Deep Learning - Theory and Practice

Deep Neural Networks

12-03-2020

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

deeplearning.cce2020@gmail.com

Established 1911

INDIAN INSTITUTE OF SCIENCE

भारतीय विज्ञान संस्थान

Logistic Regression

- 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

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

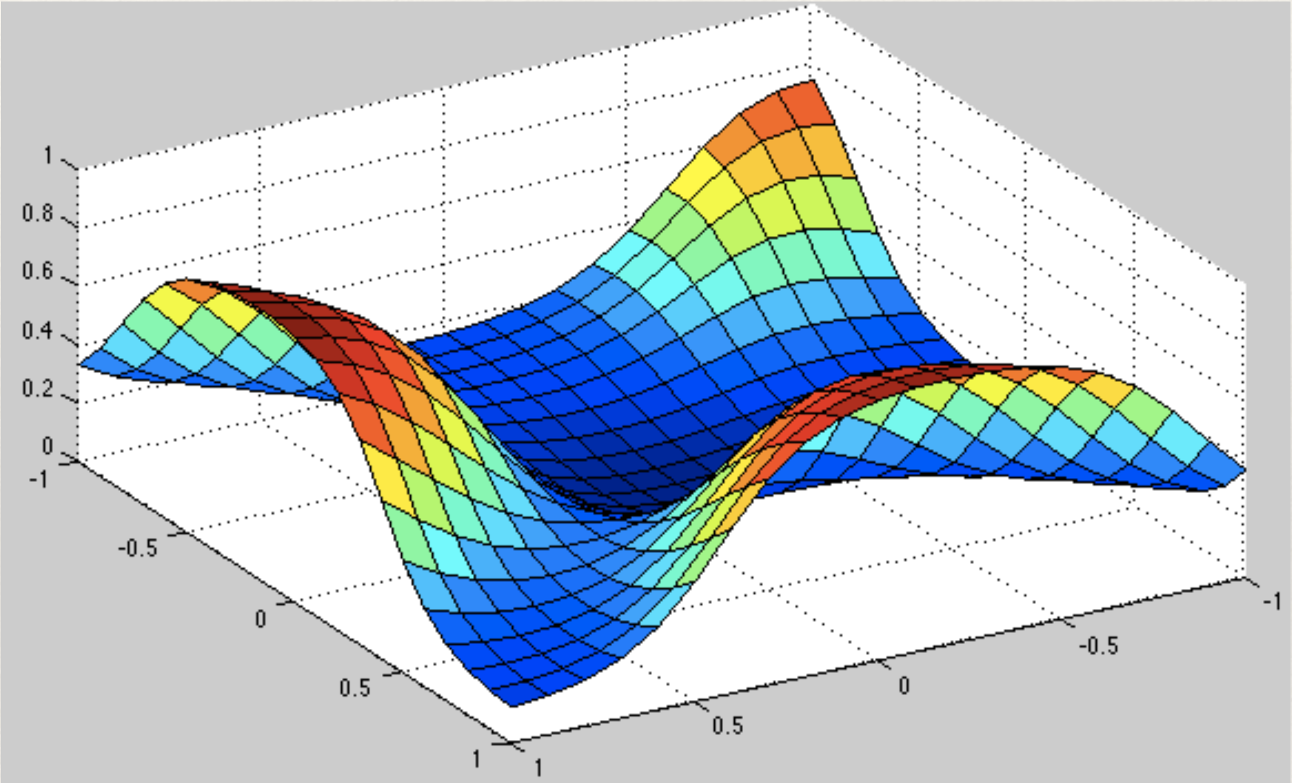
Established 1911

Bishop - PRML book (Chap 3)

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

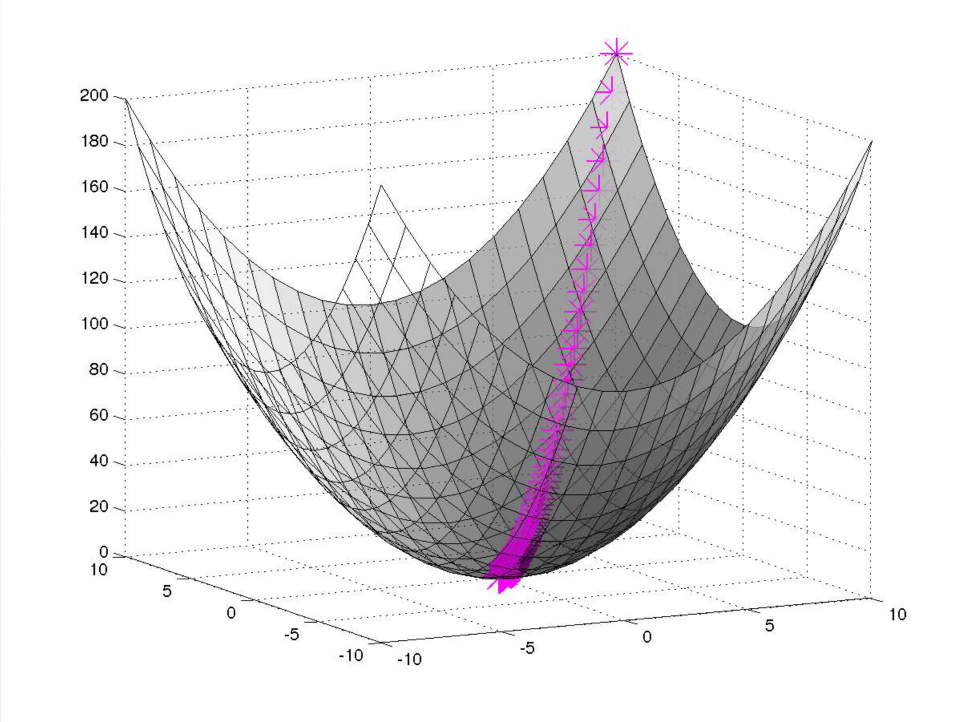
Typical Error Surfaces

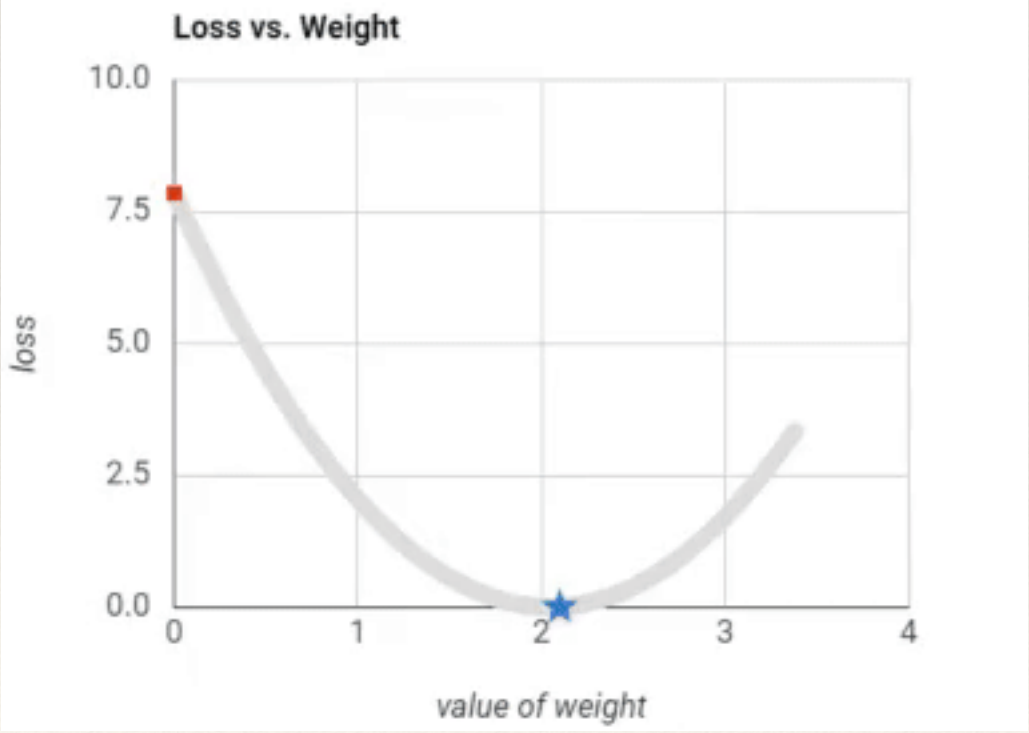
Typical Error Surface as a function of parameters (weights and biases)



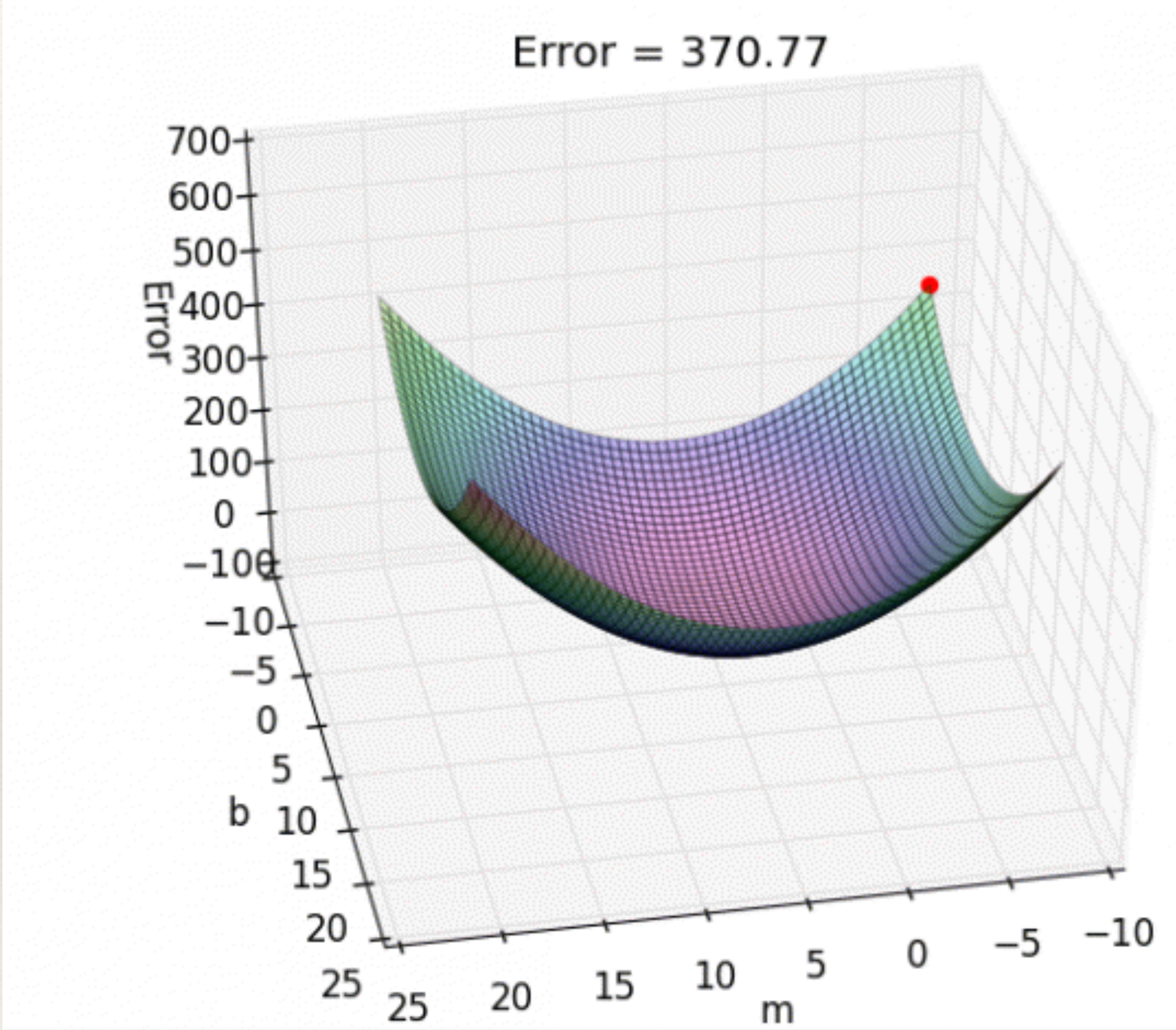
Learning with Gradient Descent

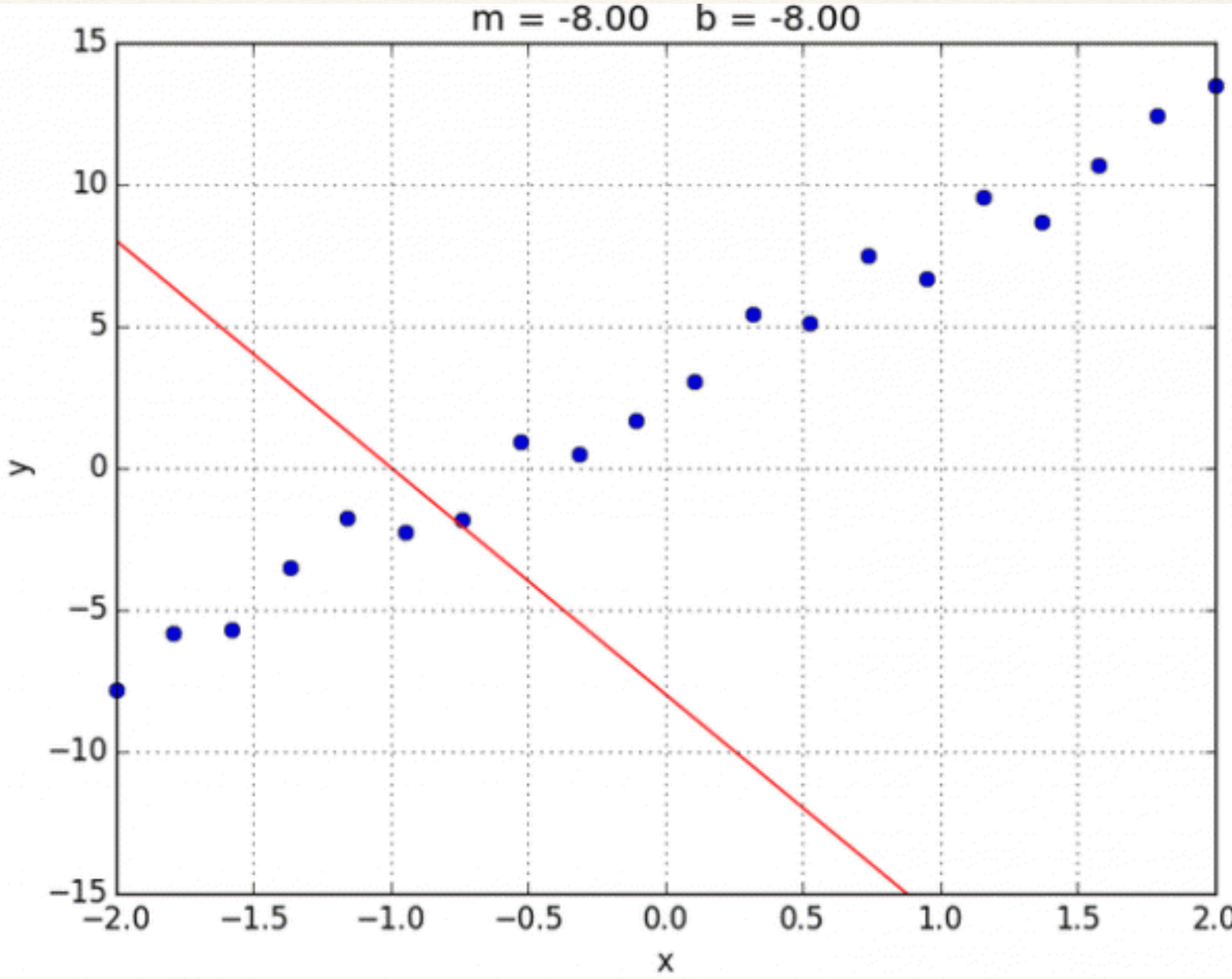
Error surface close to a local





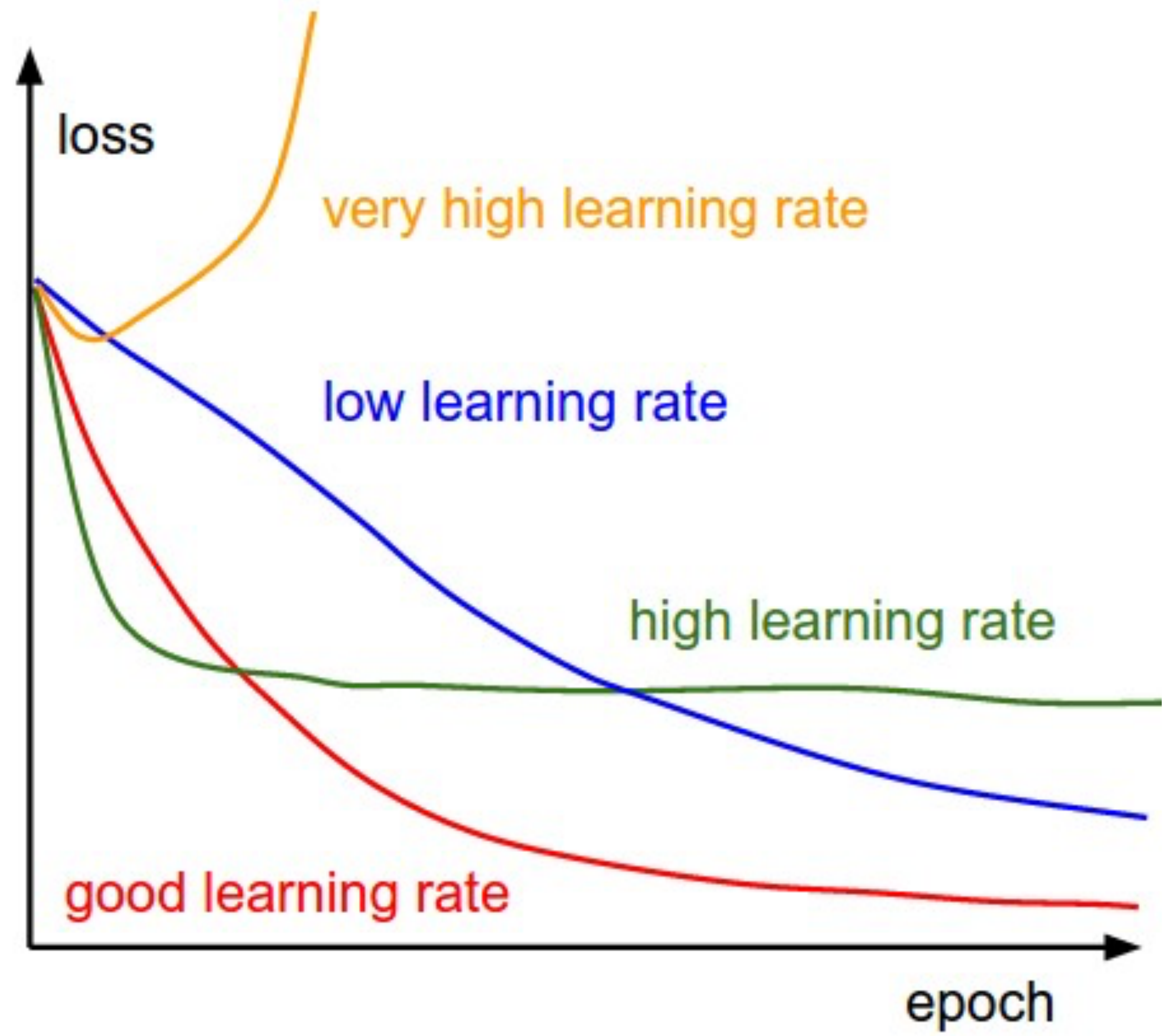
Learning Using Gradient Descent



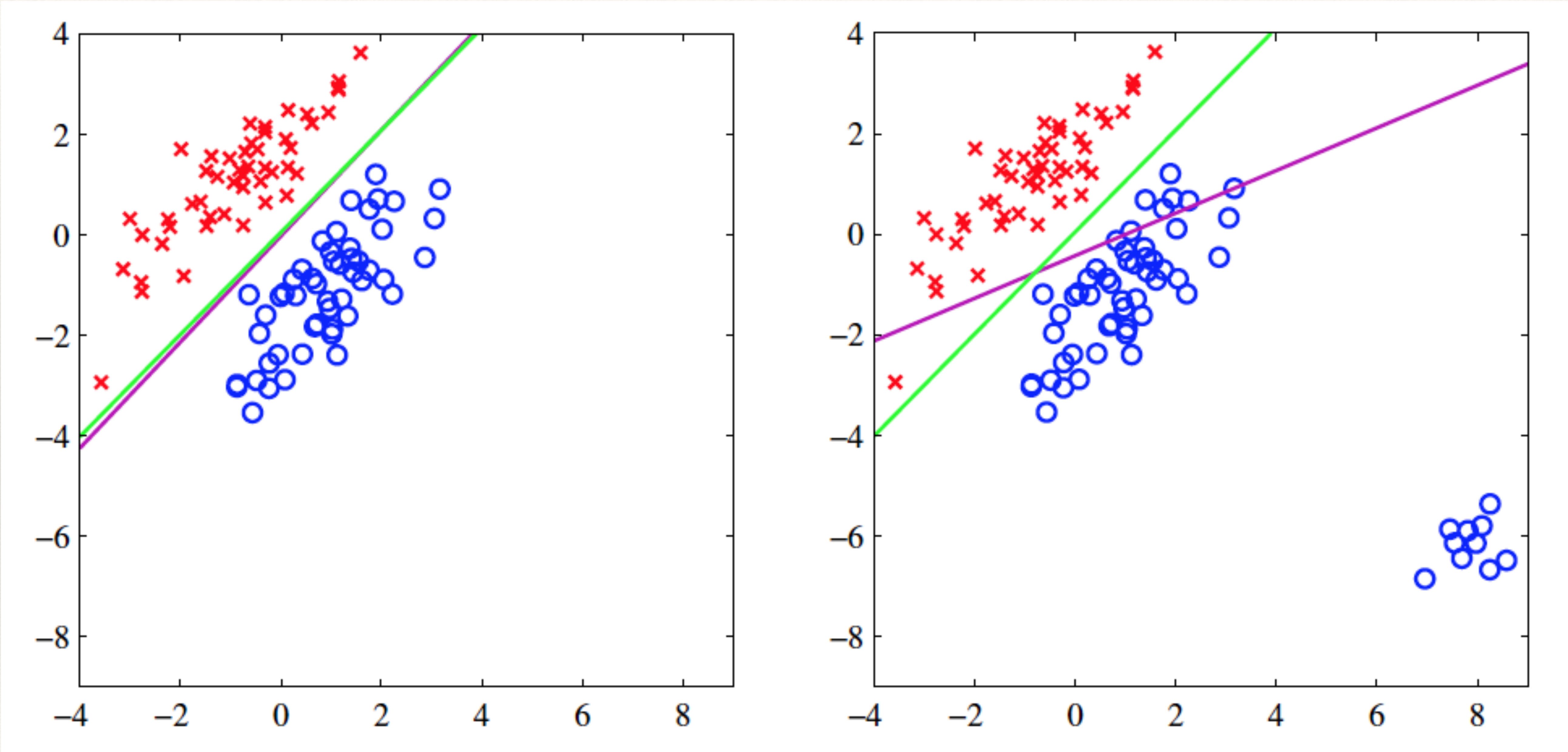


Parameter Learning

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

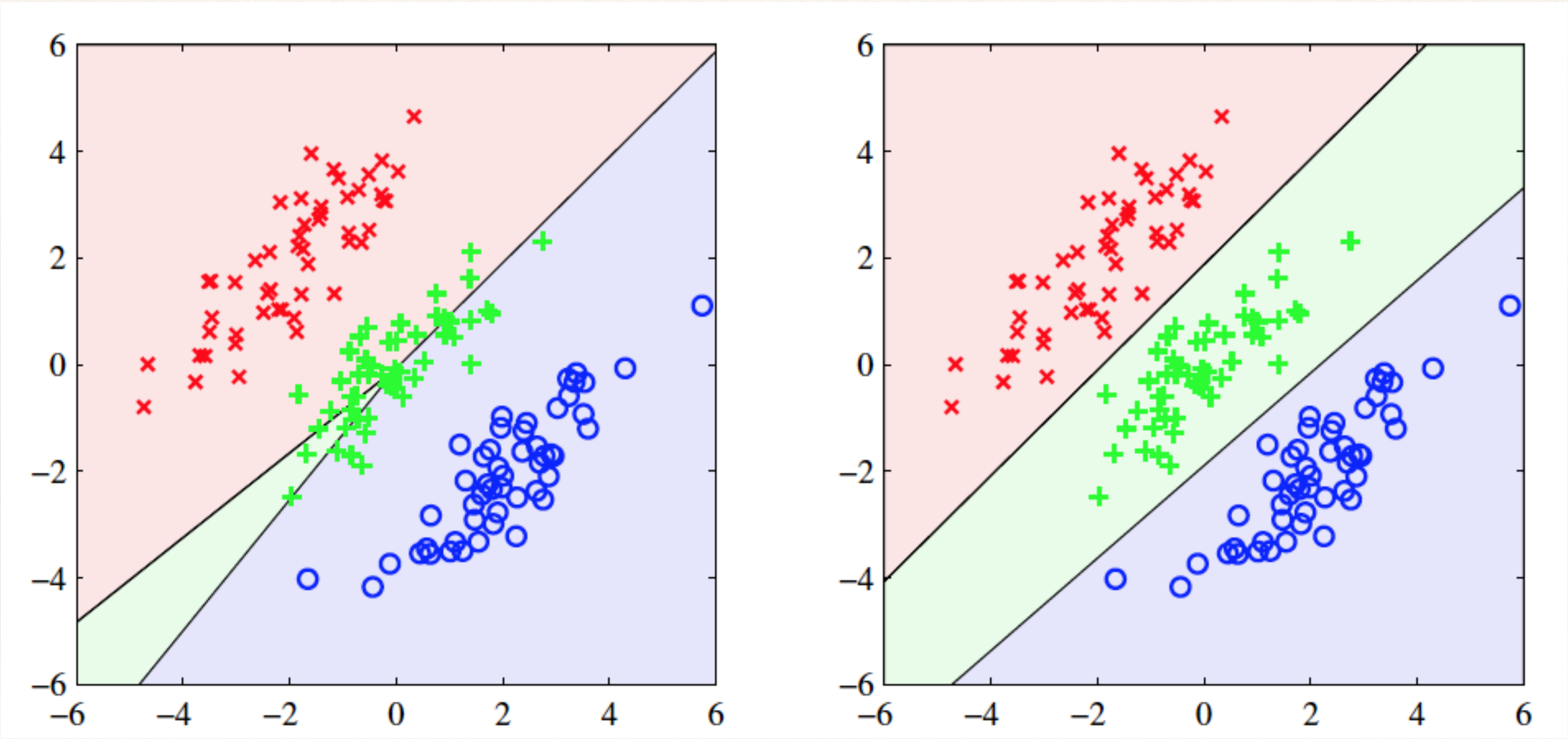


Least Squares versus Logistic Regression



Bishop - PRML book (Chap 4)

Least Squares versus Logistic Regression

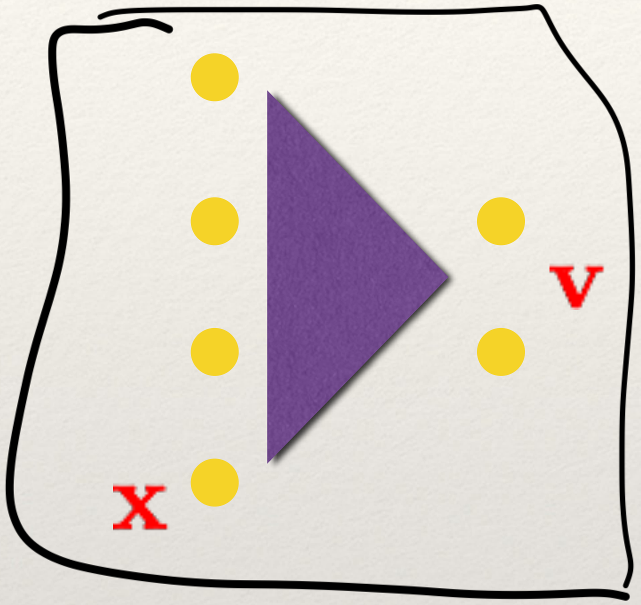


Bishop - PRML book (Chap 4)

Neural Networks

Perceptron Algorithm

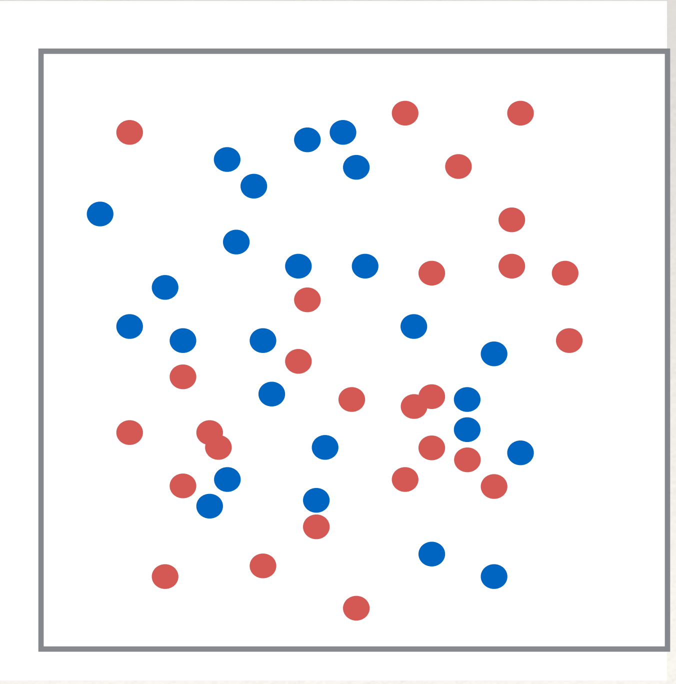
Perceptron Model [McCulloch, 1943, Rosenblatt, 1957]



Similar to the logistic regression

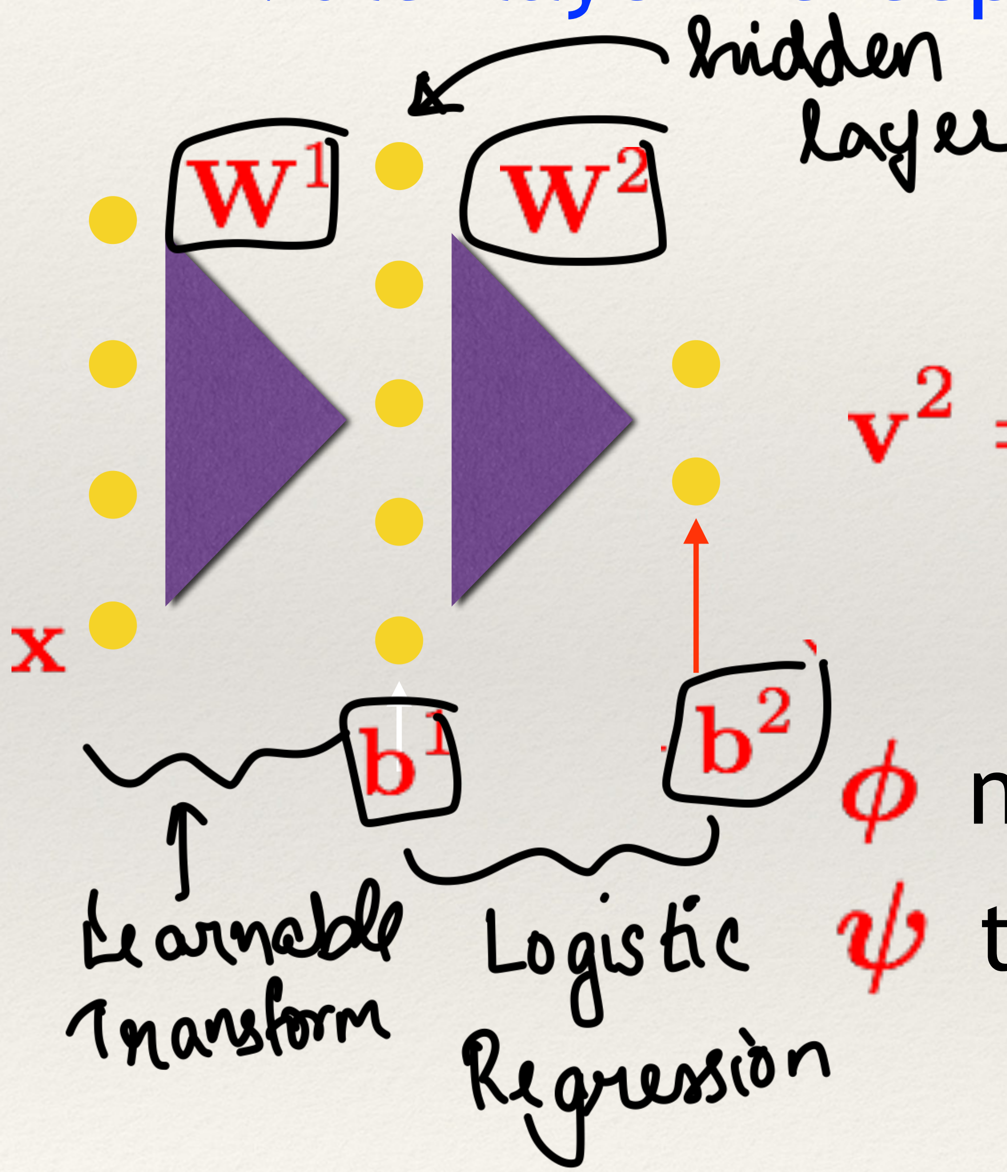
Targets are binary classes [-1,1]

What if the data is not linearly separable



Multi-layer Perceptron (MLP)

Multi-layer Perceptron [Hopfield, 1982]



hidden layer activation

$$\mathbf{v}^2 = \underbrace{\psi}_{\text{decision}}\left(\mathbf{W}^2 \underbrace{\phi(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2}_{\text{Learnable transform}}\right)$$

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

ψ thresholding function

$\phi(\underline{x})$

$$x \xrightarrow[\text{learnable}]{} \phi(x) \leftarrow \text{vector valued}$$

$\phi(x) \longrightarrow y$ [outputs]

↑ optimization done in a logistic regression

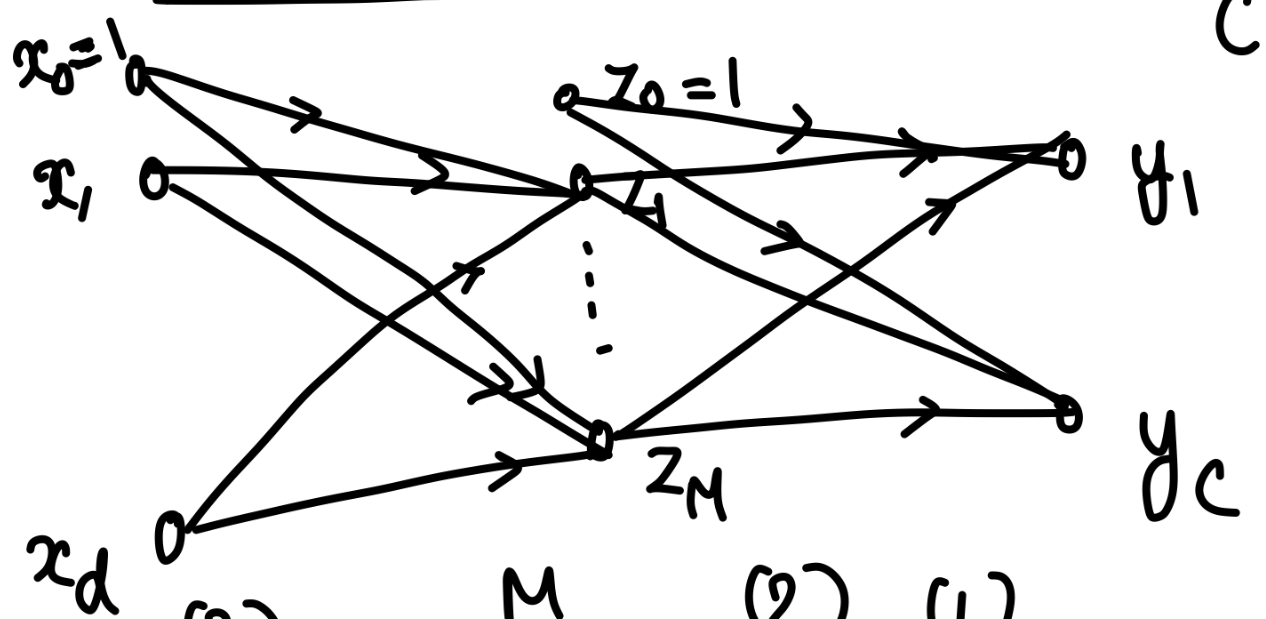
[Cost function - CE
Regression - linear - M.S.E]

Learning [S.G.D]

→ Previously only W^2, b^2 were the variables

→ $\{W^1, b^1, W^2, b^2\}$ variables in this model.

Back propagation [Neural Networks, C. Bishop CHAP-4]



Forward Direction

$$a_j^{(1)} = \sum_{i=0}^{d} w_{ji}^{(1)} x_i \qquad j = 1 \dots M$$

$$z_j^{(1)} = g(a_j)$$

g : non-linear activation.

$$a_k^{(2)} = \sum_{j=0}^{M} w_{kj}^{(2)} z_j^{(1)}$$

$$y_k = \tilde{g}(a_k^{(2)})$$

$\{x_1 \; x_2 \dots x_n \dots\}_N$

t_k^n = target for k^{th} node for x_n $\{t_1, t_2 \dots t_N\}$

$$E = \sum_n E^n$$

; All functions in the forward pass are differentiable.

Gradient descent : $\frac{\partial E^n}{\partial w_{ji}^l}$ $l = \{1, 2\}$

Let $\delta_j^l = \frac{\partial E^n}{\partial a_j^l}$: $\frac{\partial a^l}{\partial w_{ji}^l} = z_i^{l-1}$ $\boxed{z_i^0 = x_i}$

$$\boxed{\frac{\partial E^n}{\partial w_{ji}^l}} = \boxed{\frac{\partial E^n}{\partial a_j^l}} \cdot \frac{\partial a_j^l}{\partial w_{ji}^l} = \boxed{\boxed{\delta_j^l} \cdot z_i^{l-1}}$$

Deep networks >1 hidden layers

$$\boxed{\begin{aligned} a_j^l &= \sum_i \boxed{w_{ji}^l} z_i^{l-1} \\ z_j^l &= g(a_j^l) \end{aligned}}$$

For $l = L$ { L is number of layers}

$$\boxed{\delta_k^L} = \frac{\partial E^n}{\partial a_k^L} = \tilde{g}'(a_k^L) \cdot \frac{\partial E^n}{\partial y_k^n} \Big\}.$$

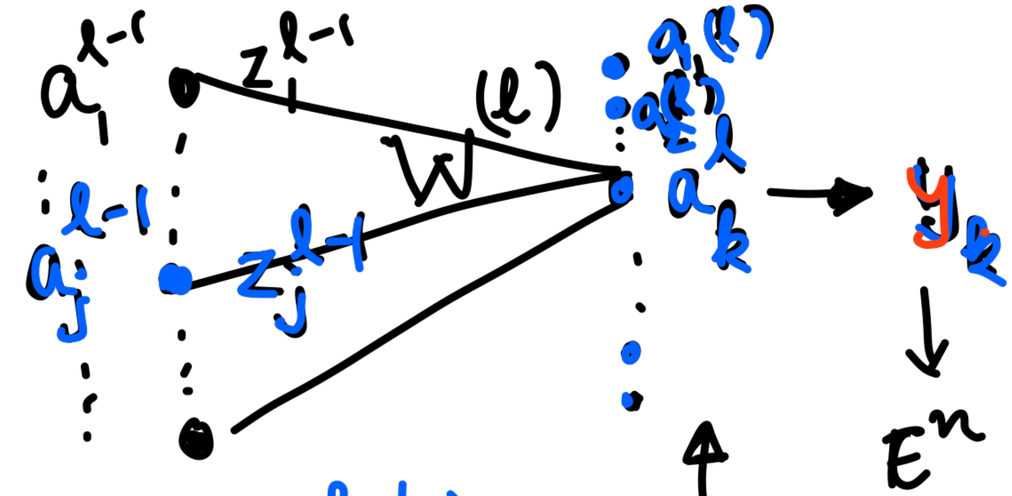
$\frac{\partial E^n}{\partial w_{ji}^L}$

Example

$$E^n = -\sum_{k=1}^{C} t_k^n \log y_k^n + \lambda R(W)$$

Now, for any layer $L, \ldots, l, \ldots 2$

$$\delta_j^{l-1} \longleftarrow \delta_k^{l}$$



$$\delta_j^{l-1} = \frac{\partial \mathcal{E}^n}{\partial a_j^{l-1}} = \frac{\partial \mathcal{E}^n}{\partial z_j^{l-1}} \cdot \frac{\partial z_j^{l-1}}{\partial a_j^{l-1}}$$

$$z_j^{l-1} = g\left(a_j^{l-1}\right)$$

$$= \sum_k \frac{\partial \mathcal{E}}{\partial a_k^{l}} \cdot \frac{\partial a_k^{l}}{\partial z_j^{l-1}} \, g'(a_j^{l-1})$$

$$a_k^{l} = \sum_j w_{kj}^{(l)} z_j^{l-1}$$

$$\boxed{\frac{\partial \mathcal{E}^n}{\partial a_k^{l}}} \ \forall k.$$

$$= \sum_k \delta_k^{l} w_{kj}^{(l)} \boxed{g'\left(a_j^{l-1}\right)}$$

Backward pass.

$$\left\{w_{kj}^{l}\right\}^{t} = \left\{w_{kj}^{l}\right\}^{t-1} - \eta \frac{\partial \mathcal{E}}{\partial w_{kj}^{l}}\Bigg|_{\left\{w_{kj}^{l} = w_{kj}^{l}\right\}^{t-1}}$$

Update equation

t-iteration

Summary

Initial set of weights.

(1) $\underline{x}^n \longrightarrow$ find activations $a_k^\ell, z_k^\ell \;\forall \ell$.

$$z_i^0 = x_i \quad ; \quad z_k^L = y_k.$$

(2) Evaluate δ at L, then apply recursion

$$\delta_k^\ell \longrightarrow \delta_j^{\ell-1} \qquad \ell = L, L-1, \ldots 2.$$

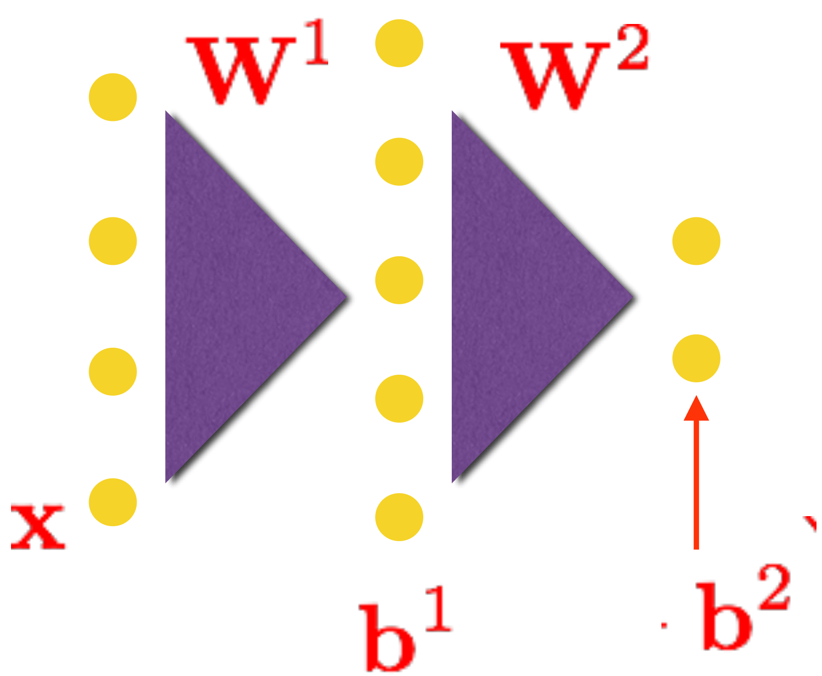
(3) Compute derivatives of weight and update.

Repeat until convergence

* fixed number of epochs
* Error in validation reaching saturation

Neural Networks

Multi-layer Perceptron [Hopfield, 1982]



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

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.

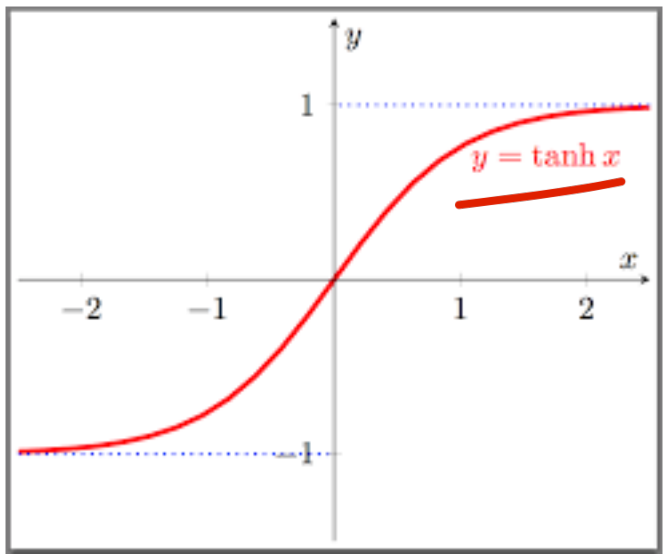
Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

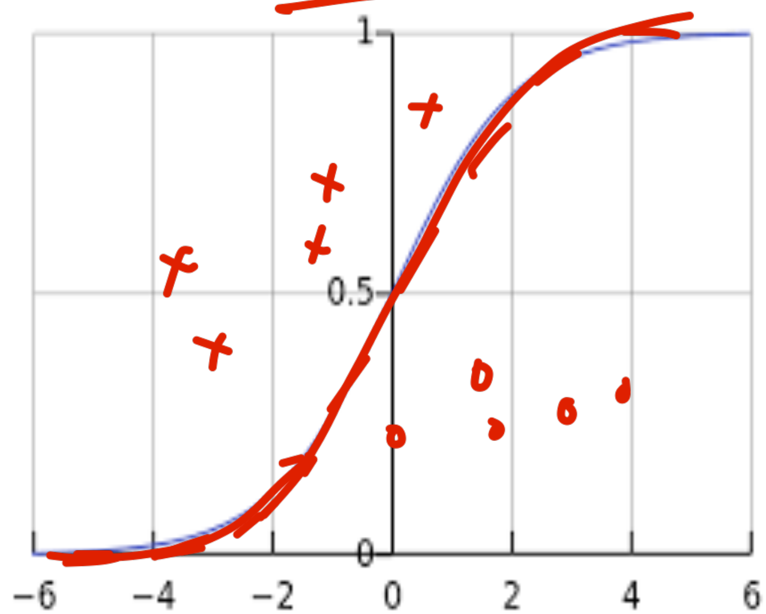
Neural Networks

Types of Non-linearities ϕ (g)

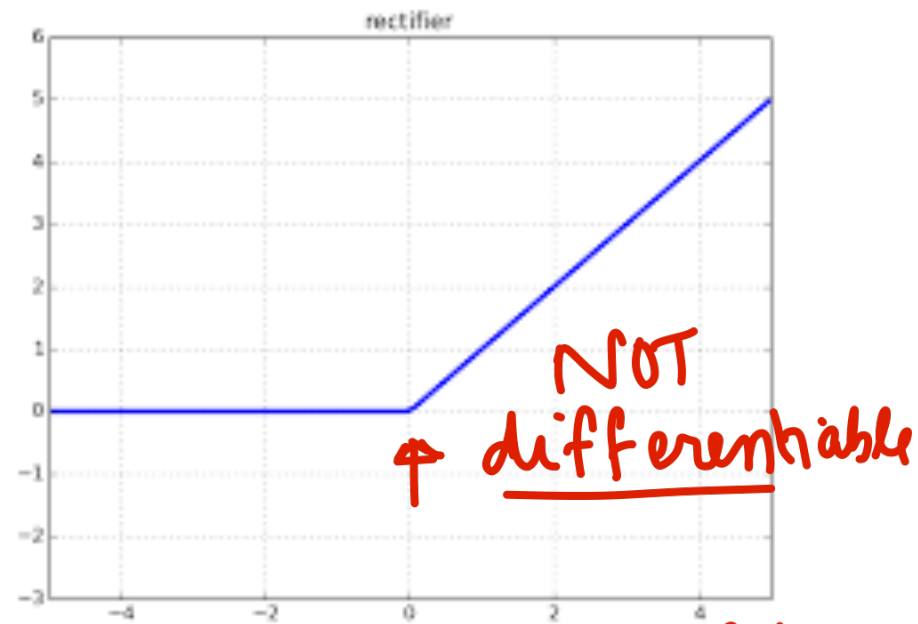
tanh



sigmoid



$\boxed{\text{ReLu}}$



$z_k^{l} = a_k^{l-1}$ if $a_k^{l-1} > 0$

$= 0$ otherwise

Cost-Function

Mean Square Error

Cross Entropy

$\underline{a}^{l-1} = \underline{W}^{(l-1)} \underline{z}^{(l-2)}$

$$E \; \underline{J_{MSE}} = \sum_{i=1}^{M} ||\mathbf{v}_i - \mathbf{y}_i||^2 \qquad E \; J_{CE} = -\sum_{i=1}^{M} \mathbf{y}_i^T \log(\mathbf{v}_i)$$

$\mathbf{y}_i$ 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

Need For Deep Networks

Modeling complex real world data like speech, image, text

- Single hidden layer networks are too restrictive.
- Needs large number of units in the hidden layer and trained with large amounts of data.
- Not generalizable enough.

Networks with multiple hidden layers - deep networks (Open questions till 2005)

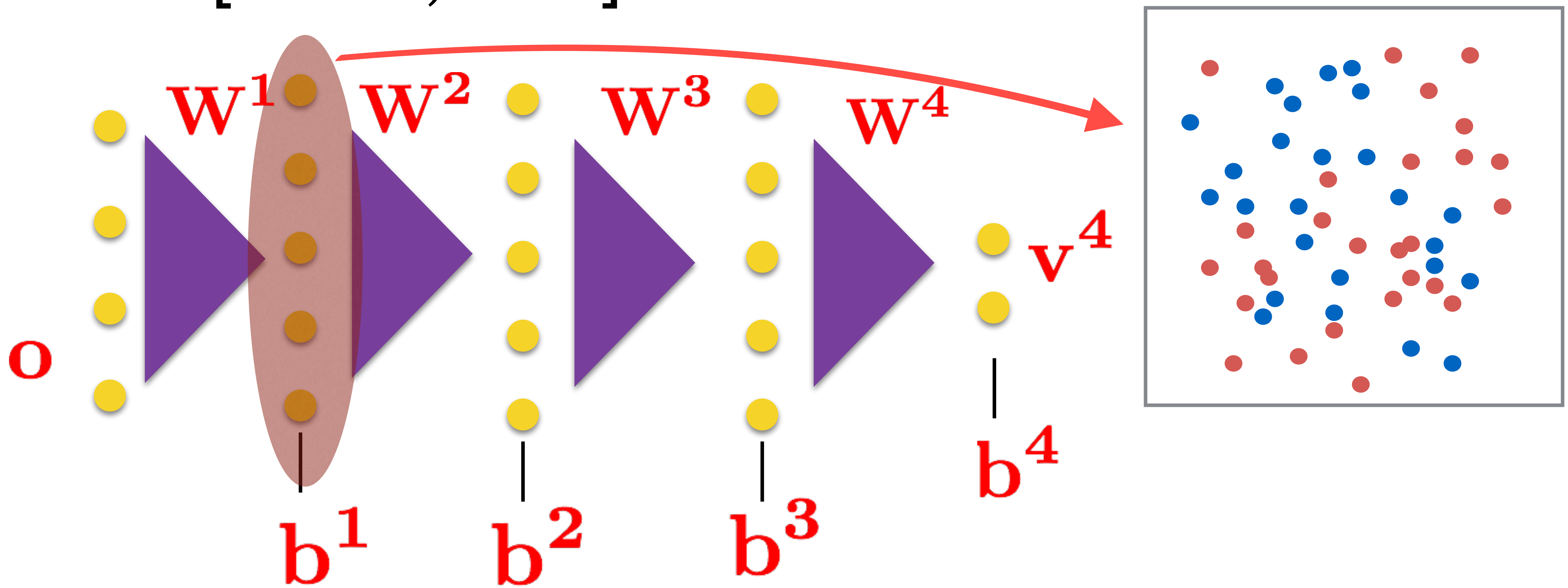
- Are these networks trainable ?
- How can we initialize such networks ?
- Will these generalize well or over train ?

Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

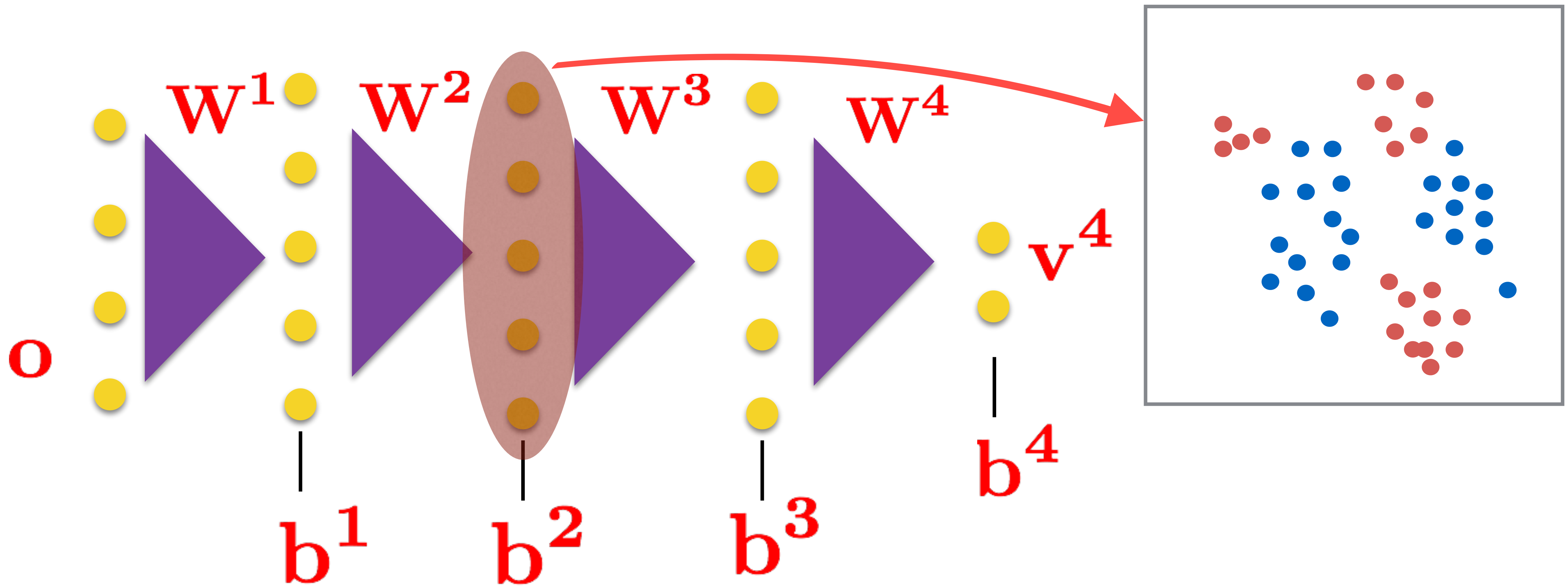
Deep Networks Intuition

Neural networks with multiple hidden layers - Deep networks [Hinton, 2006]



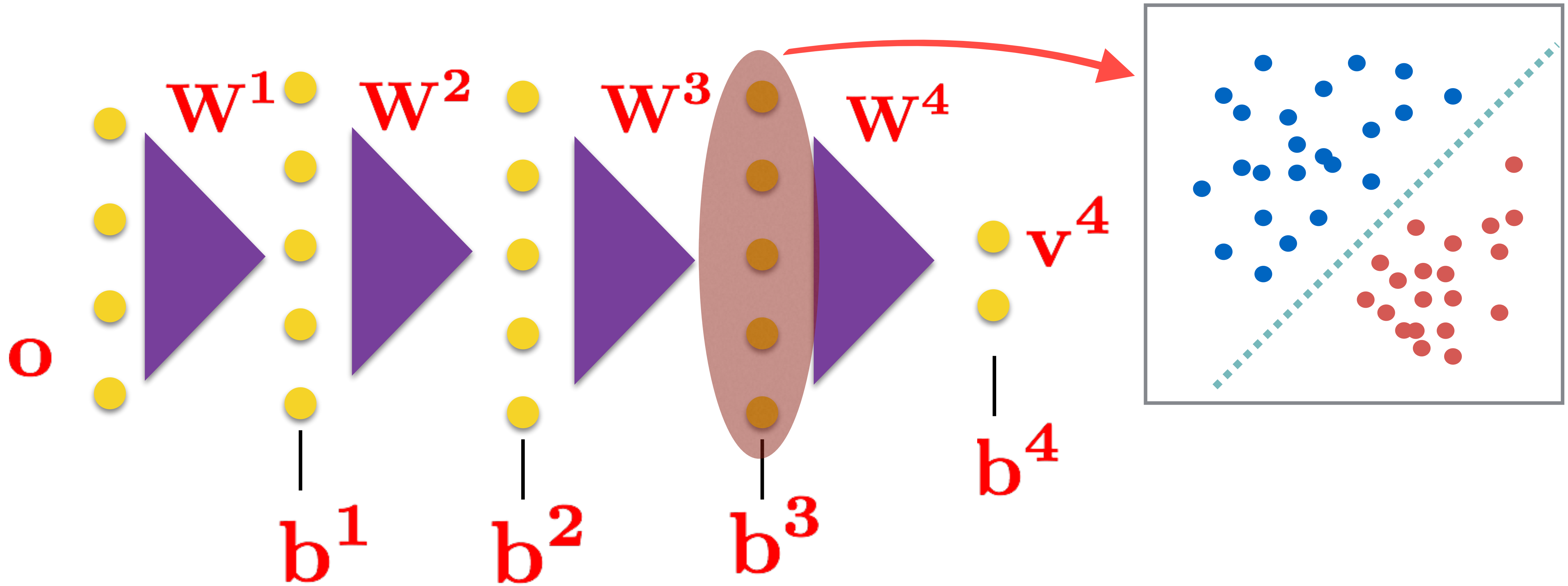
Deep Networks Intuition

Neural networks with multiple hidden layers - Deep networks



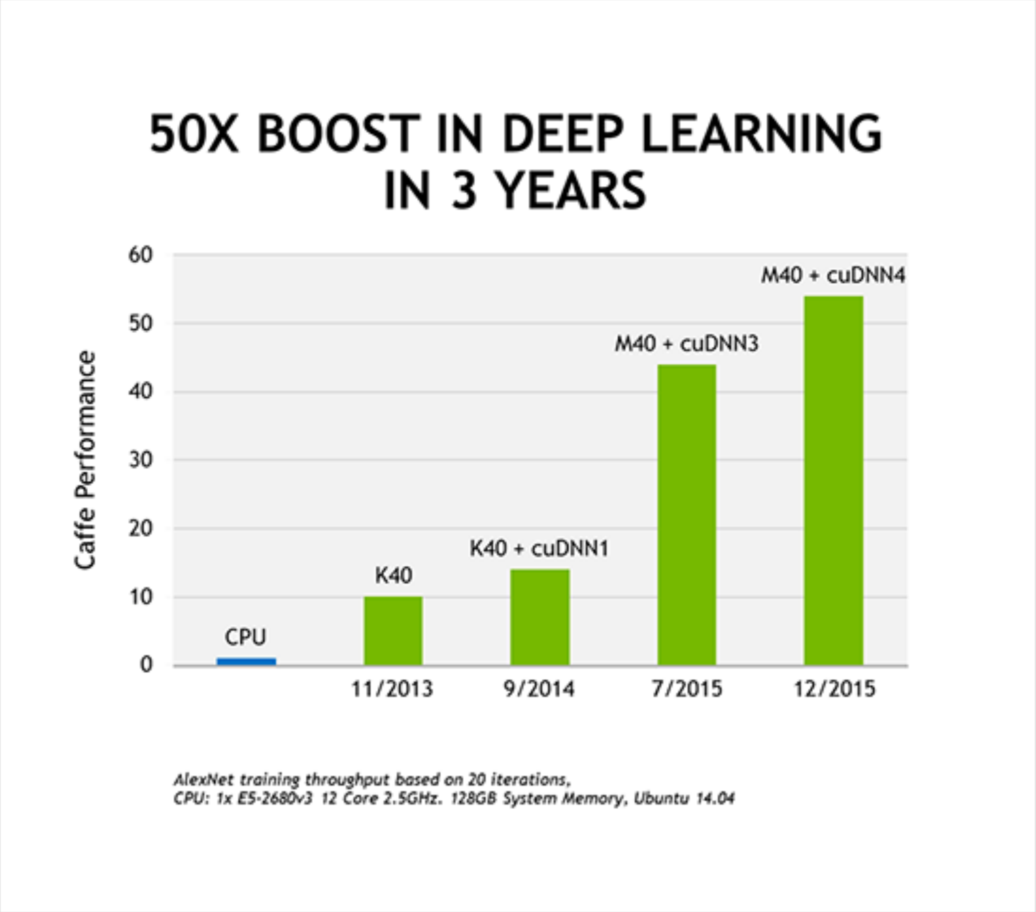
Deep Networks Intuition

Neural networks with multiple hidden layers - Deep networks



Deep networks perform hierarchical data abstractions which enable the non-linear separation of complex data samples.

Deep Networks



- Are these networks trainable ?

- Advances in computation and processing
- Graphical processing units (GPUs) performing multiple parallel multiply accumulate operations.
- Large amounts of supervised data sets

Established
1911