Deep Learning - Theory and Practice

Linear Regression, Least Squares
Classification and Logistic Regression

20-02-2020

http://leap.ee.iisc.ac.in/sriram/teaching/DL20/

deeplearning.cce2020@gmail.com

Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Least Squares for Classification

- K-class classification problem

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$$

$$t = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- With 1-of-K hot encoding, and least squares regression

$$\sum_{n=1}^{N}$$

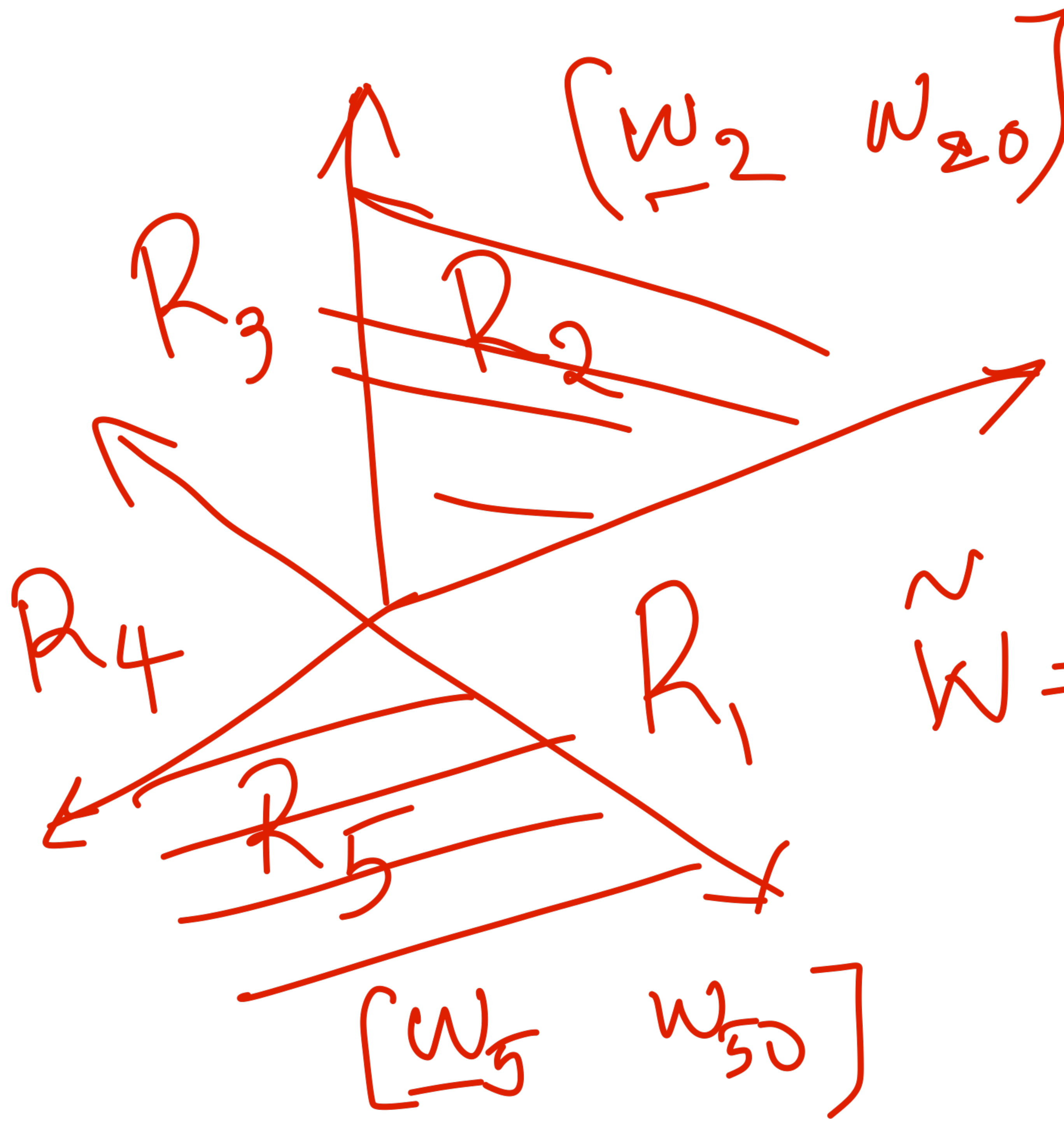
$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{\underbrace{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})}_{N \times N}\right\}$$

Bishop - PRML book (Chap 3)

Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

$$\mathrm{Tr}\left(\begin{bmatrix} \|y(x_1)-t_1\|^2 & & & \\ & \|y(x_2)-t_2\|^2 & & \\ & & \ddots & \\ & & & \|y(x_N)-t_N\|^2 \end{bmatrix}_{N\times N}\right)$$



$$\tilde{W} = \begin{bmatrix} \underline{w}_1^T & w_{10} \\ \underline{w}_2^T & w_{20} \\ \vdots & \end{bmatrix}$$

Logistic Regression (classifier)

- 2- class logistic regression

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma\left(\mathbf{w}^{\mathrm{T}}\phi\right)$$

- Maximum likelihood solution

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

- K-class logistic regression

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

$$a_k = \mathbf{w}_k^{\mathrm{T}}\phi.$$

- Maximum likelihood solution

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N}(y_{nj} - t_{nj})\,\phi_n$$

Bishop - PRML book (Chap 3)

Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Issues

* Outputs are real numbers
* Model targets are binary.
* Cost Function – least squares

Next Model. $\longrightarrow$ make outputs close to prob.

Binary classification (C_1, C_2)

$$P(C_1/\phi(x)) = \frac{P(\phi(x)/C_1)\,P(C_1)}{P(\phi(x))}$$

$\phi(x)$ ← fixed non-linear transformation

$$P(\phi(x)) = P(\phi(x), C_1) + P(\phi(x), C_2)$$

$$P(C_1/\phi(x)) = \frac{1}{1+e^{-f(x)}} ; \quad \frac{1}{1+\frac{P(\phi(x), C_2)}{P(\phi(x), C_1)}}$$

$$\boxed{f(x) = -\ln \frac{P(\phi(x), C_2)}{P(\phi(x), C_1)}}$$

$$P(C_1/\phi(x)) = \frac{1}{1+e^{-f(x)}} = \sigma(f(x))$$

logistic function

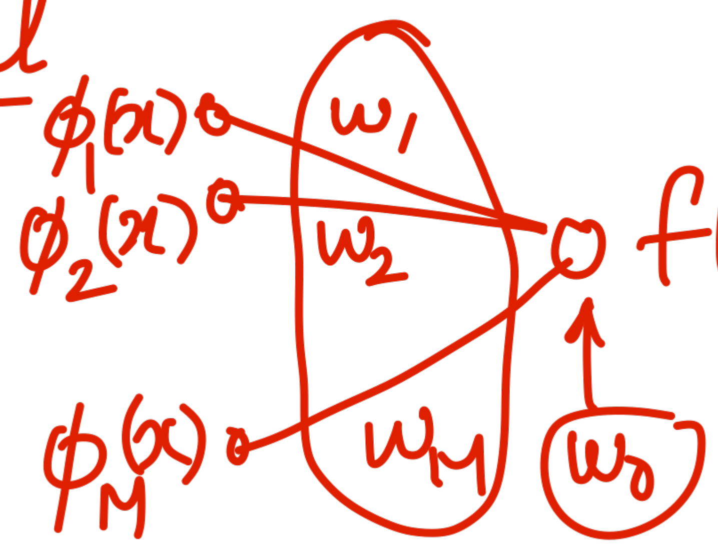
Approx.

$$f(x) = \underline{w}^T \phi(x) + w_0$$

$$\tilde{\underline{w}} = [\underline{w}^T \; w_0]^T$$

Model

$\underline{x} \xrightarrow{\phi}$

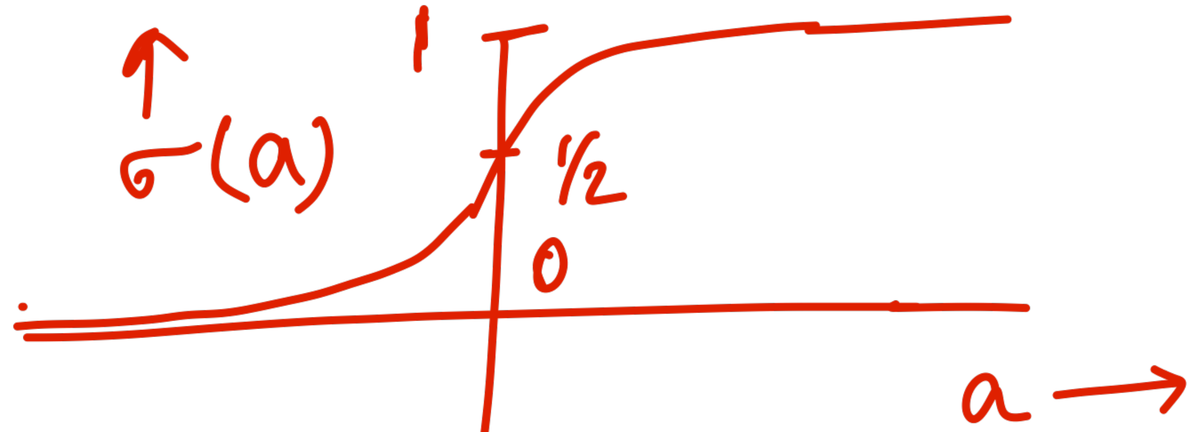


$f(x) \longrightarrow \underline{\underline{\sigma(f(x))}}$

$y(x)$

$$\underline{\underline{\hat{P}(C_1/\phi(x))}}$$

$\sigma(a)$



$$p(C_2/\phi(x)) = 1 - \sigma(f(x))$$
$$= \sigma(-f(x))$$

$$t_n = 0 \text{ if } x_n \in C_2$$
$$= 1 \text{ if } x_n \in C_1$$

$$y(x) = \underline{\sigma(f(x))} = p(C_1/\phi(x)) \quad x_1, x_2 \ldots x_N.$$

Likelihood

$$\underset{w}{\uparrow} L(X) = \prod_{n=1}^{N} (y(\underline{\underline{x_n}}))^{t_n} (1 - y(x_n))^{1-t_n}$$

w.r.t $\underline{w}$, w_0

$$\log L(X) = \sum_{n=1}^{N} t_n \log y_n + (1-t_n)\log(1-y_n)$$

B.C.E (binary cross entropy) $= -\log L(X)$

between $\left(\begin{bmatrix} y(x_n) \\ 1-y(x_n) \end{bmatrix}, \begin{bmatrix} t_n \\ 1-t_n \end{bmatrix} \right)$

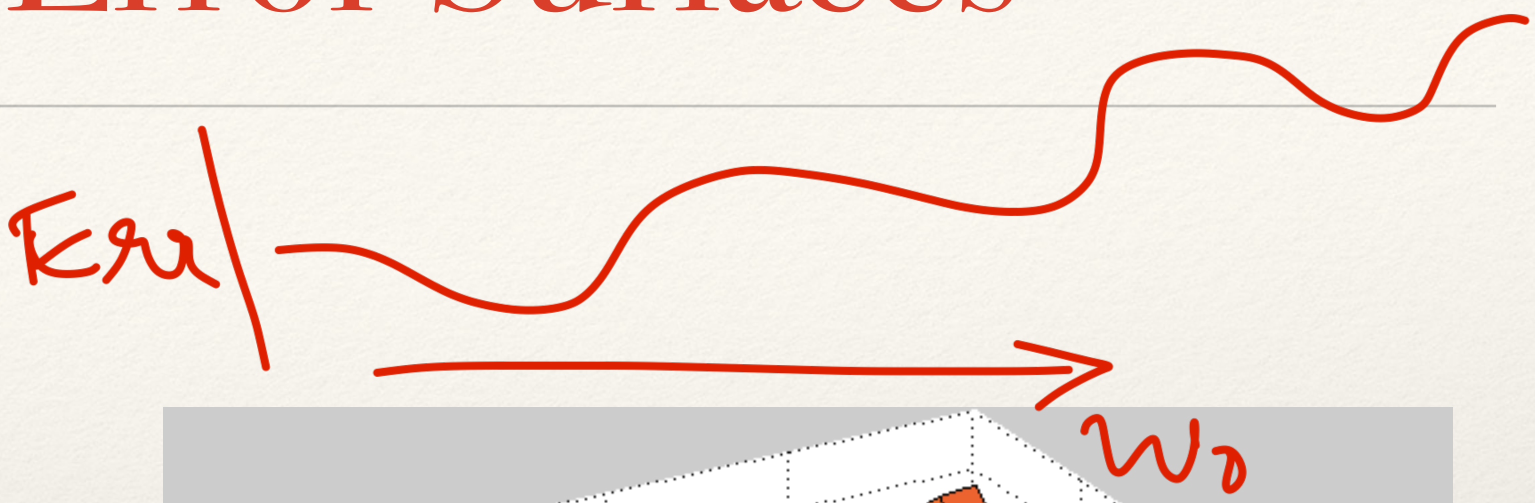
$y_n = y(x_n)$

$$y_n = y(x_n) = \frac{1}{1+e^{-\{w^T\phi(x_n)+w_0\}}}$$

$X = \{x_1 \dots x_N\}$

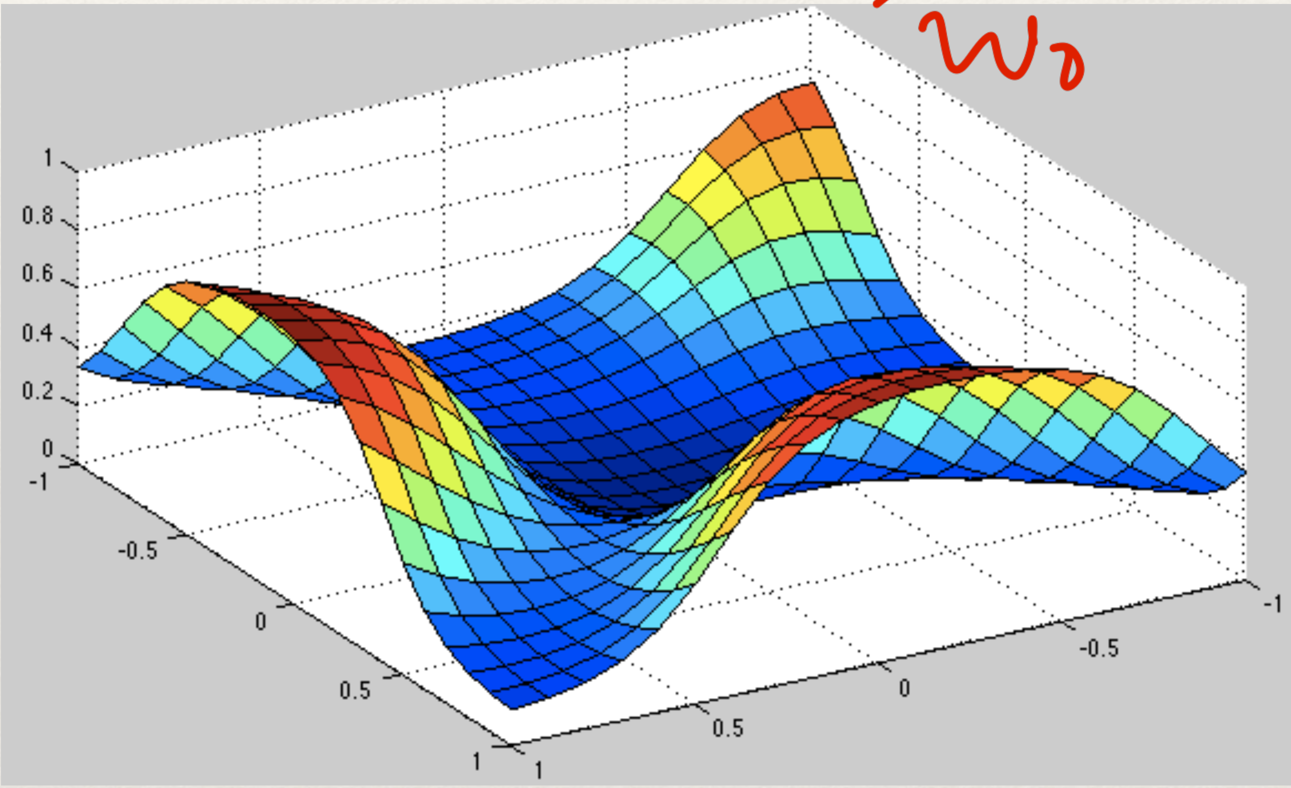
Minimize B.C.E w.r.t w, w_0

Typical Error Surfaces

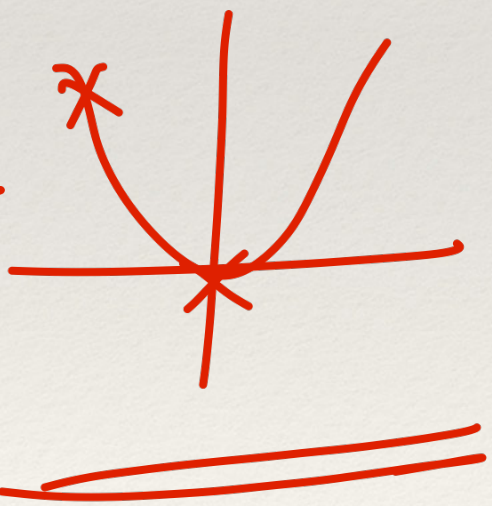


B.C.E

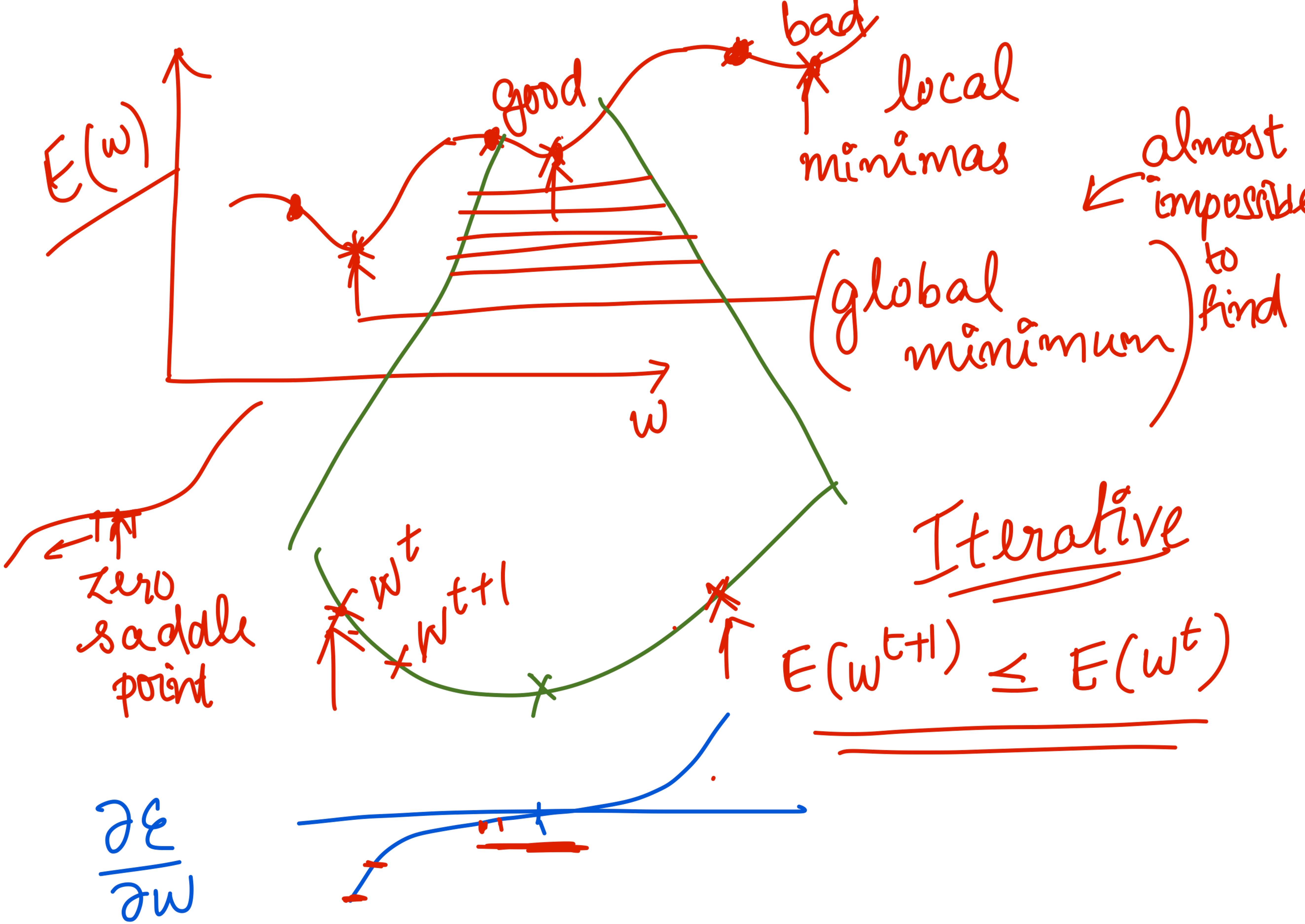
Typical Error Surface as a function of parameters (weights and biases)



Convex



Non-Convex
Multiple max and min



Iterative

$$E(w^{t+1}) \leq E(w^t)$$

Direction (Error surfaces are smooth)

Derivative $\frac{\partial E}{\partial w}$ (-ve direction of derivative)

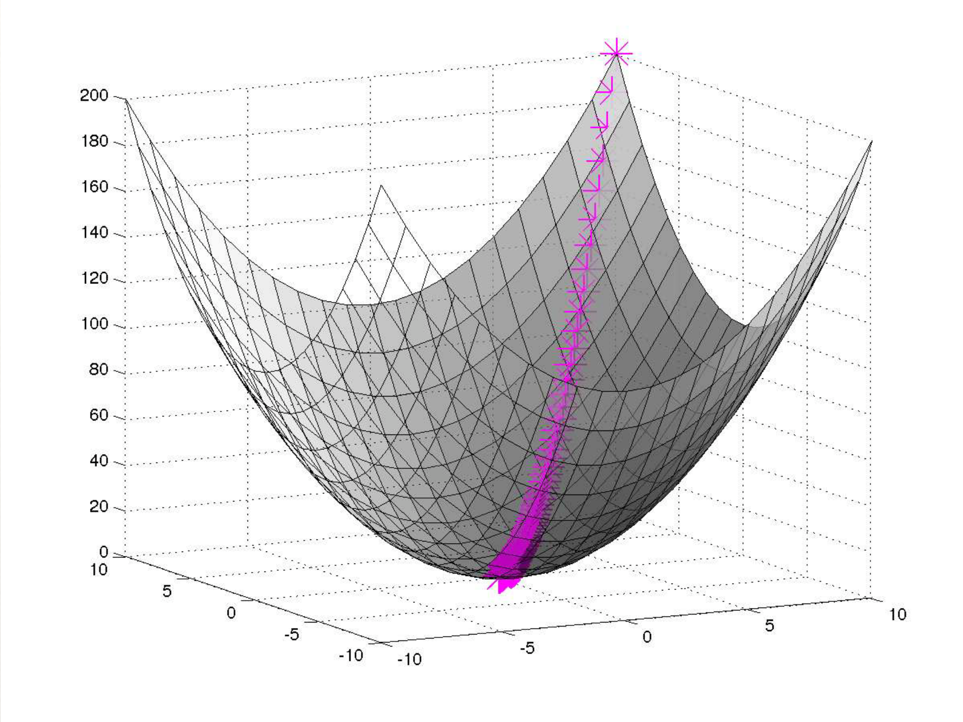
Magnitude - η (small learning rate) +ve

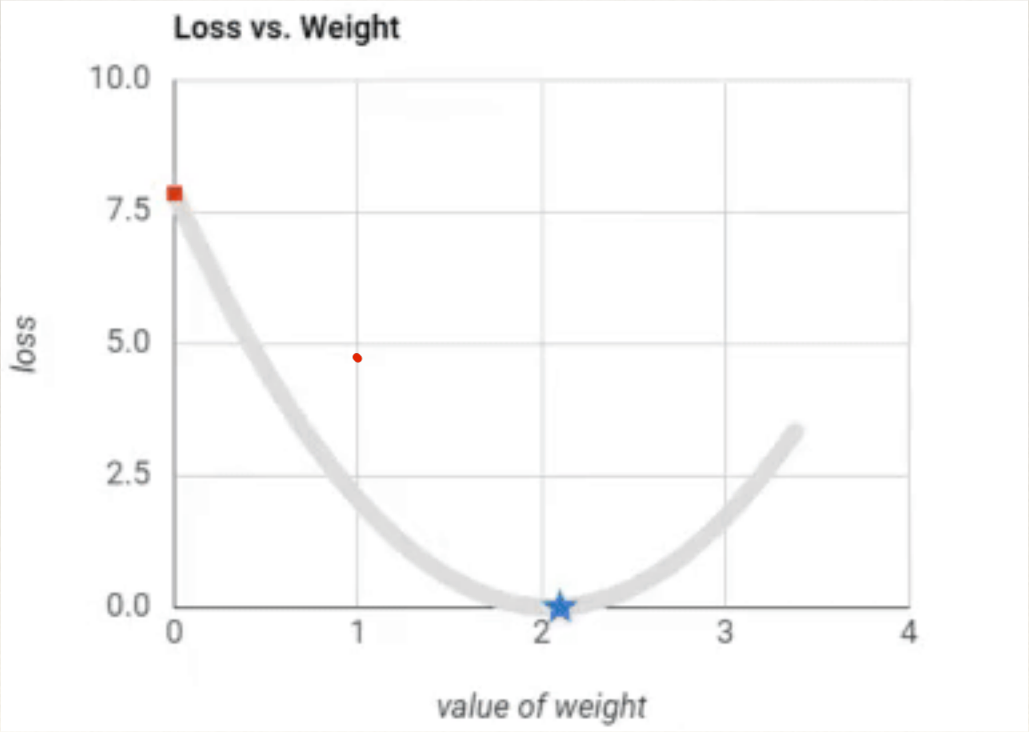
$$\boxed{w^{t+1} = w^{t} - \eta \left. \frac{\partial E}{\partial w} \right|_{w = w^{t}}}$$

Gradient Descent Algorithm

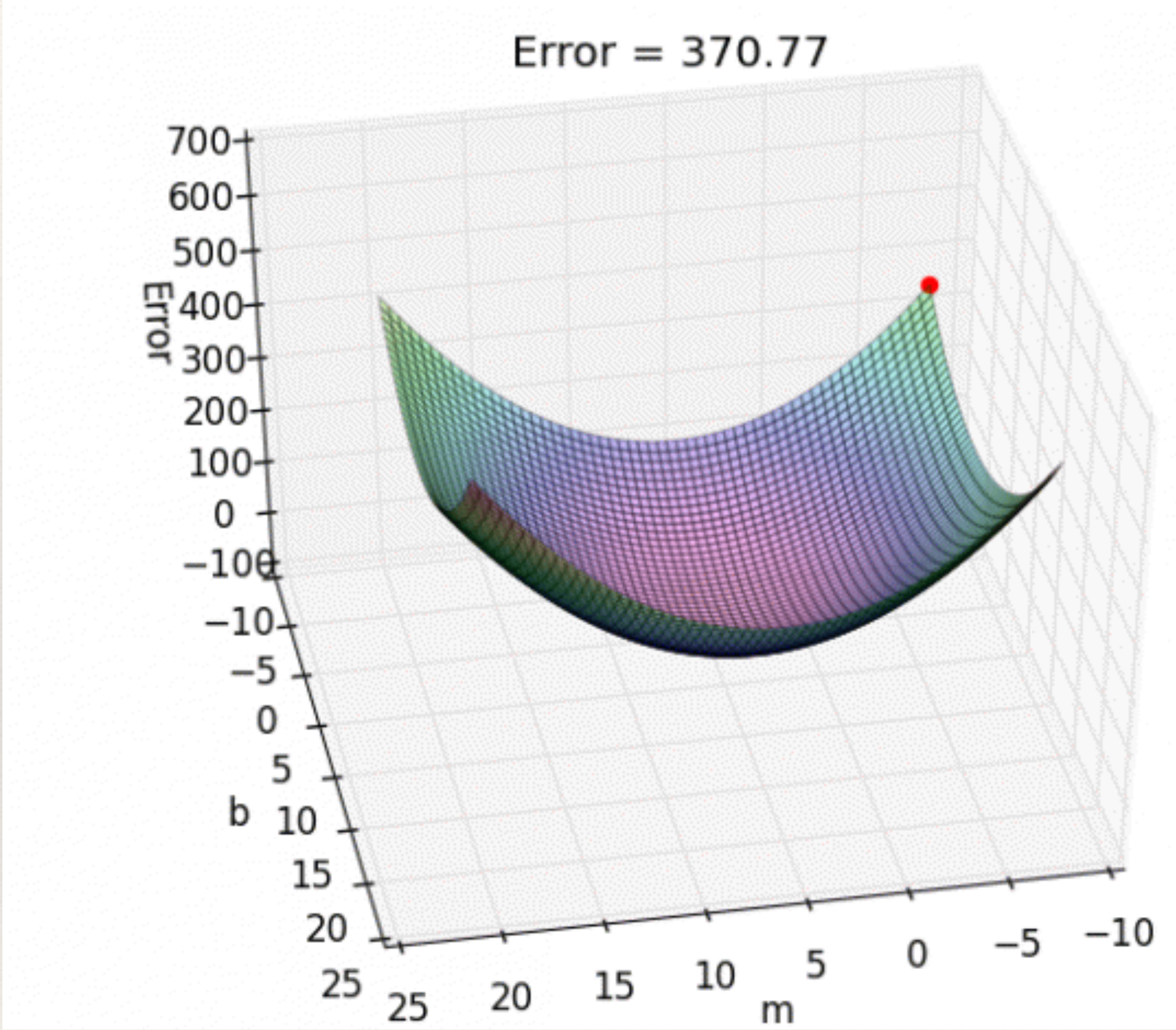
Learning with Gradient Descent

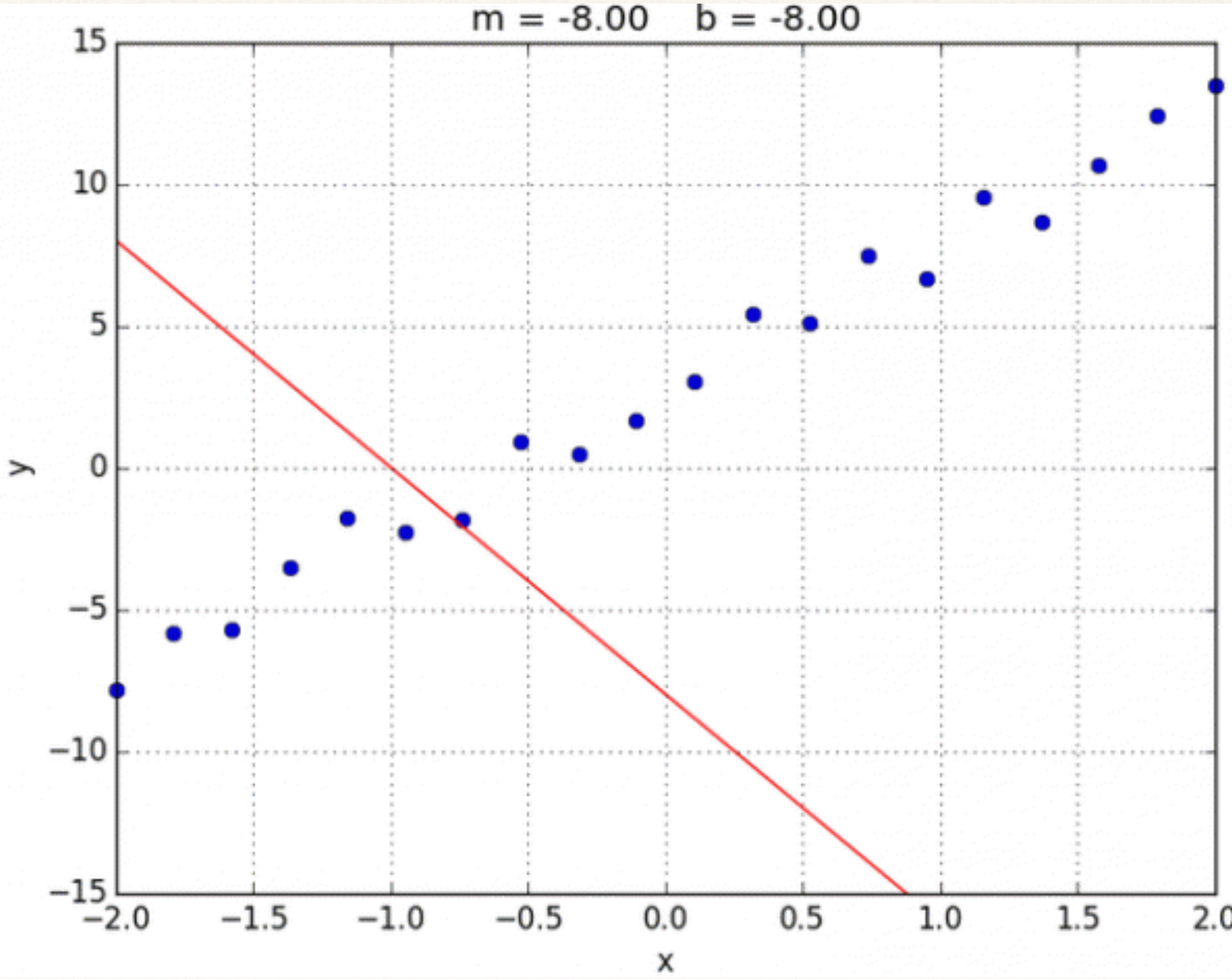
Error surface close to a local





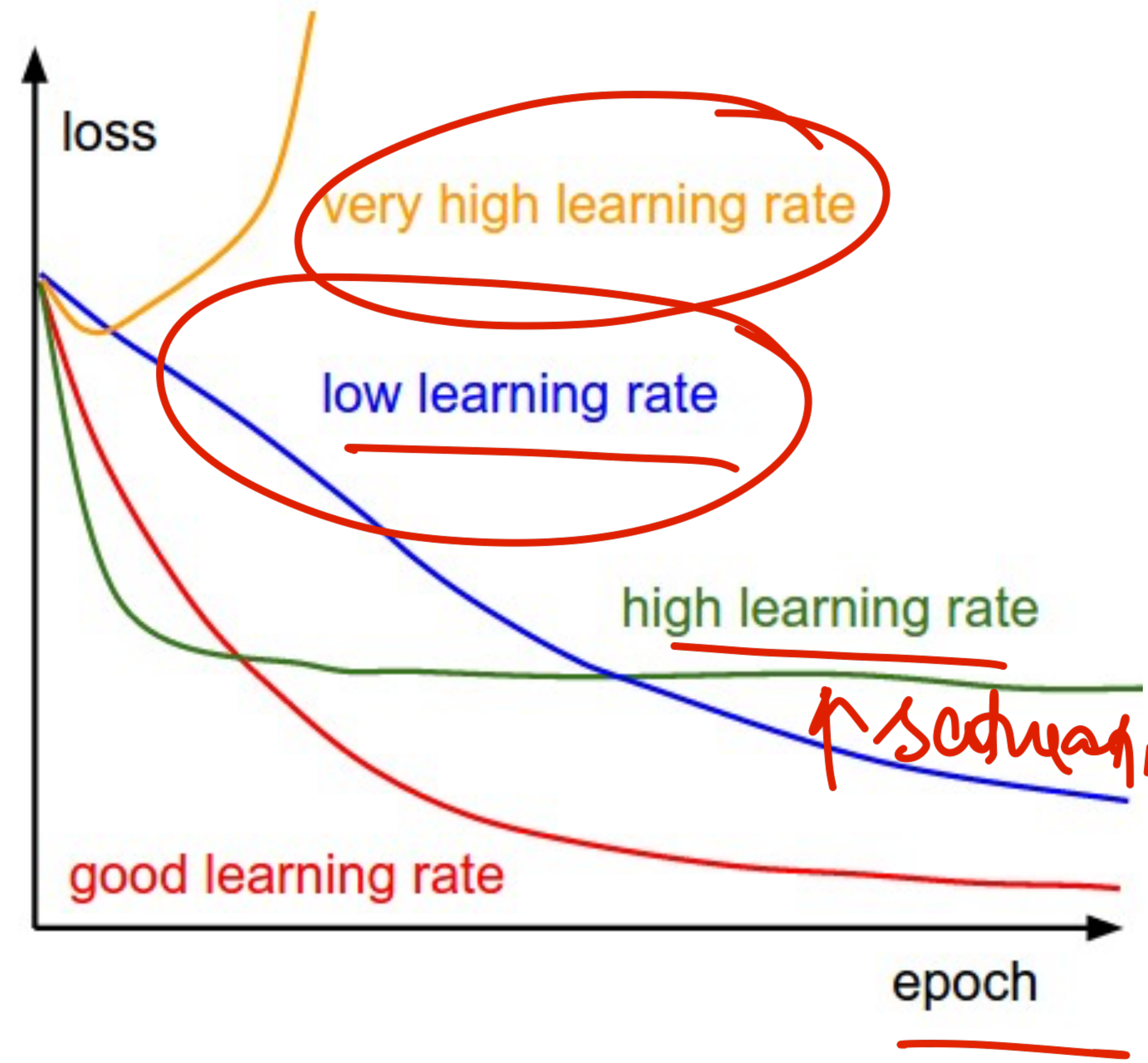
Learning Using Gradient Descent





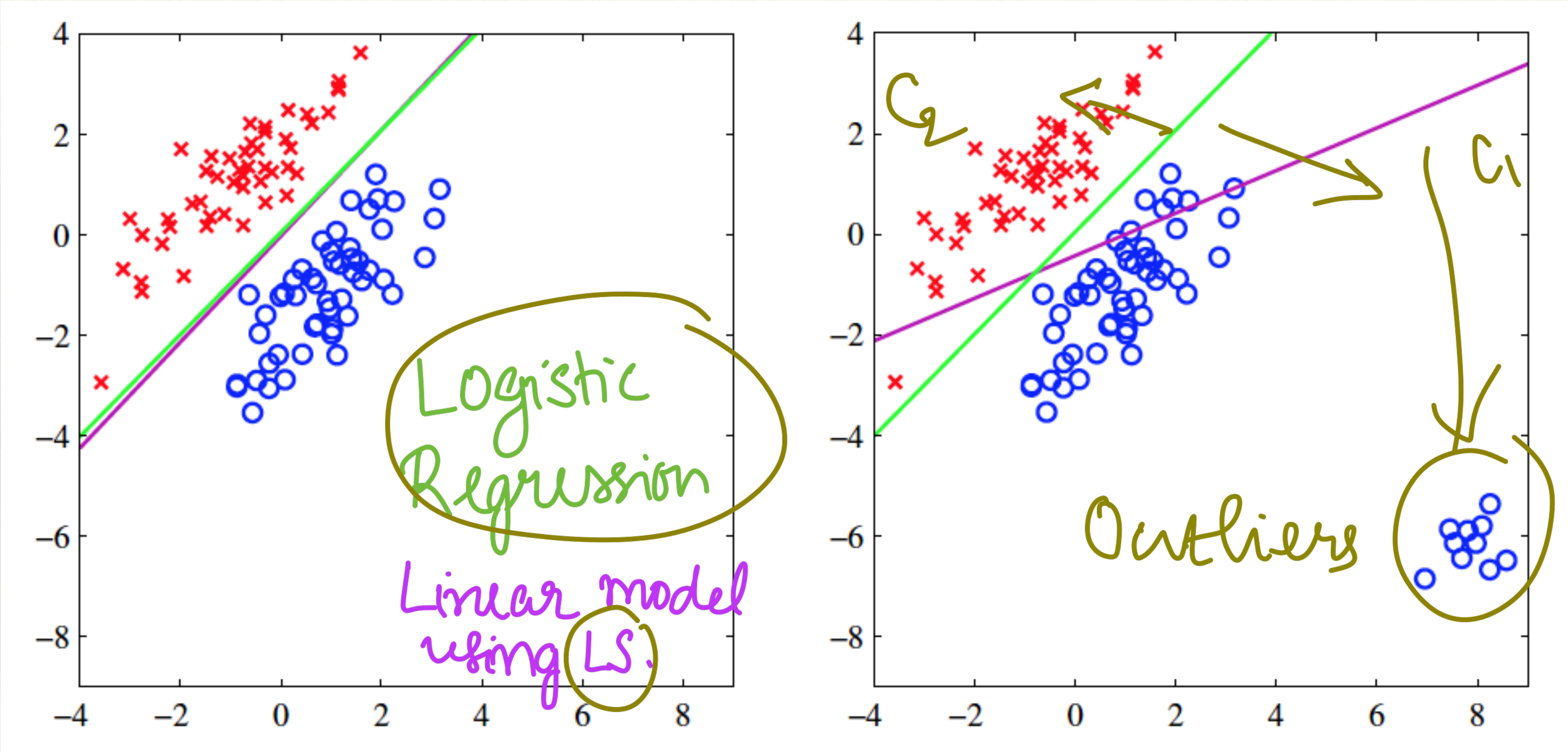
Parameter Learning

- Solving a non-convex optimization.
- Iterative solution.
- Depends on the initialization.
- Convergence to a local optima.
- Judicious choice of learning rate



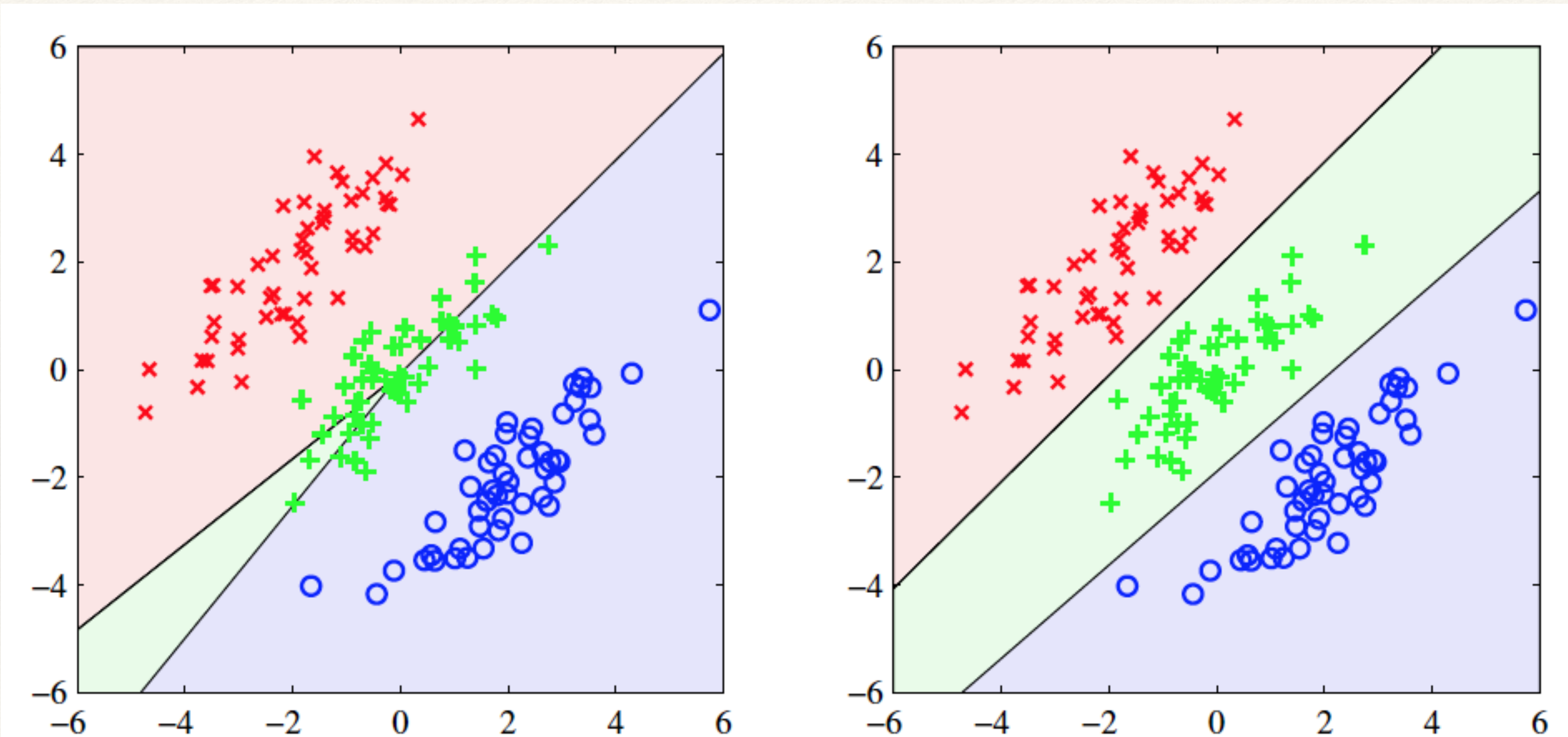
iteration

Least Squares versus Logistic Regression



Bishop - PRML book (Chap 4)

Least Squares versus Logistic Regression

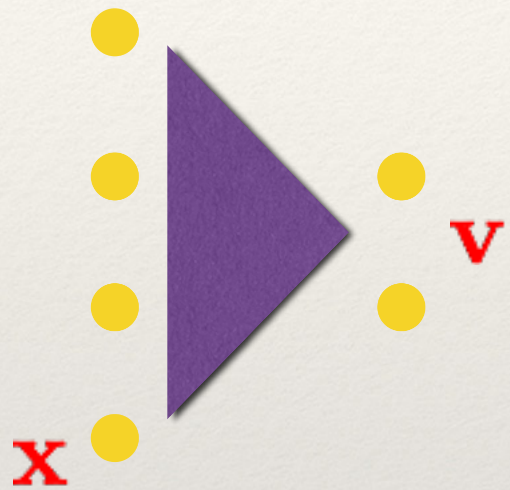


Bishop - PRML book (Chap 4)

Neural Networks

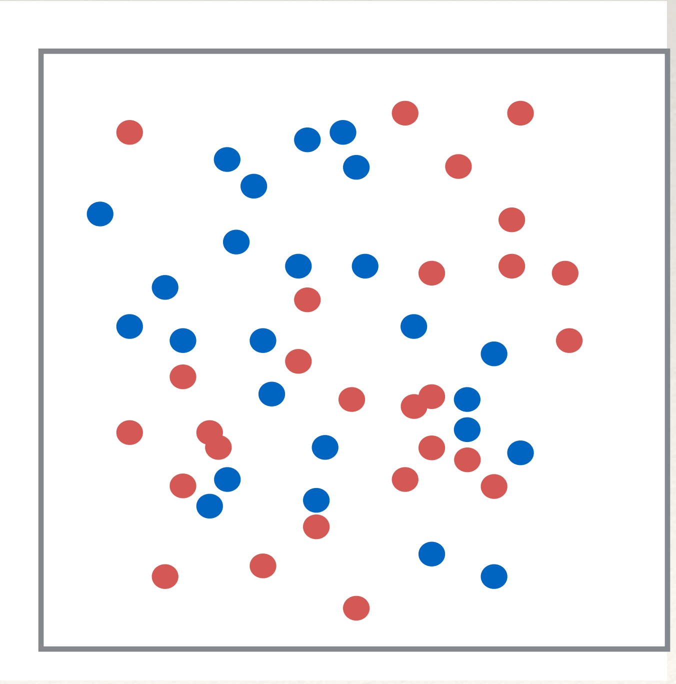
Perceptron Algorithm

Perceptron Model [McCulloch, 1943, Rosenblatt, 1957]



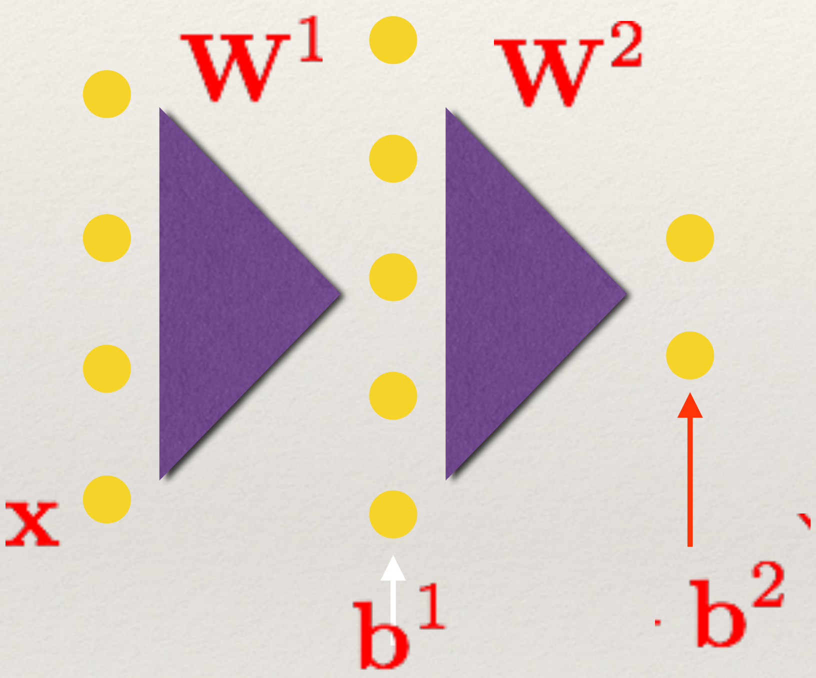
Targets are binary classes [-1,1]

What if the data is not linearly separable



Multi-layer Perceptron

Multi-layer Perceptron [Hopfield, 1982]



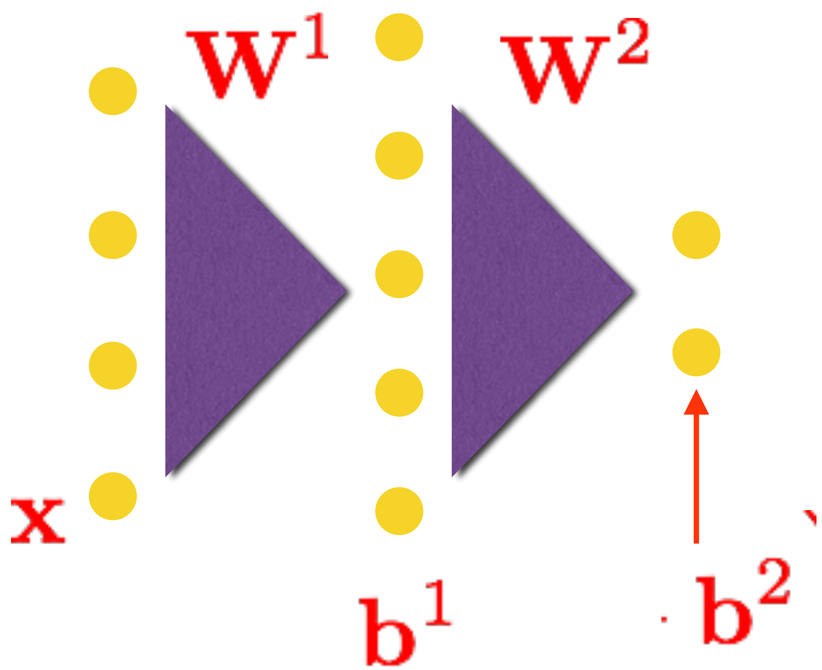
$$\mathbf{v}^2 = \psi\left(\mathbf{W}^2\phi(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2\right)$$

ϕ non-linear function (*tanh*,*sigmoid*)

ψ thresholding function

Neural Networks

Multi-layer Perceptron [Hopfield, 1982]



$$\mathbf{v}^2 = \psi\left(\mathbf{W}^2\phi(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2\right)$$

ϕ non-linear function (*tanh*,*sigmoid*)

ψ thresholding function

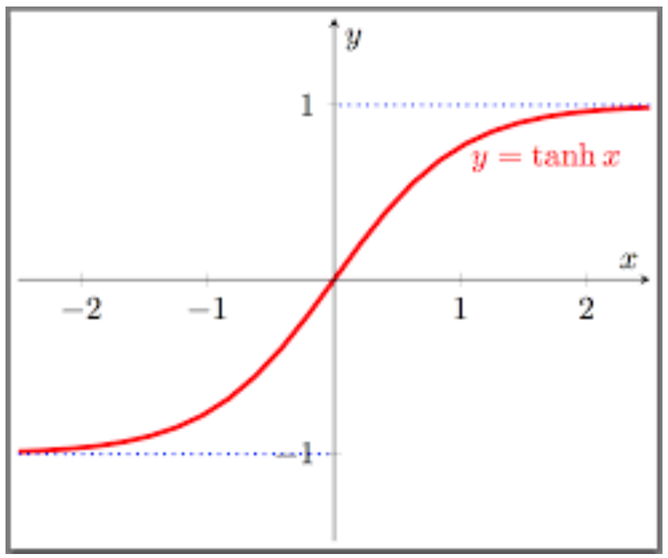
- Useful for classifying non-linear data boundaries - non-linear class separation can be realized given enough data.

Established 1911

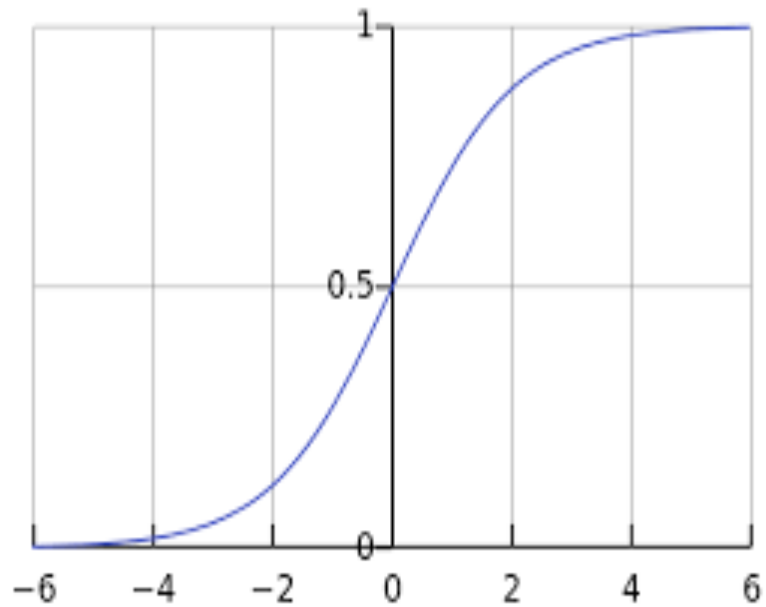
Neural Networks

Types of Non-linearities ϕ

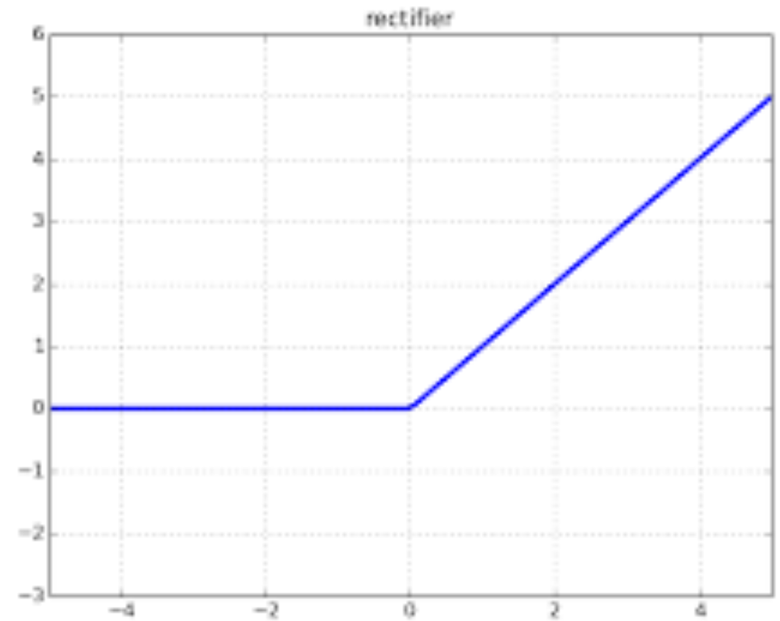
tanh



sigmoid



ReLu



Cost-Function

Mean Square Error

$$J_{MSE} = \sum_{i=1}^{M} ||\mathbf{v}_i - \mathbf{y}_i||^2$$

Cross Entropy

$$J_{CE} = -\sum_{i=1}^{M} \mathbf{y}_i^T \log(\mathbf{v}_i)$$

$\mathbf{y}_i$ are the desired outputs

Learning Posterior Probabilities with NNs

Choice of target function ψ

- Softmax function for classification

$$\psi(v_i) = \frac{e^{v_i}}{\sum_i e^{v_i}}$$

- Softmax produces positive values that sum to 1
- Allows the interpretation of outputs as posterior probabilities