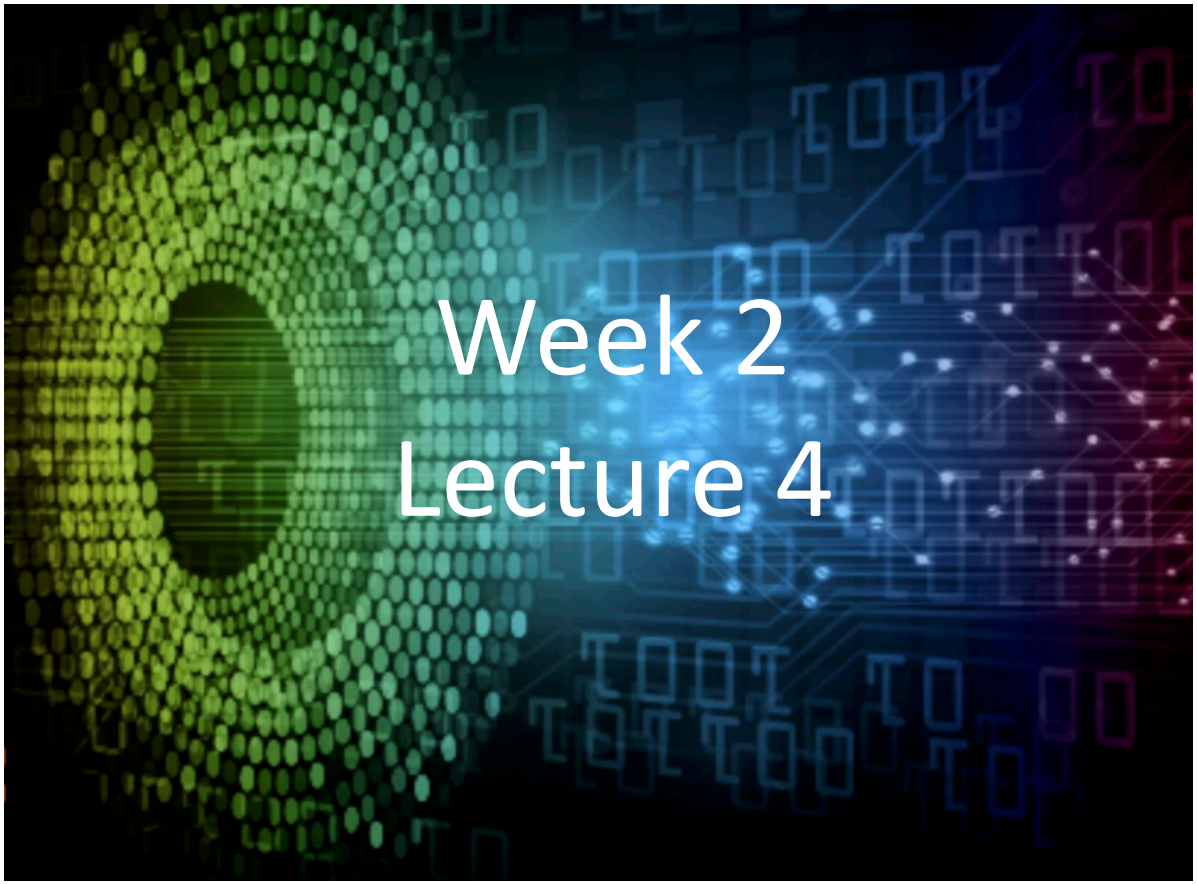
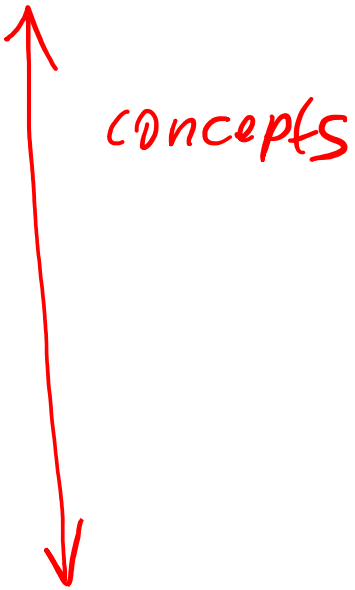
Introduction to Deep Learning
Applications and Theory

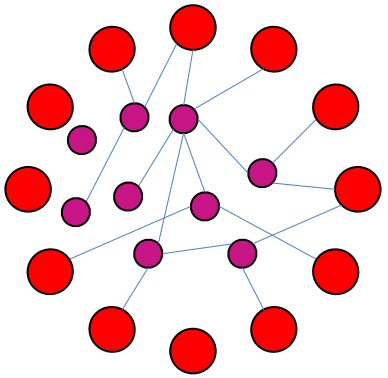


ECE 596 / AMATH 563

Previous Lecture: **Neural Networks Fundamentals**

1. Single neural unit
2. Connecting neurons
3. Flow and Recurrences
4. Layers
5. Learning and Data





Current Lecture: **Network Training/Optimization**

1. Regression /Classification
2. Loss and Cost Functions
3. Logistic Regression example
4. Gradient Descent ← optimization method

Supervised Learning – Input / Output

$X \in \mathbb{R}^n$

input

input ↔ output

$D : \{(X^{(1)}, Y^{(1)}), ..., (X^{(m)}, Y^{(m)})\}$

training set

typically stacked

test set, validation set

$Y \in \{0, 1\}$

$Y \in \{0, 1\}^k$

classification

$Y \in \mathbb{R}^k$

Regression/ synthesis

Input - Output

Regression

output input parameters

$$Y = f(X, W)$$

model

linear

Linear Regression

input

$$\vec{x} \in \mathbb{R}^n$$

output

$$y \in \mathbb{R}$$

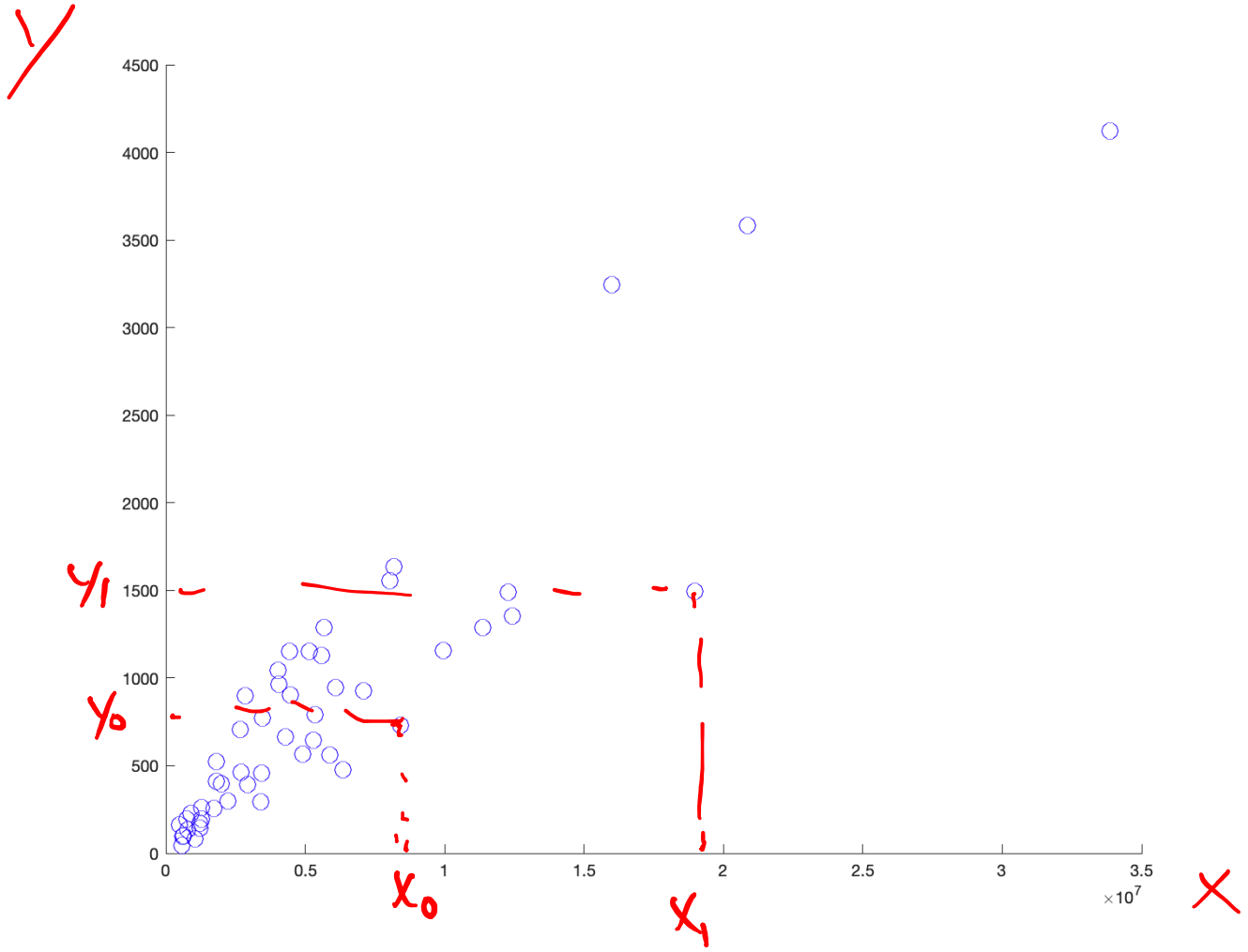
Least Squares Fit of Polynomial

$$y(x) = p_0 + p_1 x + \ldots + p_m x^m$$

$$\vec{\tilde{x}} = (1, x, x^2, \ldots, x^m)$$

$$y = \vec{p} \cdot \vec{\tilde{x}}$$

Least Squares Fit



Linear Regression

Fit the model based on the given points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

n given points

$$X = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^m \\ 1 & x_1 & x_1^2 & \ldots & x_1^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{pmatrix} \begin{matrix} \tilde{x}_0 \\ \\ \\ \\ \tilde{x}_n \end{matrix}$$

Fit ("Learn") the model:

$$J = E_2 = \sum_{i=1}^{n} (\vec{p} \cdot \tilde{\vec{x}}_i - y_i)^2$$

LSE

cost — total error — $i=1$ — $\vec{p}$ — loss $\approx L$

Linear Regression

$$J = E_2 = \sum_{i=1}^{n} (\vec{p} \cdot \tilde{\vec{x}}_i - y_i)^2 = \sum_{i=1}^{n} (f(p, x_i) - y)^2$$

$$\vec{p}^* = \arg\min_{\vec{p}} J(\vec{p})$$

optimize cost

$$\forall i: \quad \frac{\partial J}{\partial p_j} = 0$$

$$\frac{\partial J}{\partial p_j} = \sum_{i=1}^{n} 2(\vec{p} \cdot \tilde{\vec{x}}_i - y_i)\tilde{\vec{x}}_i = 0$$

$$= \sum 2(f - y)\frac{\partial f}{\partial p} = 0$$

$$\vec{p}^* = (X^\mathsf{T} X)^{-1} X^\mathsf{T} \vec{y}$$

Q1. What would be Least Squares fit for a general nonlinear model of y = f(X,W)?

Over- and Under-Determined

$$\vec{p}^* = \arg\min_{\vec{p}} L(\vec{p})$$

$$A\vec{p} = \vec{b}$$

- Over-determined:

objective

regularization

$$\arg\min_{\vec{p}} \left(||A\vec{p} - \vec{b}||_2 + \lambda g(p) \right)$$

- Under-determined:

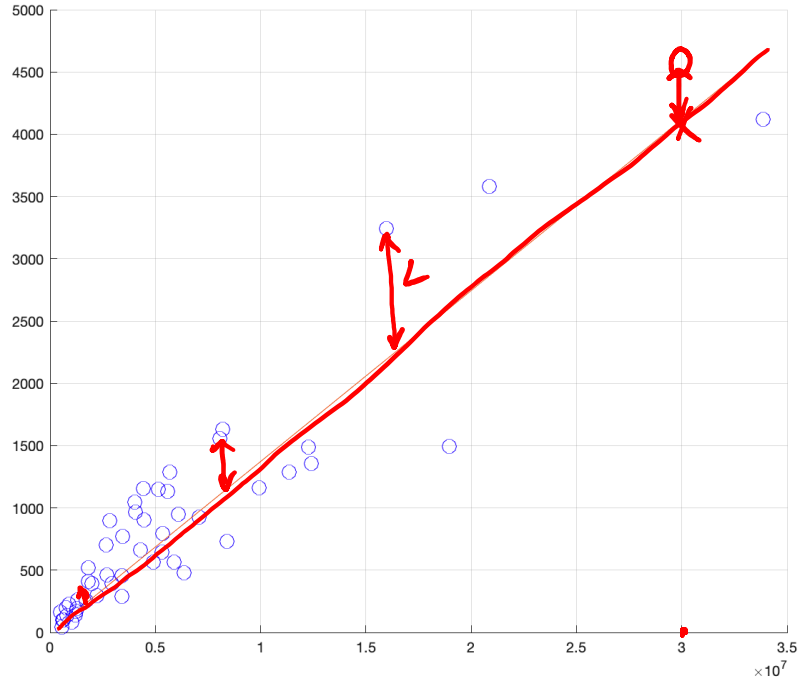
$$\arg\min_{\vec{p}} g(p) \quad \text{subject to} \quad ||A\vec{p} - \vec{b}||_2 \leq \epsilon$$

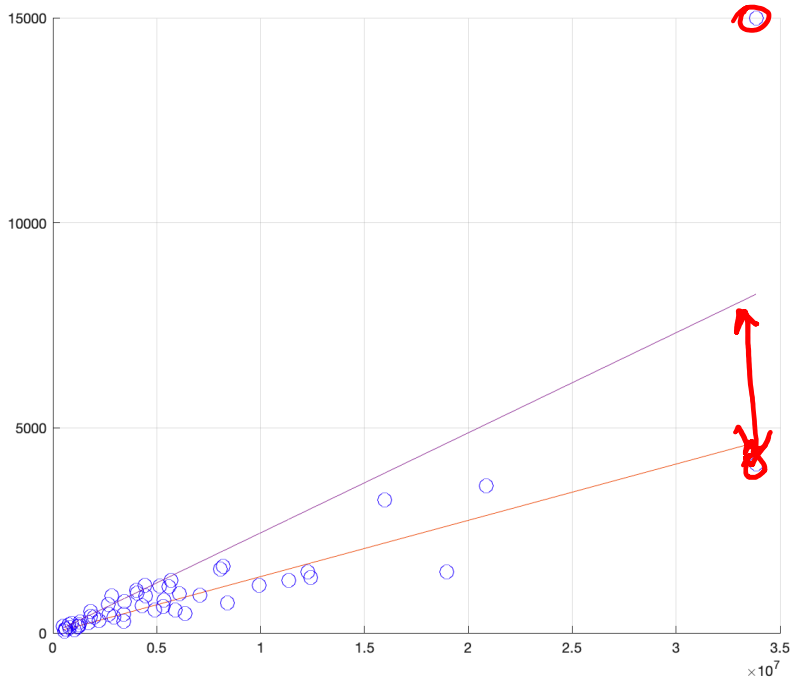
Linear Regression

Test the model:

Cross validation 80% training
20% testing

Given x_k $\longrightarrow$ $y_k = \vec{p}^* \cdot \vec{\tilde{x}}_k$





Nonlinear

$$W^* = \arg\min_W L(X, W)$$

Over-determined:

$$\arg\min_W \left(L(X, W) + \lambda g(W)\right)$$

Under-determined:

$$\arg\min_W g(W) \quad \text{subject to} \quad L(X, W) \leq \epsilon$$

Norm Choice

$$J_2 = \frac{1}{n}\sum_{i=1}^{n}(f(\tilde{x}_i, W) - y_i)^2$$

Mean Square Error

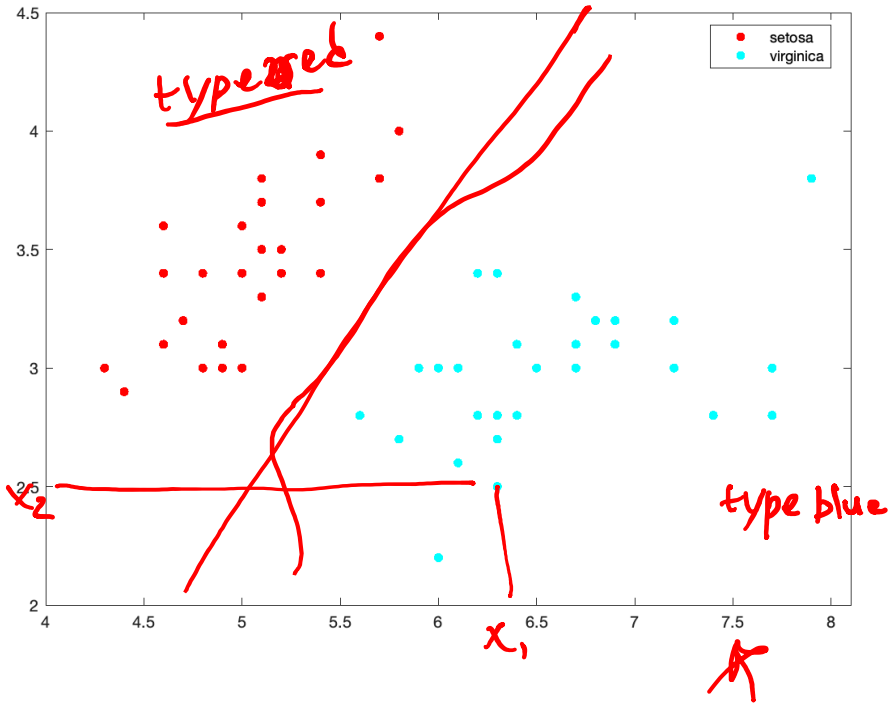
$$J_1 = \frac{1}{n}\sum_{i=1}^{n}|f(\tilde{x}_i, W) - y_i|$$

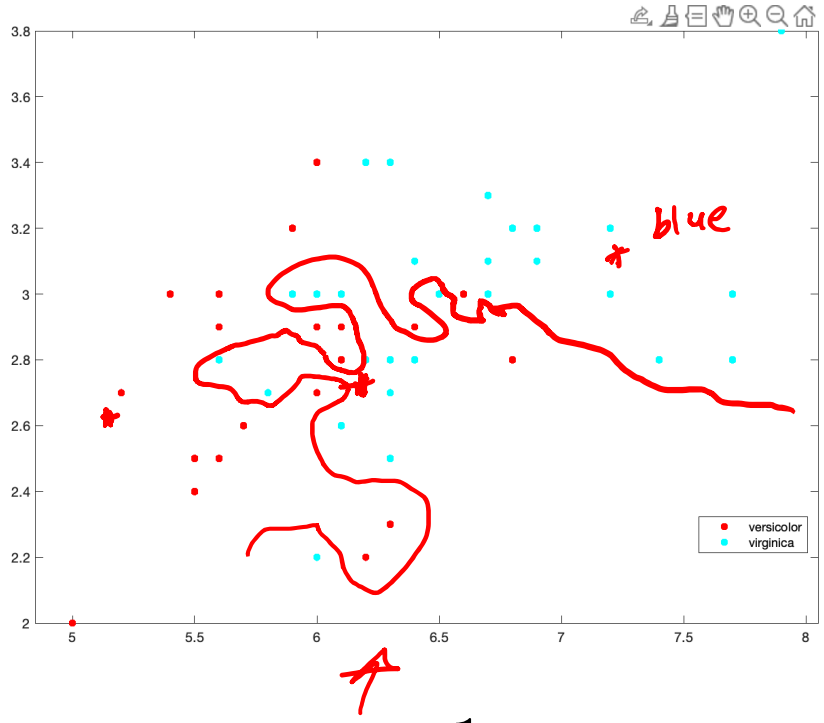
Mean Absolute Error

$$J_\infty = \max_{1<i<n}|f(\tilde{x}_i, W) - y_i|$$

max error

Classification





$$f = \frac{1}{1 + e^{-x}}$$

logistic regression

Supervised Learning – Input / Output

$X \in \mathbb{R}^n$

input

$D : \{(X^{(1)}, Y^{(1)}), ..., (X^{(m)}, Y^{(m)})\}$

training set

typically stacked

test set, validation set

$Y \in \{0, 1\}$

$Y \in \{0, 1\}^k$

classification

$Y \in \mathbb{R}^k$

synthesis

Outputs

$\hat{y} = P(y = 1|X)$

sigmoid

Bernoulli Distribution

$\hat{y}_i = P(y_i = 1|X)$

$softmax(\hat{y})_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$

Multinoulli Distribution

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{i=1}^{n} p(\mathrm{c} = i \mid \boldsymbol{x}) \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}^{(i)}(\boldsymbol{x}), \boldsymbol{\Sigma}^{(i)}(\boldsymbol{x})).$$

Actual values (Gaussian Mixture Model)

Loss

$$L(\hat{y}, y) = \frac{1}{N} \sum_i (\hat{y}_i - y_i)^2$$

MSE

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y}))$$

Cross Entropy

$$L(\hat{y}, y) = -\sum_c (y_{o,c}) \log p_{o,c}$$

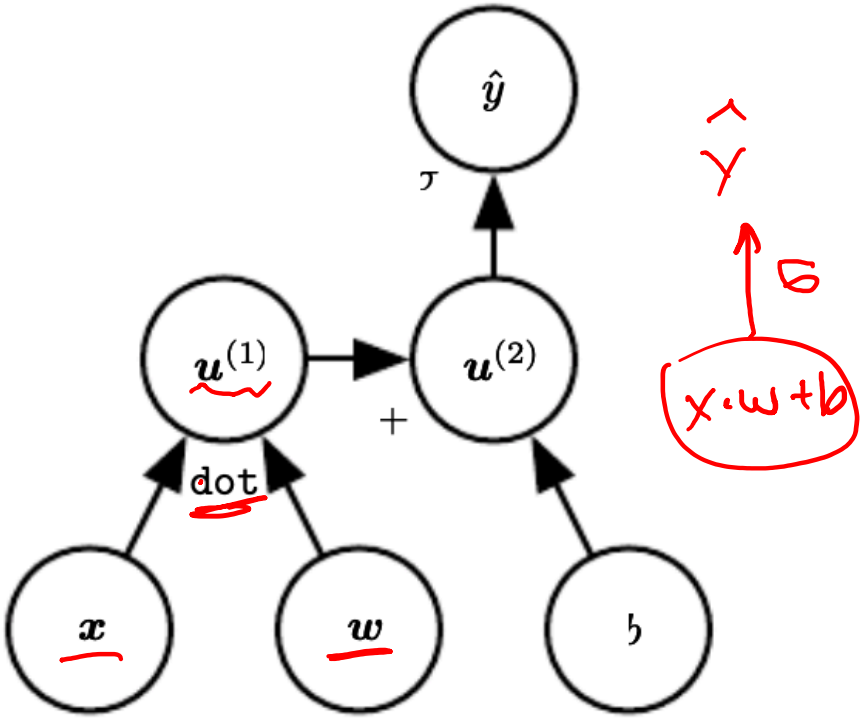
Cross Entropy Multi class

Cost

$$J(W,b) = \frac{1}{M} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

Computation Graph

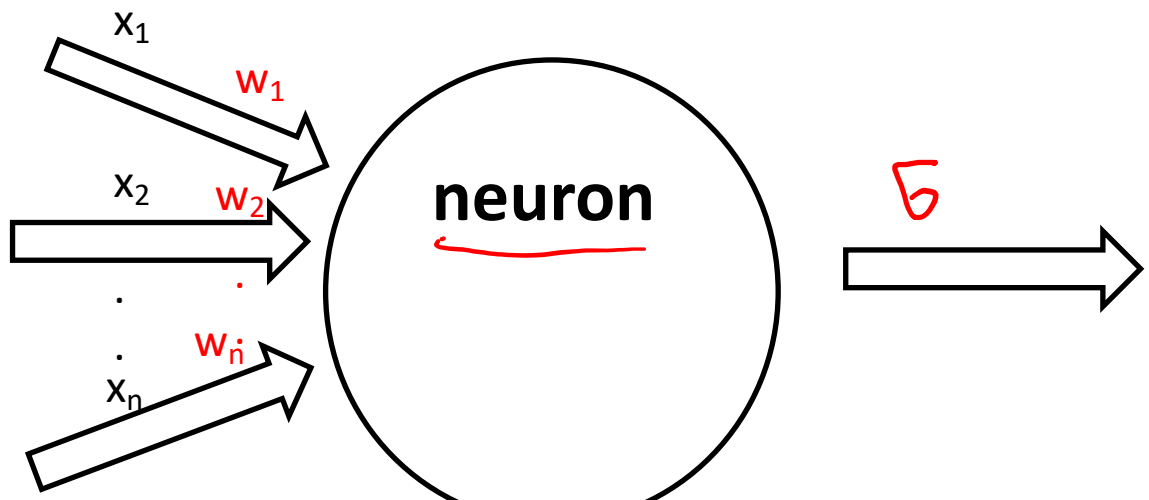
- Efficient
- Sequential Operation
- Chain rule



Q2. How would you break the perceptron online training into a computation graph?

Logistic Regression

We'll start with single neuron logistic regression



input

$$\vec{x} \in \mathbb{R}^n$$

output

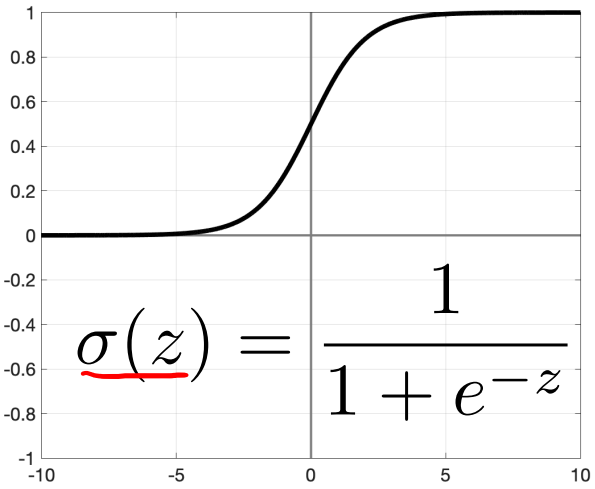
$$\hat{y} = P(y = 1|\vec{x})$$

$$0 \leq \hat{y} \leq 1$$

Activation

input

$$\vec{x} \in \mathbb{R}^n$$



Sigmoid

output

$$\hat{y} = P(y = 1|\vec{x})$$

$$0 \leq \hat{y} \leq 1$$

$$\hat{y} = \sigma(\vec{w}^T \vec{x} + b)$$

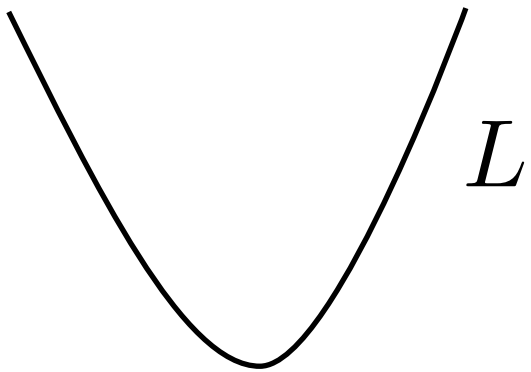
Loss

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y}))$$

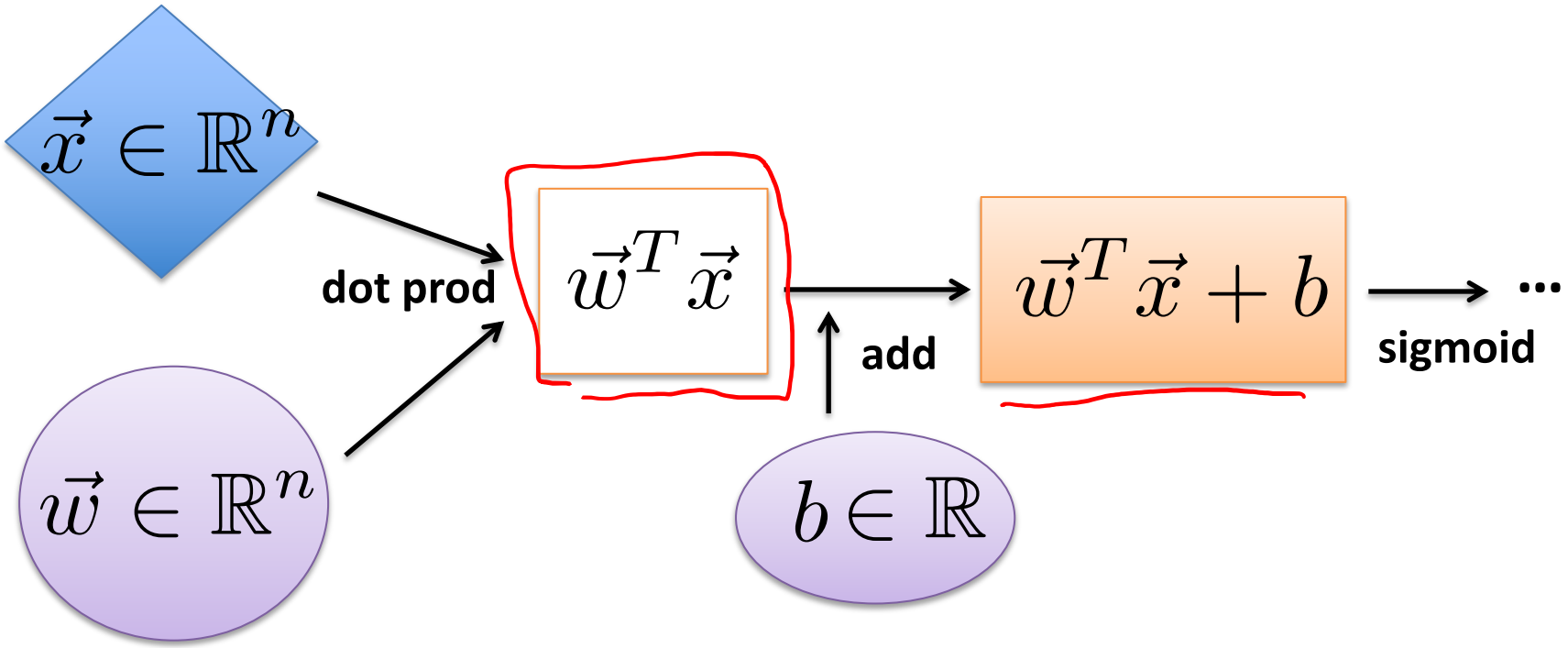
exp = y, pred = $\hat{y}$

Cross Entropy

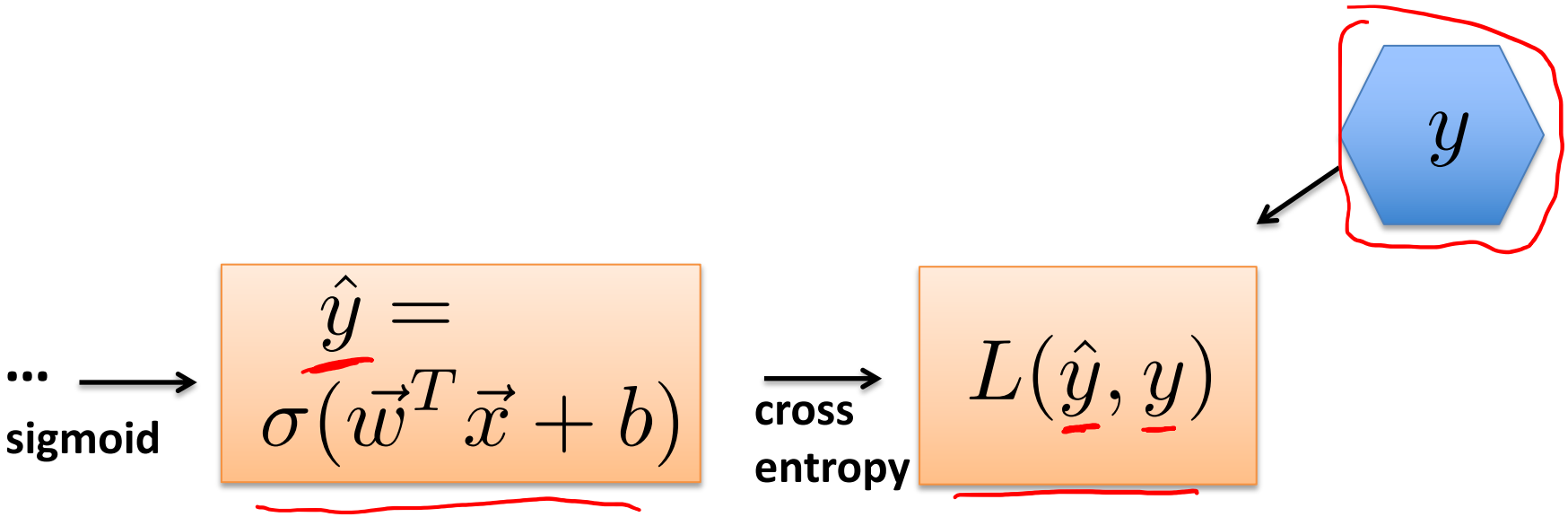
- Maximum Likelihood
- Convex Optimization



Computation Graph



Computation Graph cont.



Cost

For the training set:

$$D : \{(\vec{x}^{(1)}, y^{(1)}), .., (\vec{x}^{(i)}, y^{(i)}), .., (\vec{x}^{(m)}, y^{(m)})\}$$

The cost is:

$$J(\{\hat{y}\}^m, \{y\}^m; \{\vec{x}\}^m) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

$$= -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}))$$

Computation Graph