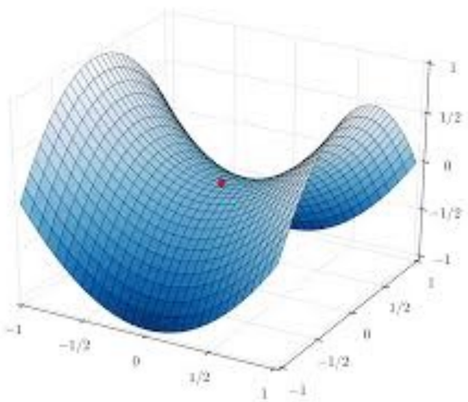


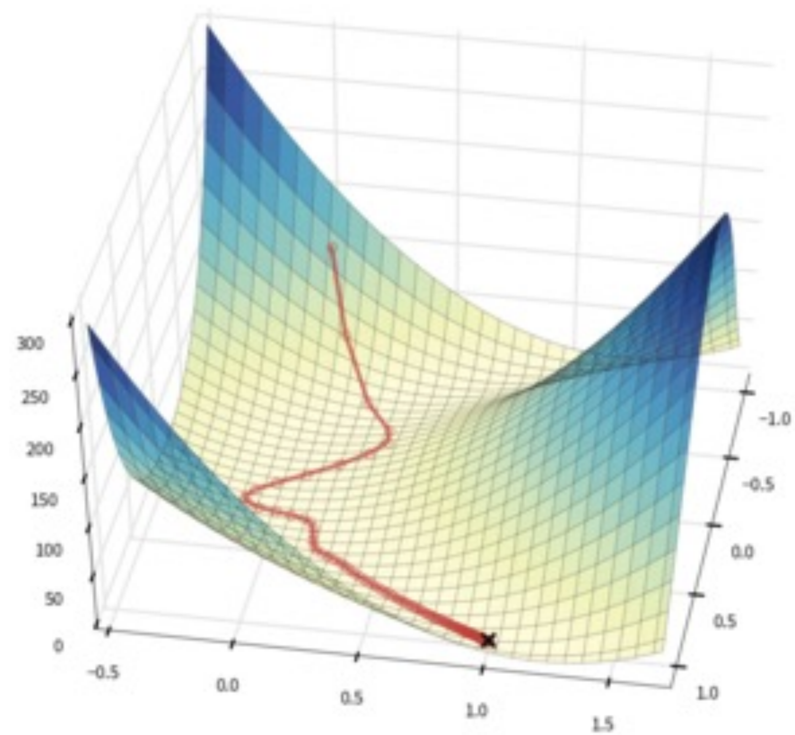
Gradient Descent: Initialization





Gradient Descent: Saddle Points

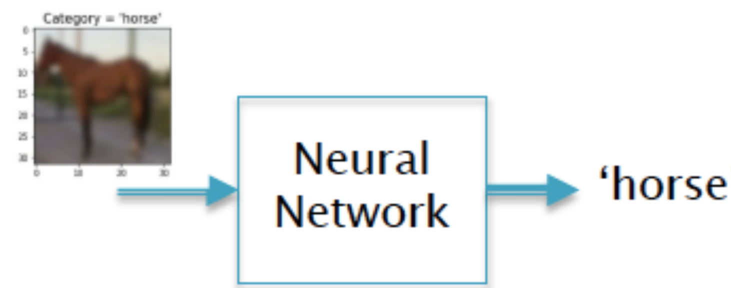




Linear Classification Models with K Classes

Image Classification: CIFAR-10 Image Dataset

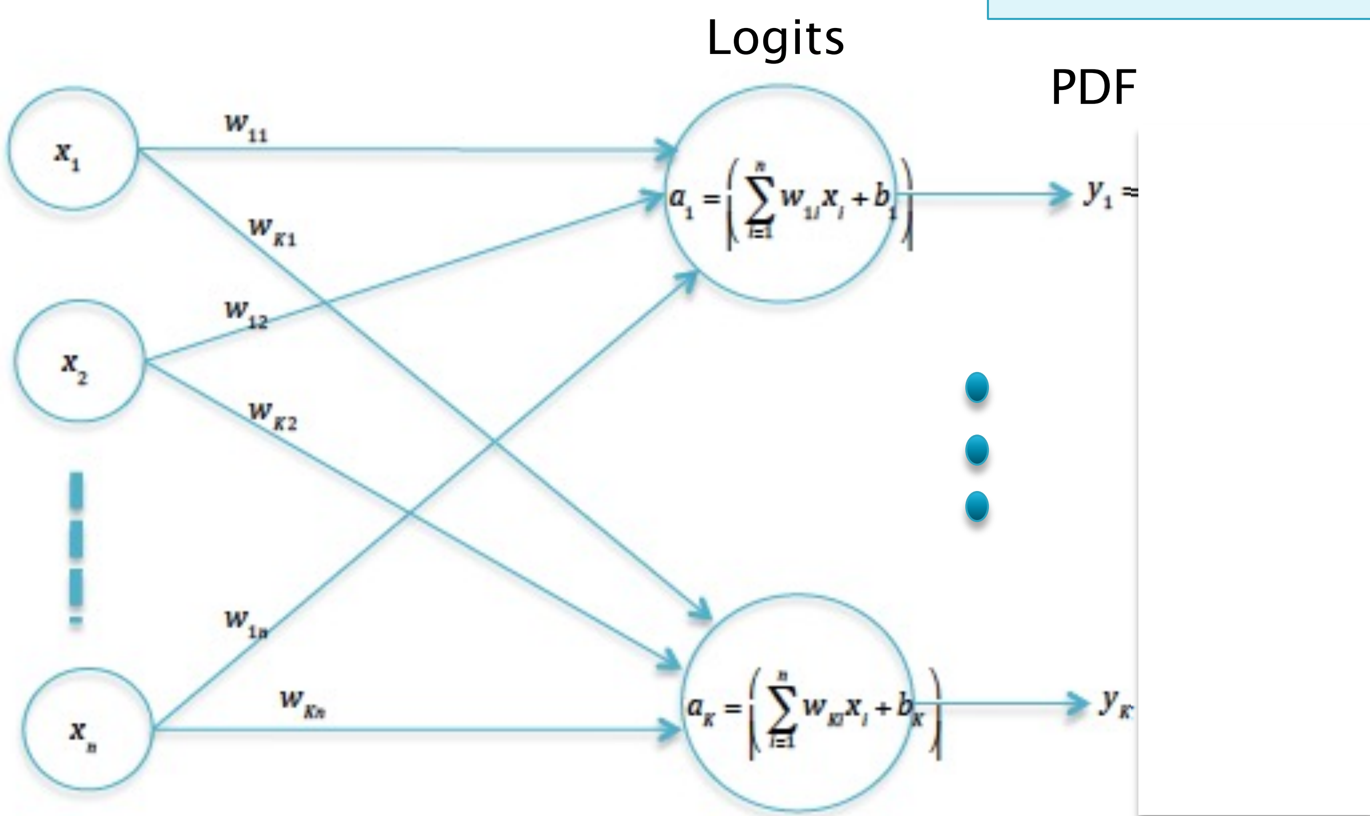
10 classes
50,000 training images
10,000 testing images



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

K-ary Classification

How to convert the logits into a PDF?



$$A = WX + B$$
$$Y = h(A)$$

The SoftMax Classifier

$$y_k = h_k(a_1, \ldots, a_K) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}$$

Sum of all K outputs is 1
Results in a probability distribution

- Appropriate for K-ary classification networks

Loss Function

Loss Function for the s[th] Sample

$$\mathcal{L}(s) = -\sum_{k=1}^{K} t_k\,(\mathrm{s})\log y_k(s)$$

Loss Function for the Entire Training Set

$$L(W) = -\frac{1}{M}\sum_{s=1}^{M}\sum_{k=1}^{K} t_k\,(s)\log y_k(\mathrm{s})$$

Gradient Calculation

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial \mathcal{L}}{\partial w_{kj}}$$

Evaluate $\frac{\partial \mathcal{L}}{\partial w_{kj}}$, *where*

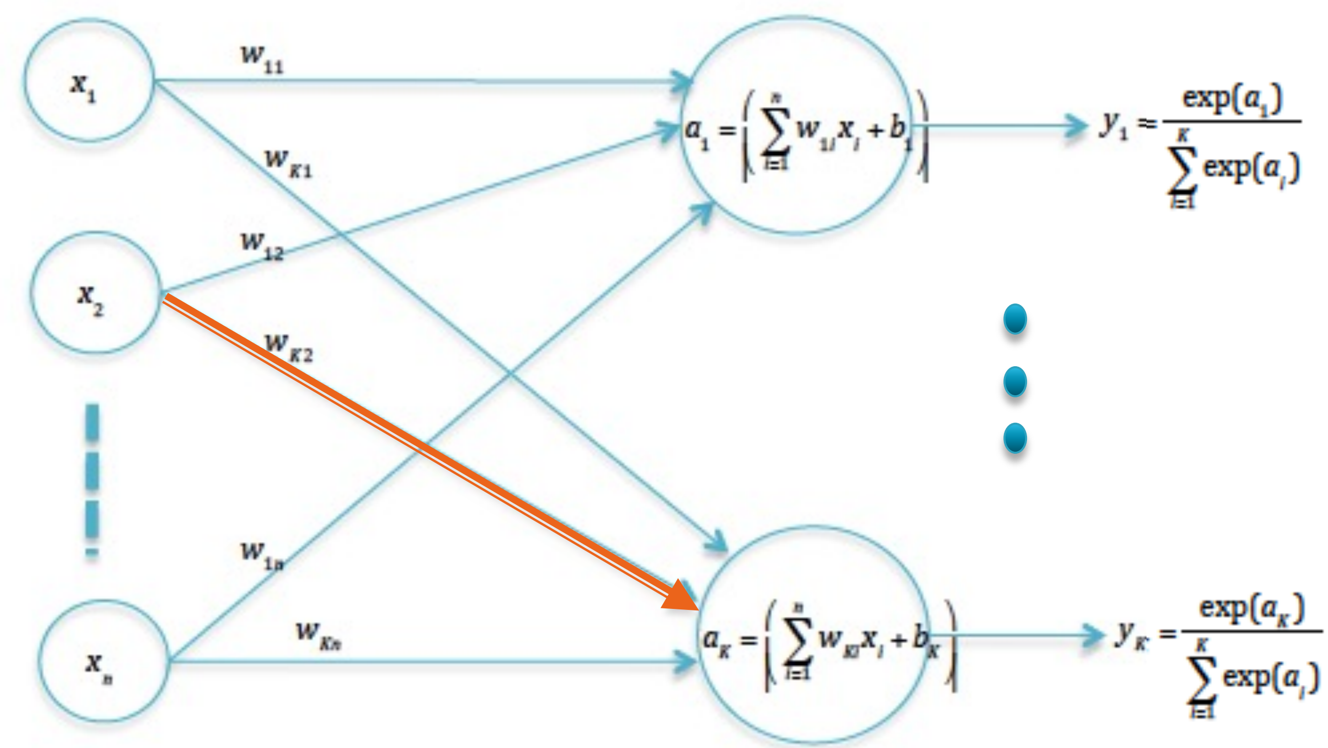
$\mathcal{L} = -\sum_{k=1}^{K} t_k \log y_k$, and

$y_k = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}$, $a_k = \sum_{j=1}^{N} w_{kj}\, x_j + b_k$

Answer:

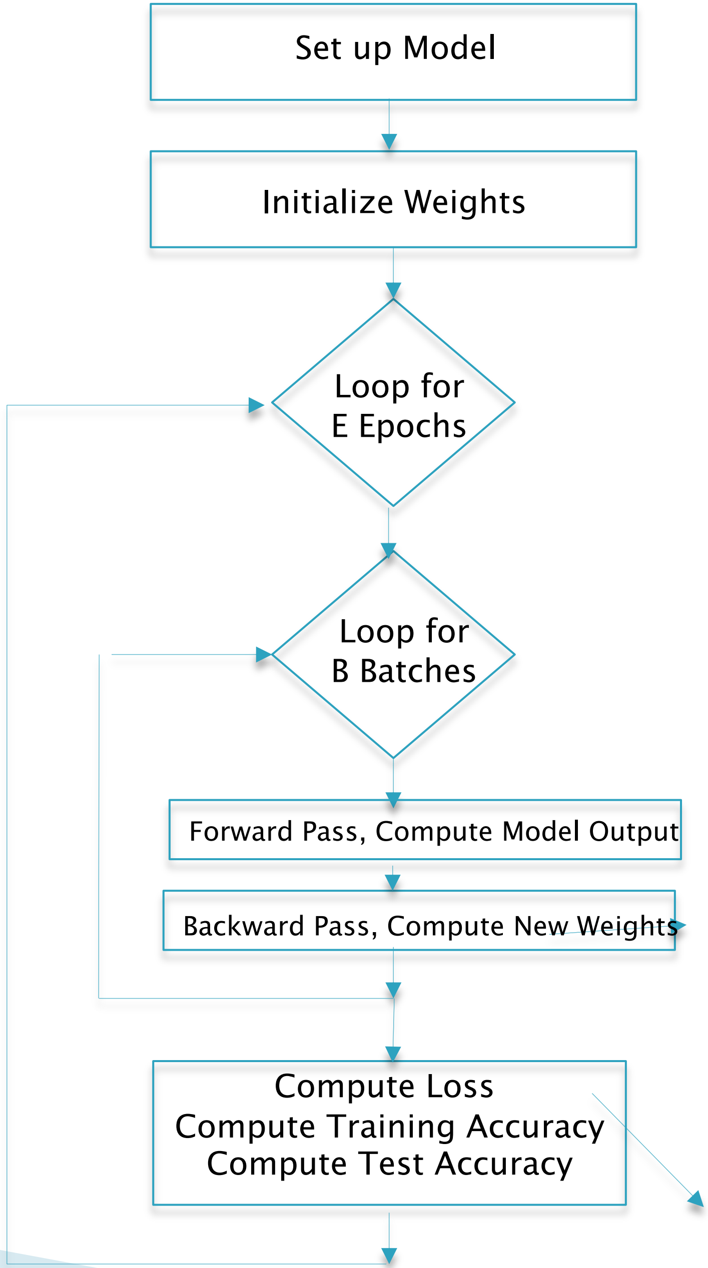
$$\frac{\partial \mathcal{L}}{\partial w_{kj}} = x_j\,(y_k - t_k)$$

K-ary Classification



If $\mathcal{L} = -\sum_{k=1}^{K} t_k \log y_k$
then
$$\frac{\partial \mathcal{L}}{\partial w_{kj}} = x_j\,(y_k - t_k)$$

Training Using Batch Gradient Descent For K-ary Classification



$$y_k = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}, \quad a_k = \sum_{j=1}^{N} w_{kj}\, x_j + b_k$$

$$w_{kj} \leftarrow w_{kj} - \frac{\eta}{B} \sum_{j=1}^{B} x_j(s)[y_k(s) - t_k(s)]$$

$$\mathrm{L} = -\frac{1}{B} \sum_{s=1}^{B} \sum_{k=1}^{K} t_k(s) \log y_k(s)$$

Supplementary Reading

- Chapter 2: Pattern Recognition
- Chapter 5: Supervised Learning
- Chapter 6: Linear Learning Models

https://srdas.github.io/DLBook2/