MACHINE LEARNING FOR SIGNAL PROCESSING

17-2-2025

Sriram Ganapathy
LEAP lab, Electrical Engineering, Indian Institute of Science,
sriramg@iisc.ac.in

Viveka Salinamakki, Varada R.
LEAP lab, Electrical Engineering, Indian Institute of Science

http://leap.ee.iisc.ac.in/sriram/teaching/MLSP25/

STORY SO FAR

EM algorithm

Gaussian Modeling

Gaussian Mixture Modeling

Decision Theory

Generative Modeling

Classification Problem

Function Modeling

Linear Models for Regression and Classification

Kernel Machines & Max-margin classifiers

Support Vector Machines

Data Representations PCA, LDA

Discriminative Modeling

Neural Networks

Gradient Descent

SVM Formulation

- Goal - **1) Correctly classify all training data**

$$\mathbf{w}^T\phi(\mathbf{x}_n) + b \geq 1 \quad if \quad t_n = +1$$
$$\mathbf{w}^T\phi(\mathbf{x}_n) + b \leq 1 \quad if \quad t_n = -1$$

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1$$

2) Define the Margin

$$\frac{1}{||\mathbf{w}||} min_n \; [t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)]$$

3) Maximize the Margin

$$argmax_{\mathbf{w},b}\left\{\frac{1}{||\mathbf{w}||} min_n \; [t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)]\right\}$$

- Equivalently written as

$$argmin_{\mathbf{w},b} \; \frac{1}{2}||\mathbf{w}||^2 \text{such that} \quad t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1$$

Solving the Optimization Problem

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* a_n is associated with every constraint in the primary problem:
- The dual problem in this case is maximized

Find $\{a_1, .., a_N\}$ such that

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} t_n t_m a_n a_m k(\mathbf{x}_n, \mathbf{x}_m) \text{ maximized}$$

and $\sum_n a_n t_n = 0, \quad a_n \geq 0$

Solving the Optimization Problem

- The solution has the form:

$$\mathbf{w} = \sum_{n=1}^{N} a_n \phi(\mathbf{x}_n)$$

- Each non-zero a_n indicates that corresponding $\mathbf{x_n}$ is a support vector. Let S denote the set of support vectors.

$$b = y(\mathbf{x}_n) - \sum_{m \in S} a_m k(\mathbf{x}_m, \mathbf{x}_n)$$

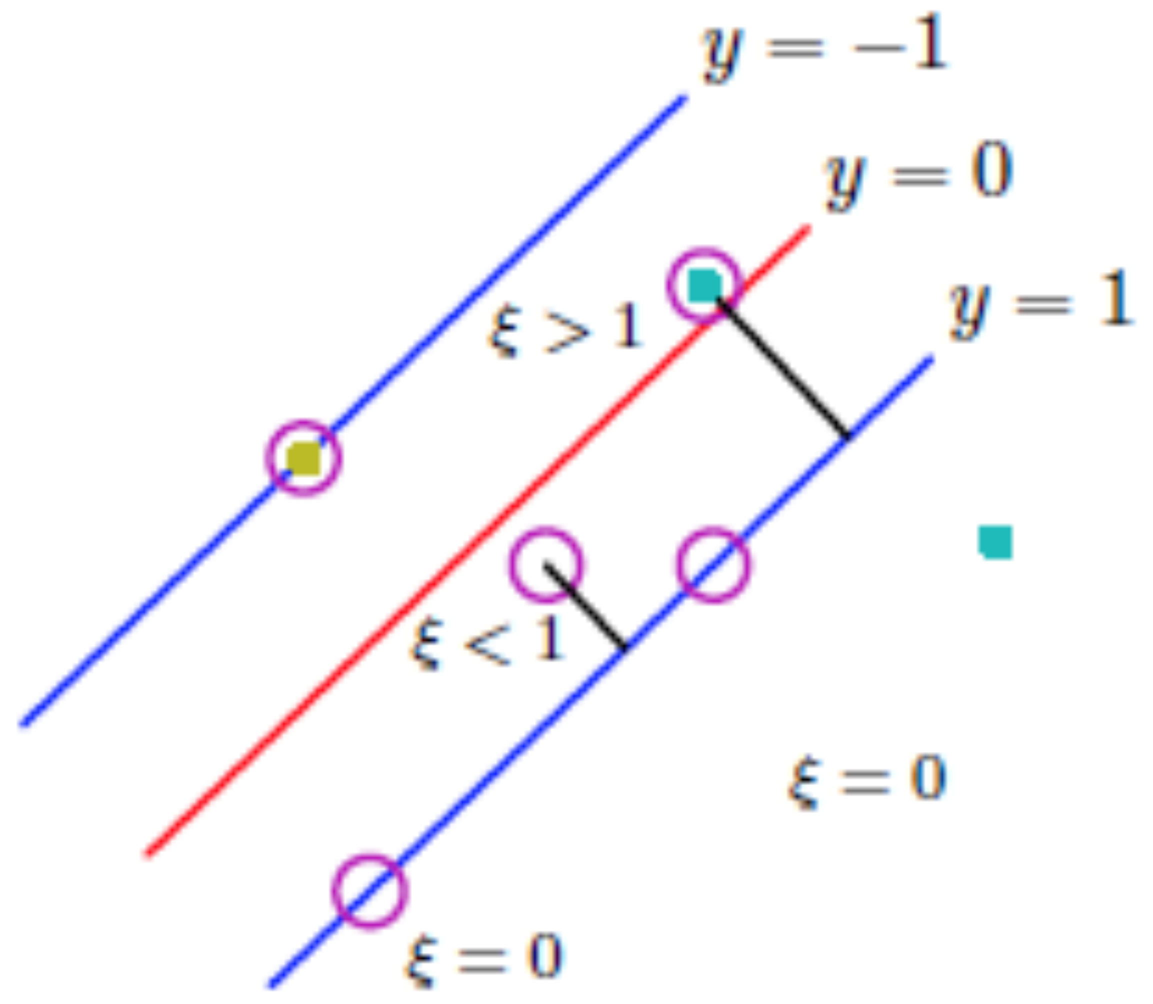
- And the classifying function will have the form:

$$y(\mathbf{x}) = \sum_{n \in S} a_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Overlapping class boundaries

- The classes are not linearly separable - Introducing slack variables ζ_n
- Slack variables are non-negative $\zeta_n \geq 0$
- They are defined using

$$t_n y(\mathbf{x}_n) \geq 1 - \zeta_n$$



- The upper bound on mis-classification

$$\sum_n \zeta_n$$

- The cost function to be optimized in this case

$$C \sum_n \zeta_n + \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

SVM Formulation - overlapping classes

- Formulation very similar to previous case except for additional constraints

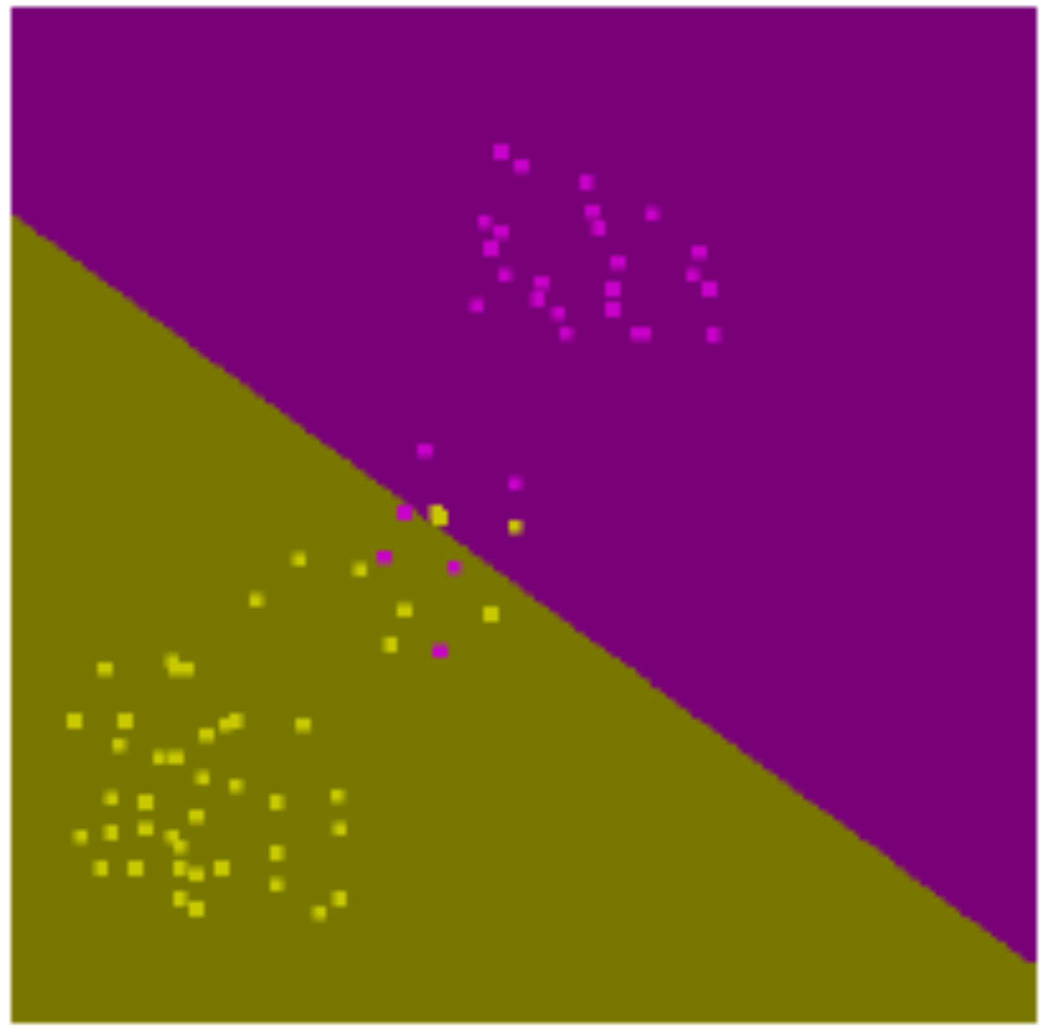
$$0 \leq a_n \leq C$$

- Solved using the dual formulation - sequential minimal optimization algorithm
- Final classifier is based on the sign of

$$y(\mathbf{x}) = \sum_{n \in S} a_n k(\mathbf{x}_n, \mathbf{x}) + b$$

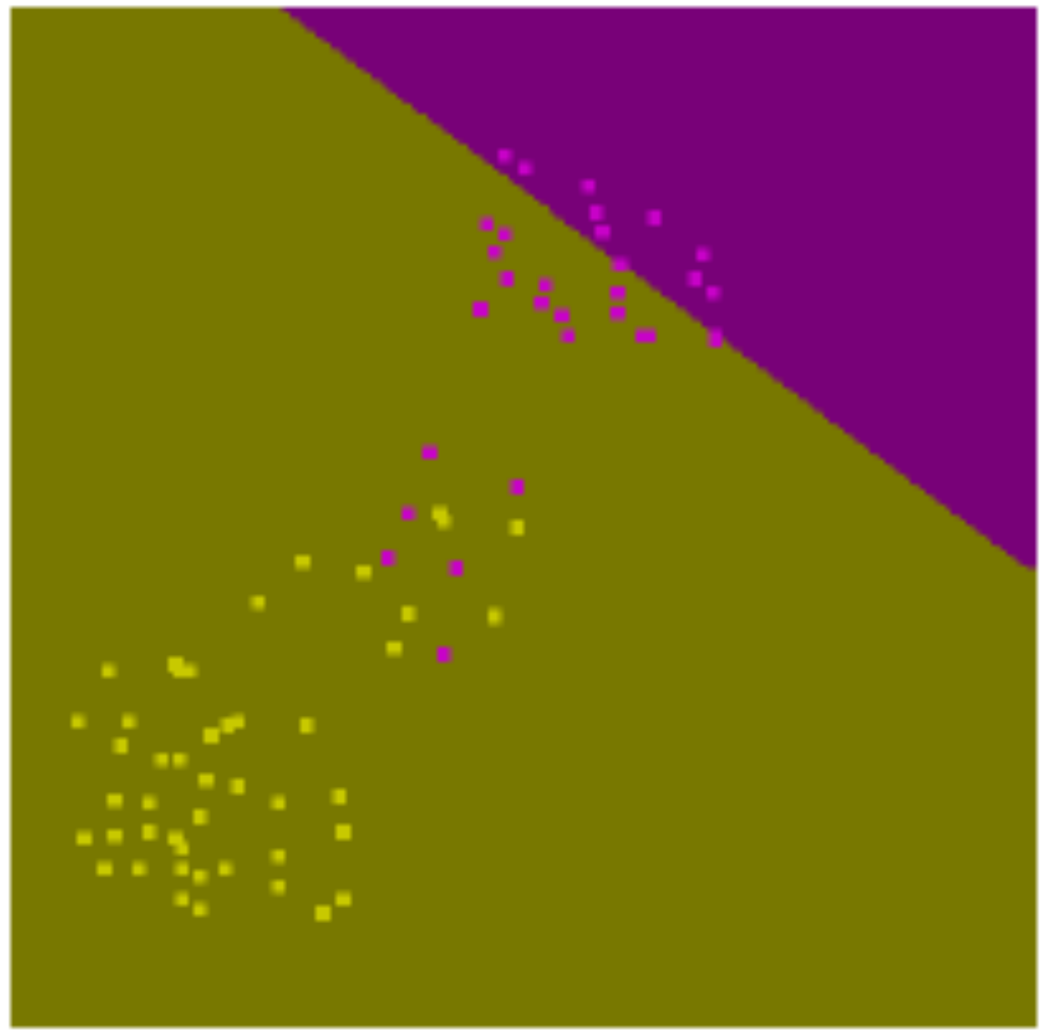
Overlapping class boundaries

C=100



C=1

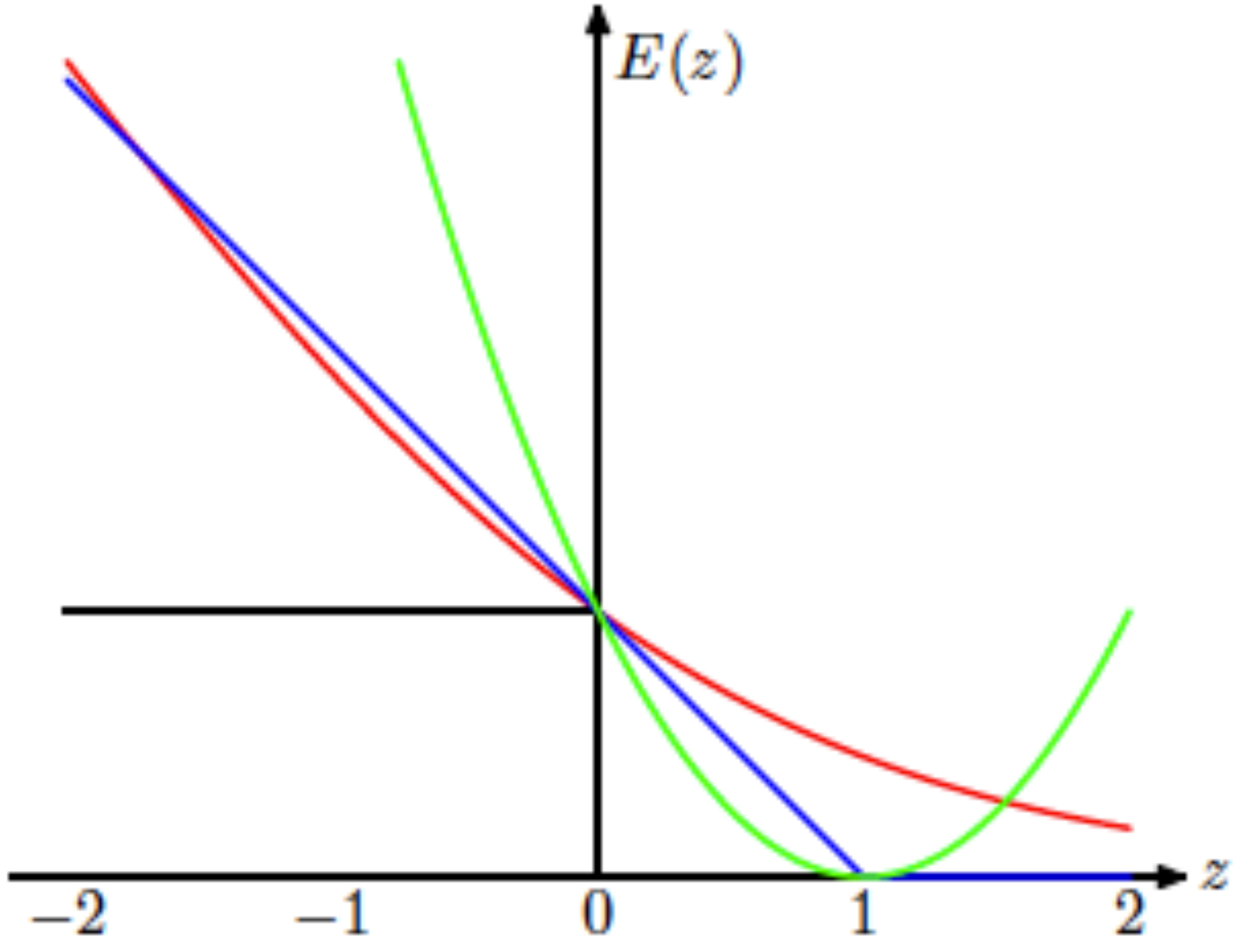
C=0.15



C=0.1

CONNECTION WITH OTHER MODELS

Plot of the 'hinge' error function used in support vector machines, shown in blue, along with the error function for logistic regression, rescaled by a factor of $1/\ln(2)$ so that it passes through the point $(0, 1)$, shown in red. Also shown are the misclassification error in black and the squared error in green.



Properties of SVM

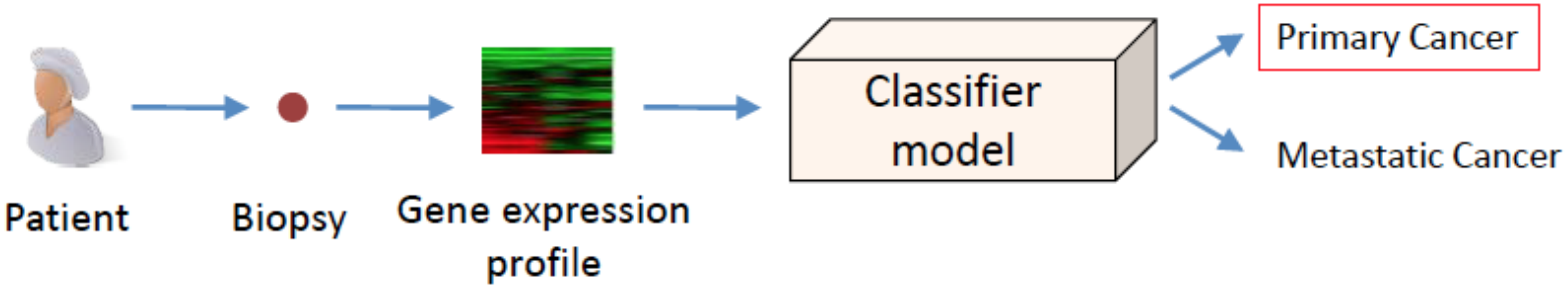
- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection

SVM Applications

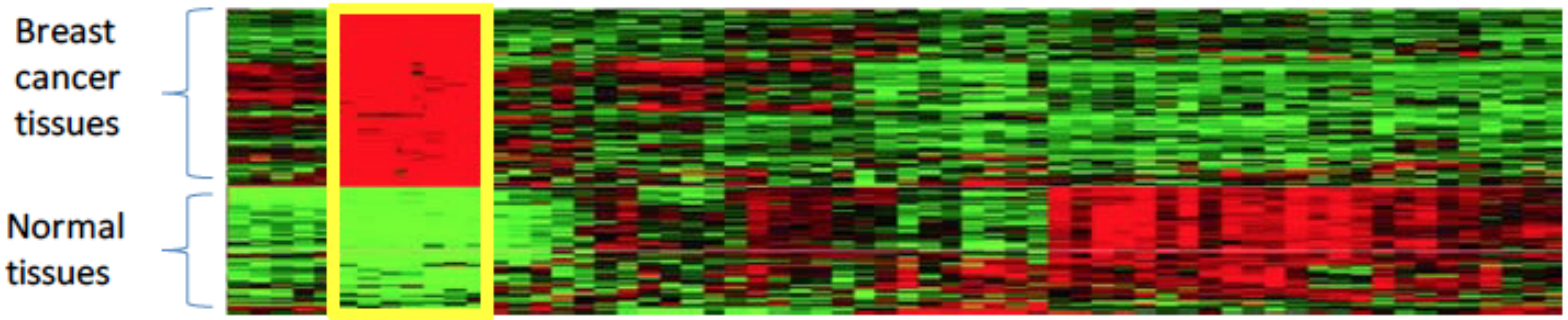
- SVM has been used successfully in many real-world problems
  - text (and hypertext) categorization
  - image classification
  - bioinformatics (Protein classification, Cancer classification)
  - hand-written character recognition

Application 1: Cancer Classification

- SVM has been used successfully in many real-world problems
  - text (and hypertext) categorization
  - image classification
  - bioinformatics (Protein classification, Cancer classification)
  - hand-written character recognition

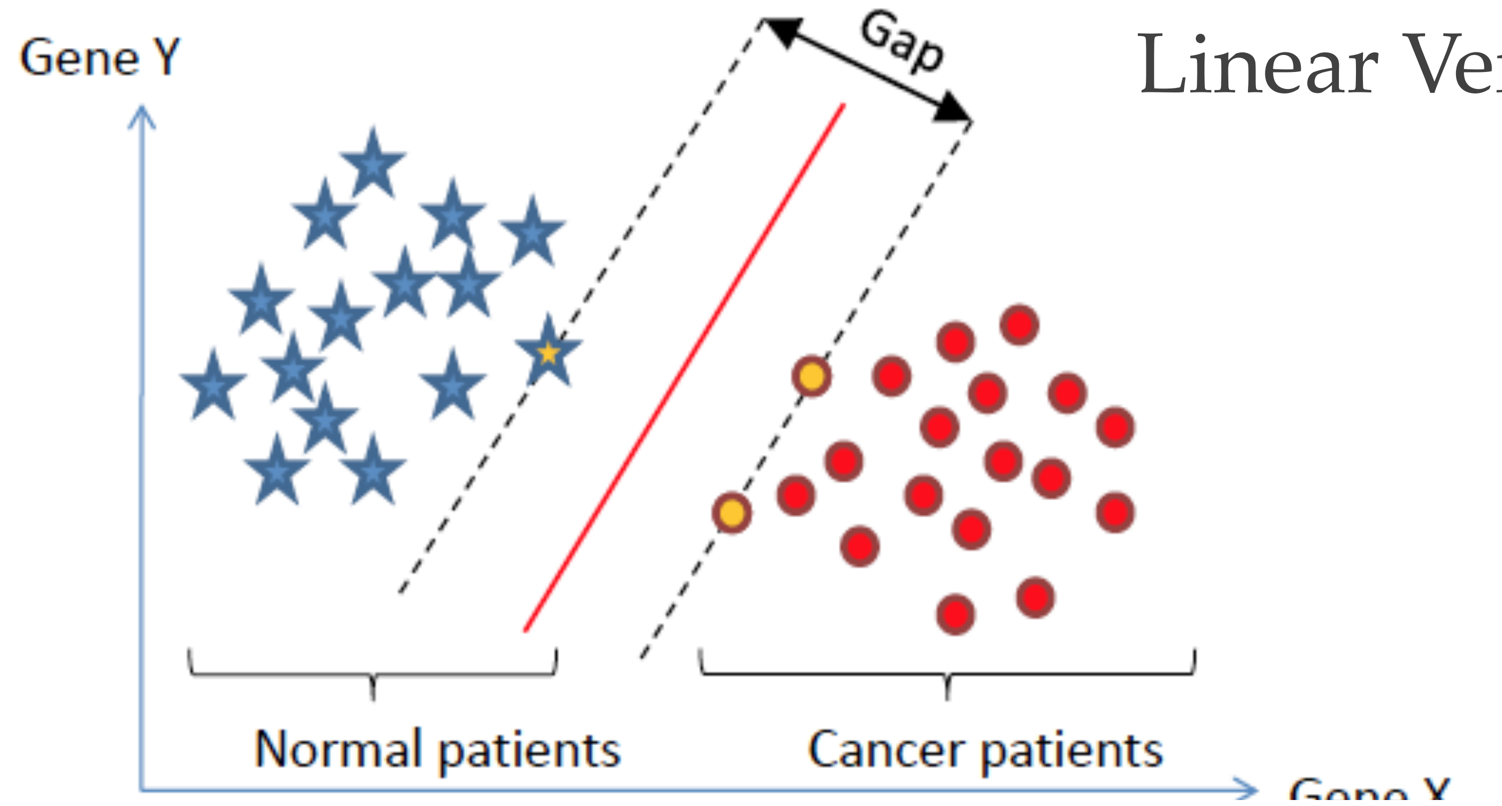


- E.g., find the most compact panel of breast cancer biomarkers from microarray gene expression data for 20,000 genes:

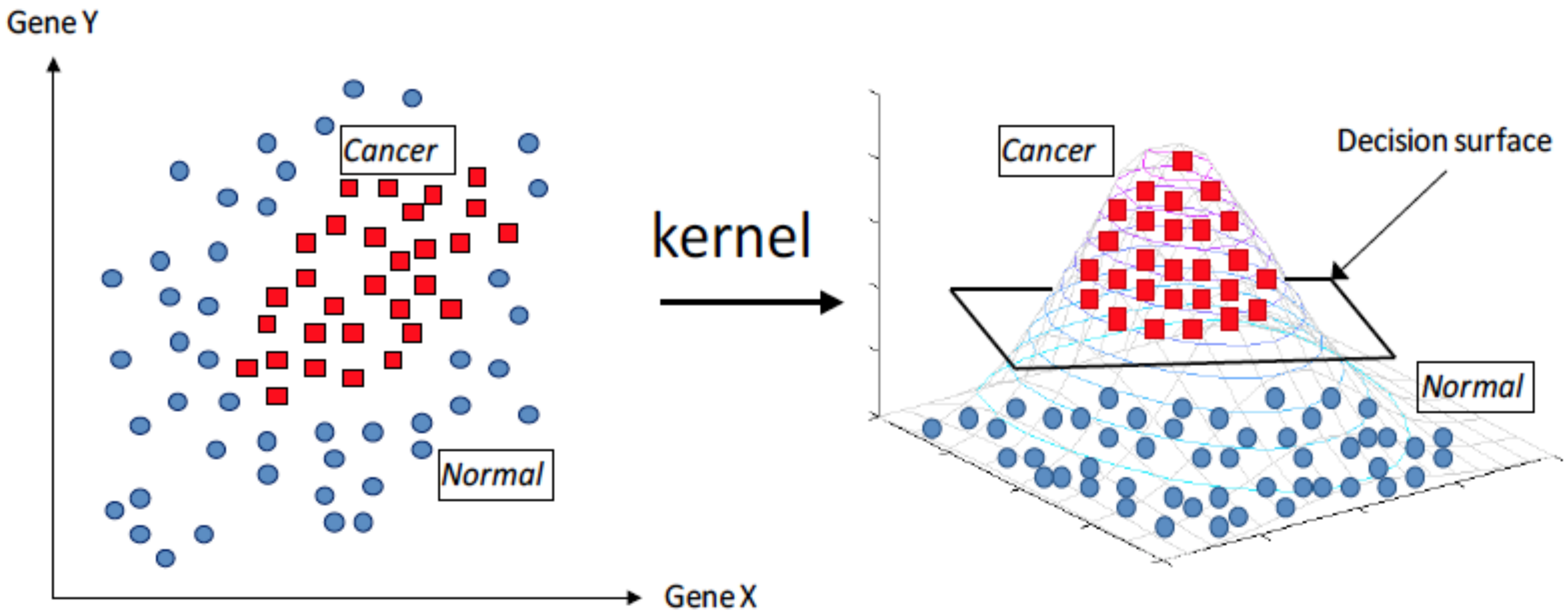


"Gentle Introduction to SVMs in Biomedicine", Statnikvo et al. NYU school of medicine

Application 1: Cancer Classification



Linear Versus Non-linear SVMs



"Gentle Introduction to SVMs in Biomedicine", Statnikvo et al. NYU school of medicine

Weakness of SVM

- **It is sensitive to noise**

  - A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

  - how to do multi-class classification with SVM?

  - Answer:

1) with output m, learn m SVM's

- **SVM 1 learns "Output==1" vs "Output != 1"**
- **SVM 2 learns "Output==2" vs "Output != 2"**
- **:**
- **SVM m learns "Output==m" vs "Output != m"**

2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Application 2: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.

  - email filtering, web searching, sorting documents by topic, etc..

- A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category.

Application 2: Text Categorization

IR's vector space model (aka bag-of-words representation)

- A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms
- Values of an entry can be binary or weights

$$\phi_i(x) = \frac{\mathrm{tf}_i \log(\mathrm{idf}_i)}{\kappa},$$

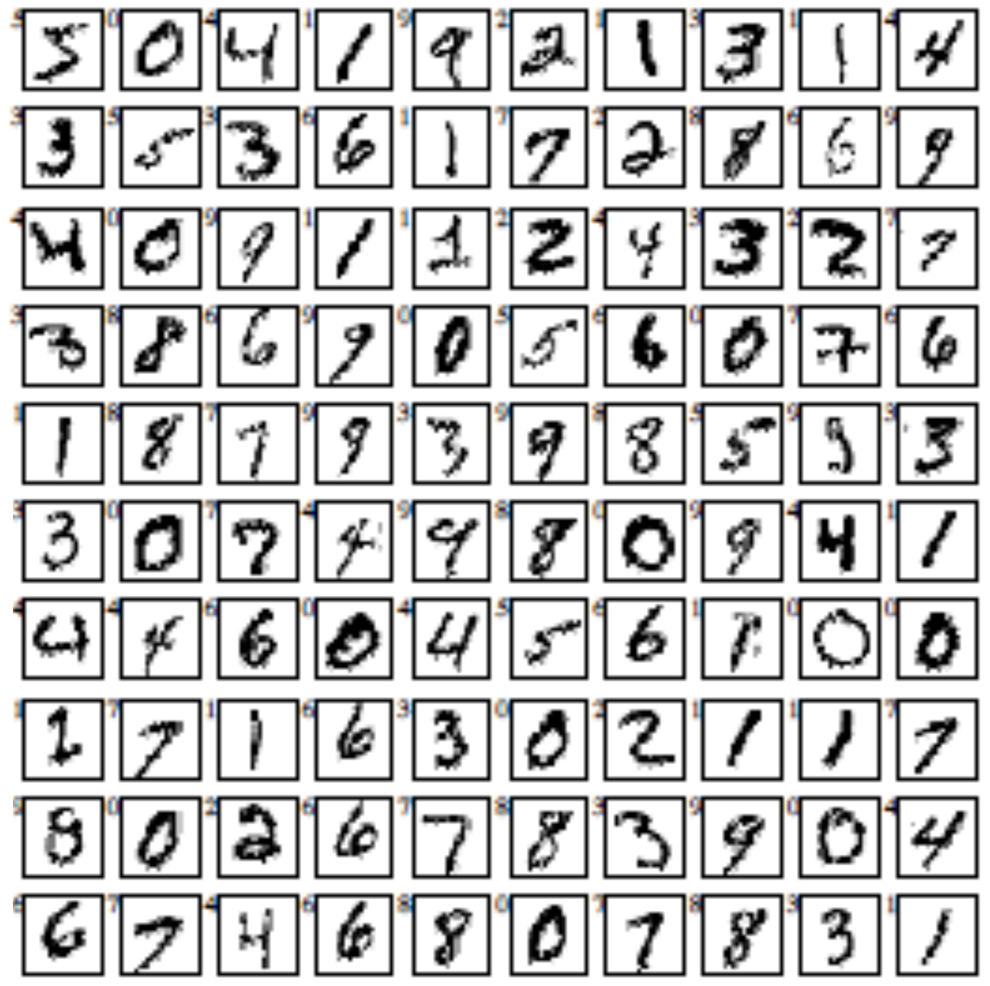
- Doc $\mathbf{x}$ => $\phi(x)$

Application 3: Handwriting Recognition

- The distance between two documents is $<\phi(x)\ \phi(z)>$
- $K(x,z) = <\phi(x)\ \phi(z)>$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination.
- Why SVM?
  - High dimensional input space
  - Few irrelevant features (dense concept)
  - Sparse document vectors (sparse instances)
  - Text categorization problems are linearly separable

For example MNIST hand-writing recognition.
60,000 training examples, 10000 test examples, 28x28.

Linear SVM has around 8.5% test error.
Polynomial SVM has around 1% test error.



23 SVMs : full MNIST results

Classifier	Test Error
linear	8.4%
3-nearest-neighbor	2.4%
RBF-SVM	1.4 %

Some Considerations

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures

- Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

- Optimization criterion – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested

Softwares

30 SVMs : software

Lots of SVM software:

- LibSVM (C++)
- SVMLight (C)

As well as complete machine learning toolboxes that include SVMs:

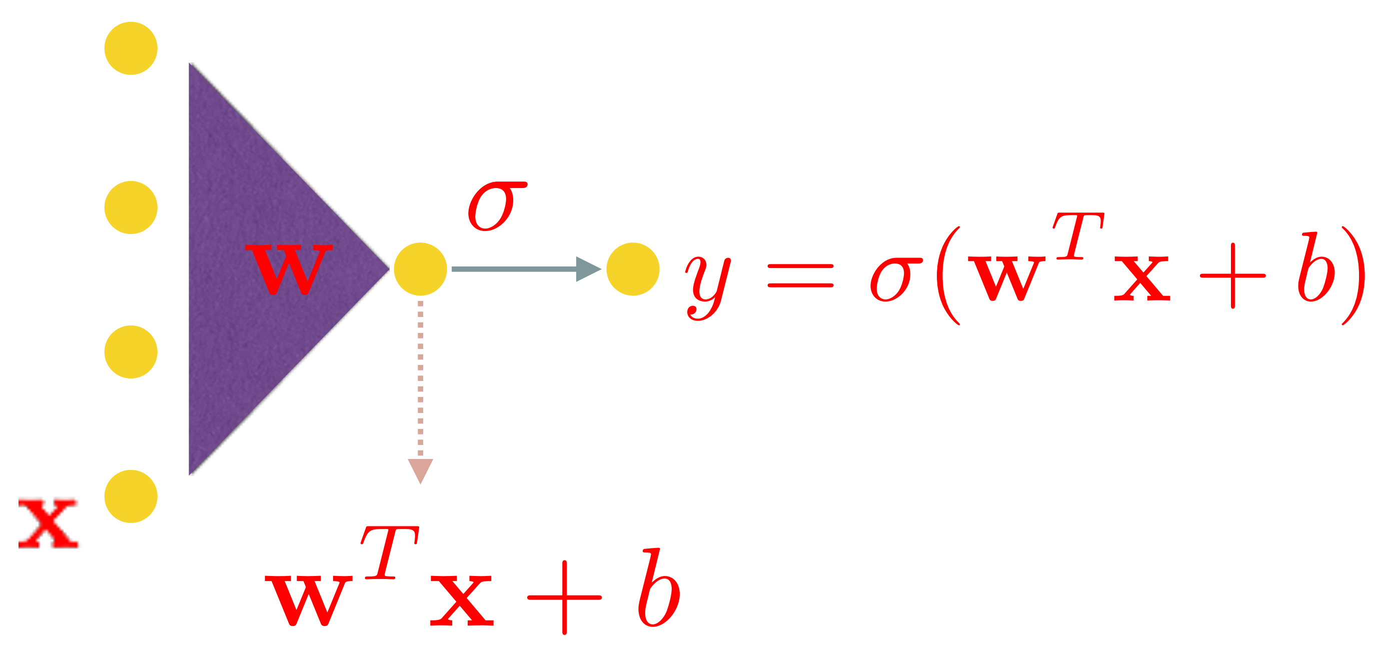
- Torch (C++)
- Spider (Matlab)
- Weka (Java)

All available through `www.kernel-machines.org`.

NEURAL NETWORKS AND DEEP LEARNING

VISUALIZING LOGISTIC REGRESSION AS A NEURAL NETWORK

- A logistic regression is the simplest neural network
  - Number of parameters in the model - D+1

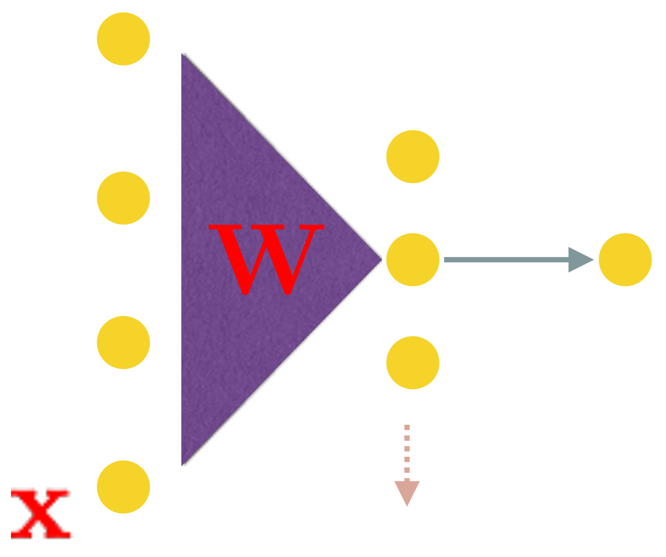


MULTI-CLASS LOGISTIC REGRESSION

- Targets are one-hot encoded vectors
  - Model approximates class posteriors using
    - softmax function

$$\begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

$$softmax(\mathbf{a}) = \begin{bmatrix} \frac{e^{a_1}}{\sum_j e^{a_j}} \\ \frac{e^{a_2}}{\sum_j e^{a_j}} \\ . \\ . \\ \frac{e^{a_K}}{\sum_j e^{a_j}} \end{bmatrix}$$



$$\mathbf{y} = softmax(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

$$\mathbf{Wx} + \mathbf{b} \qquad \mathbf{W} \in \mathcal{R}^{K \times D} \quad \mathbf{b} \in \mathcal{R}^{K}$$

SOFTMAX FUNCTION

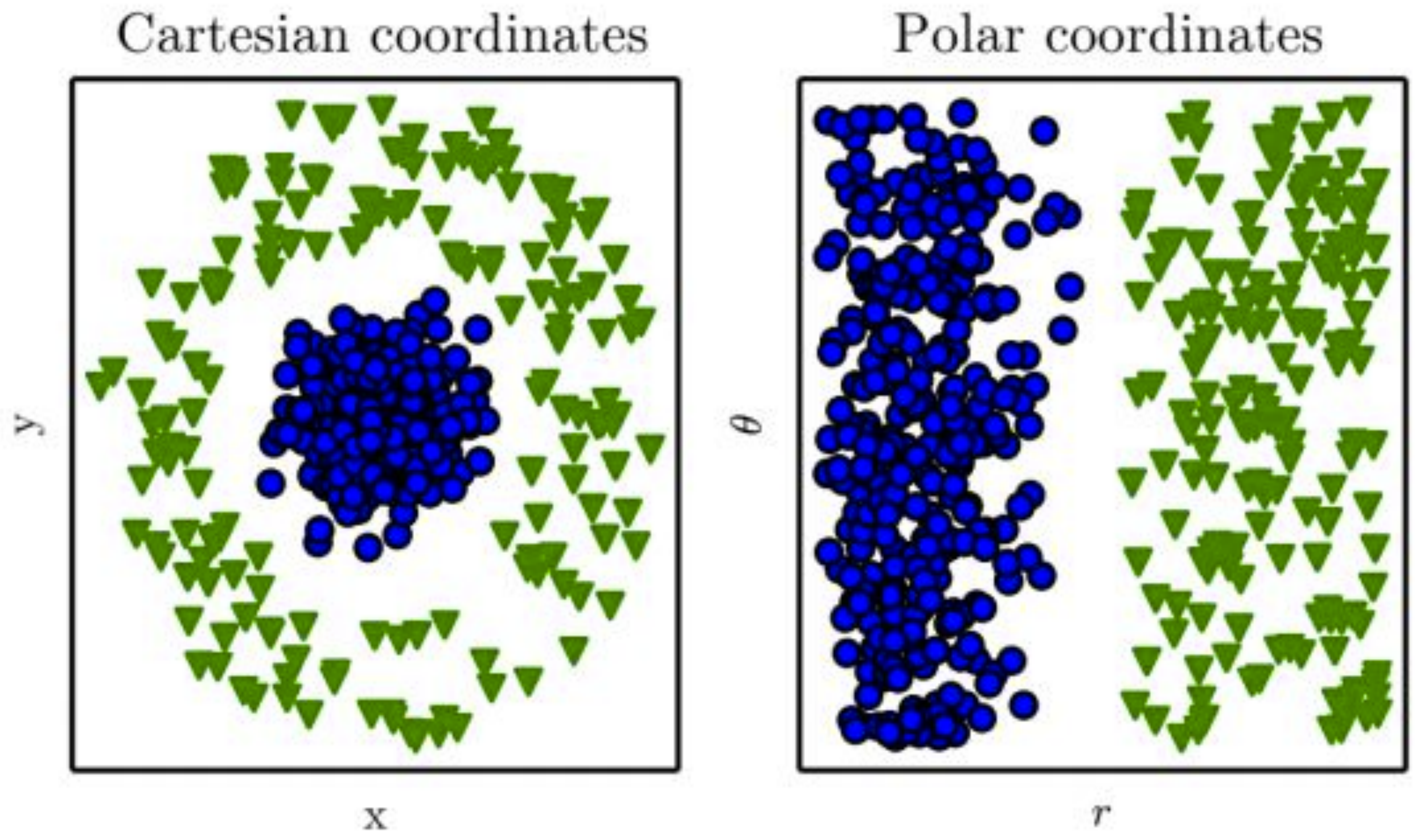
- Each value is positive
- Sum of the vector is 1.0
  - Can be interpreted as class posterior probabilities

$$softmax(\mathbf{a}) = \begin{bmatrix} \frac{e^{a_1}}{\sum_j e^{a_j}} \\ \frac{e^{a_2}}{\sum_j e^{a_j}} \\ . \\ . \\ \frac{e^{a_K}}{\sum_j e^{a_j}} \end{bmatrix}$$

$$\begin{bmatrix} p(C_1|\mathbf{x}) \\ p(C_2|\mathbf{x}) \\ . \\ . \\ p(C_K|\mathbf{x}) \end{bmatrix}$$

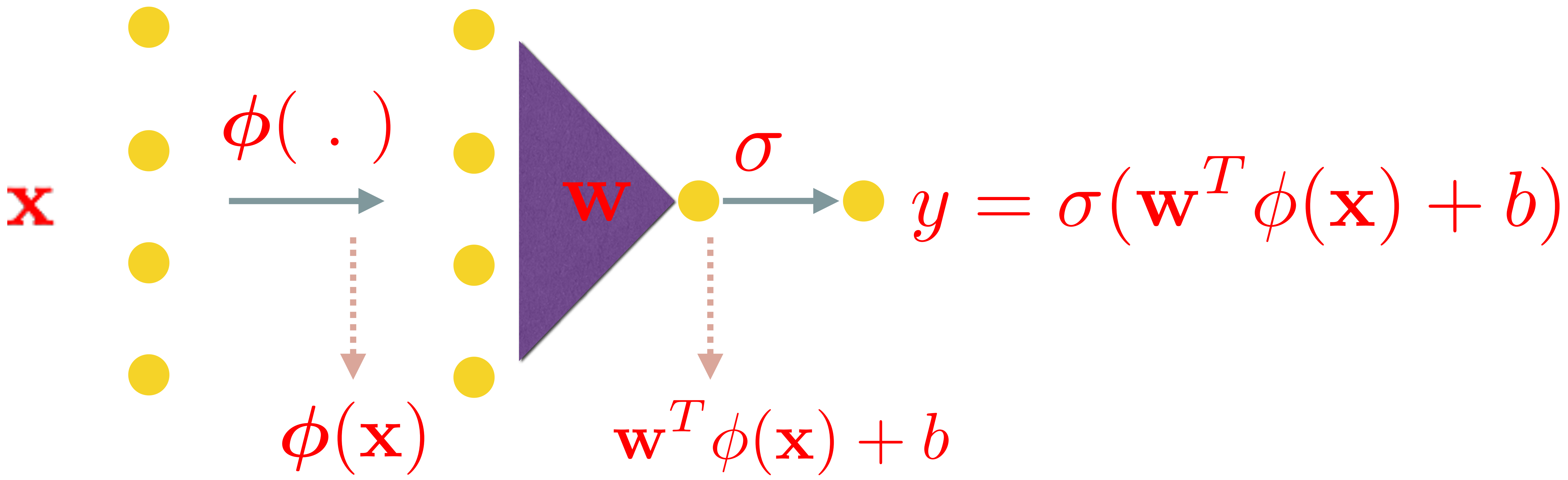
QUESTION

- **Can we transform the data to linearly separable space**
  - then apply the logistic regression to find the classifier.
    - Example



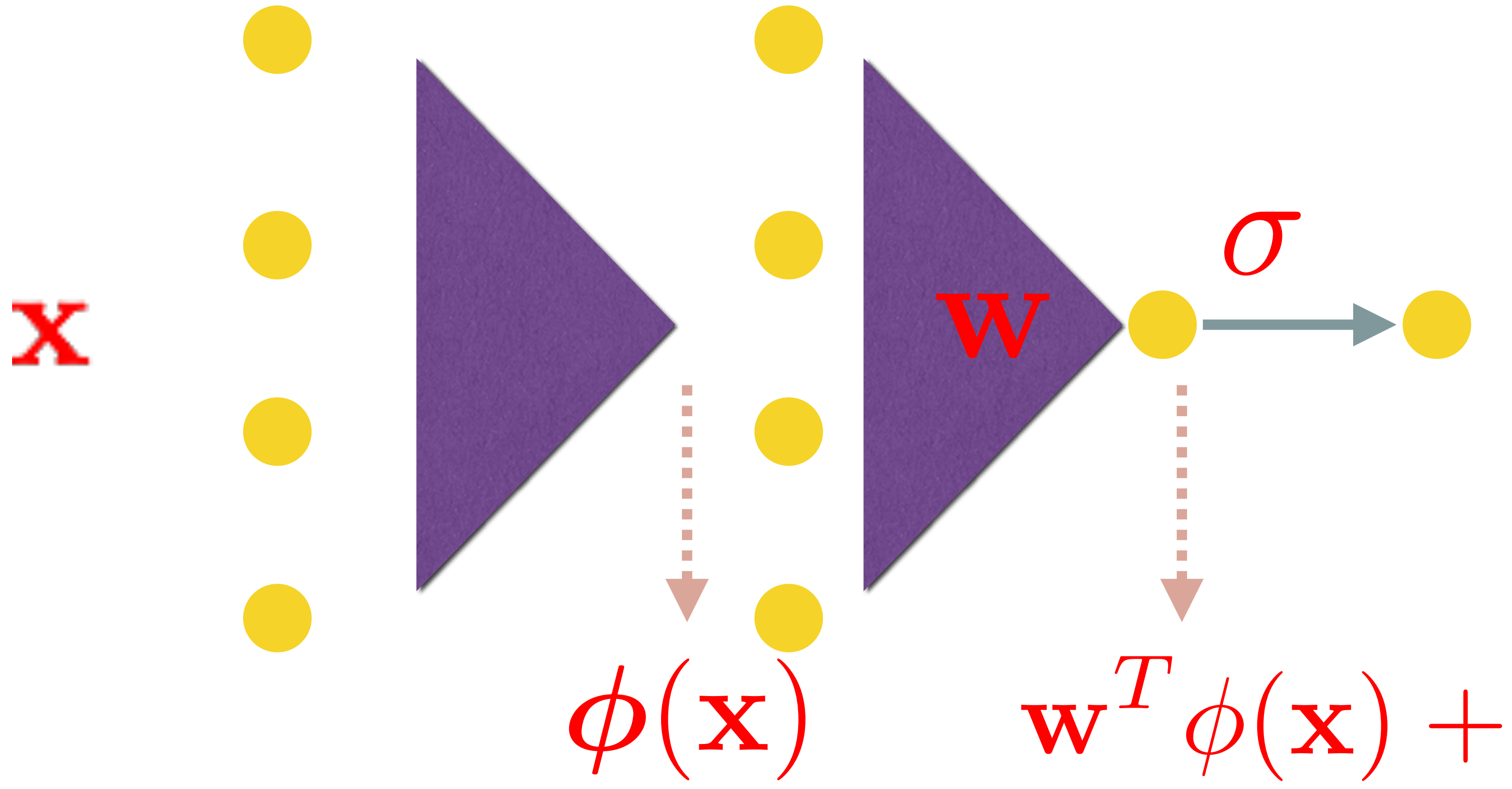
QUESTION

- **Can we transform the data to linearly separable space**
  - then apply the logistic regression to find the classifier.
- Can we learn such a transform from the data itself
  - non-linear transformation of the data is needed



QUESTION

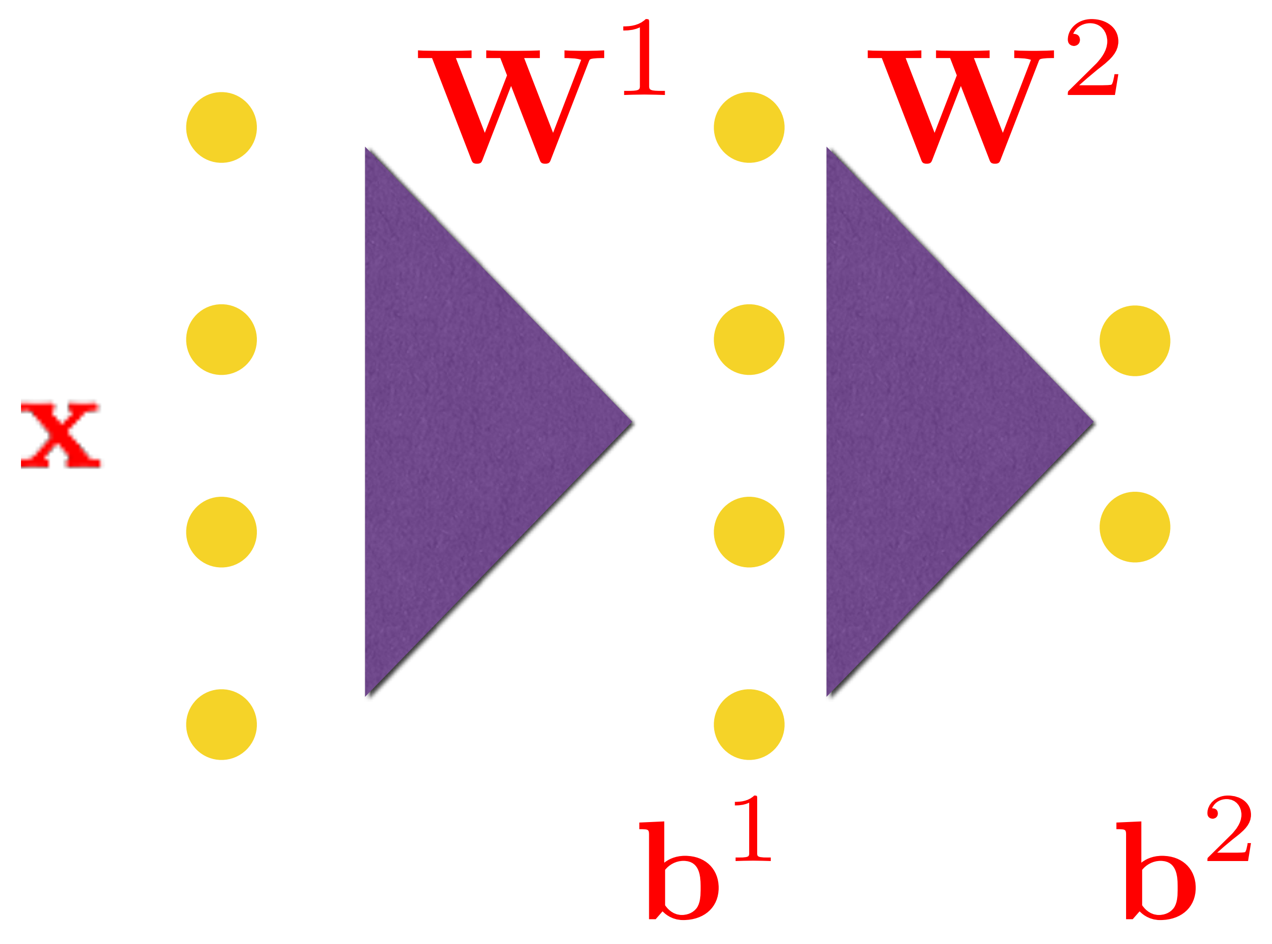
- **Can we transform the data to linearly separable space**
  - then apply the logistic regression to find the classifier.
- **Can we learn such a transform from the data itself**
  - non-linear transformation of the data is needed.
  - can this also be realized as neural layer



$y = \sigma(\mathbf{w}^T \phi(\mathbf{x}) + b)$

Neural Network - 1- Hidden Layer

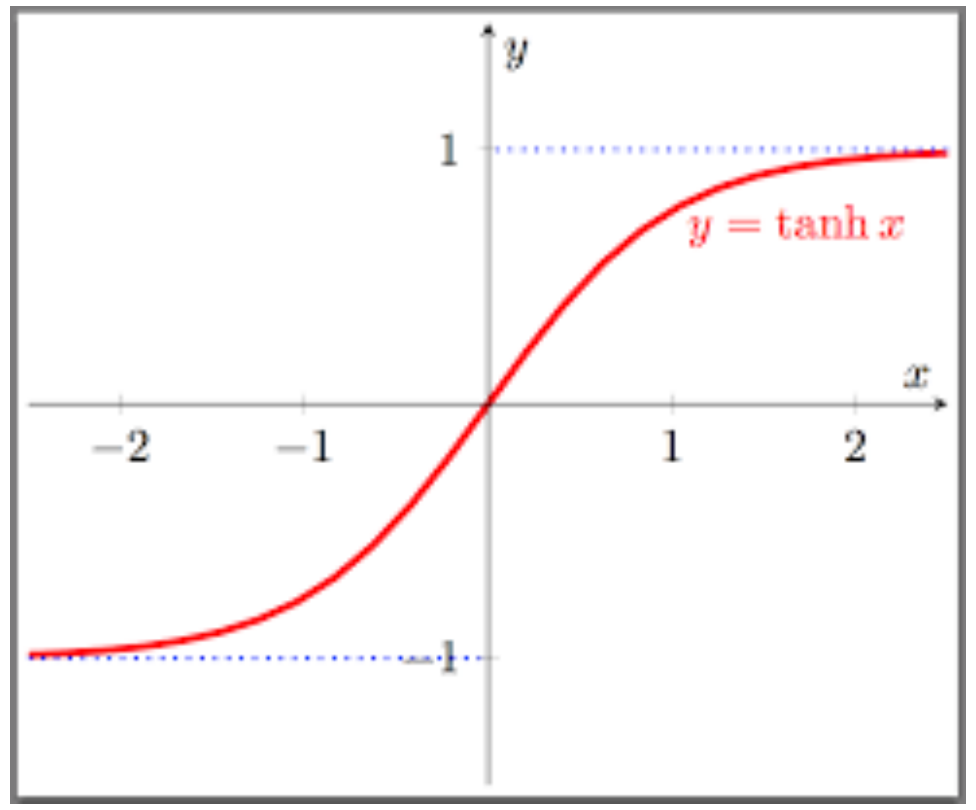
- Has more capacity than logistic regression
  - can learn non-linear data separation functions
  - both 2-class and K-class classification possible
  - can be learnt using gradient descent



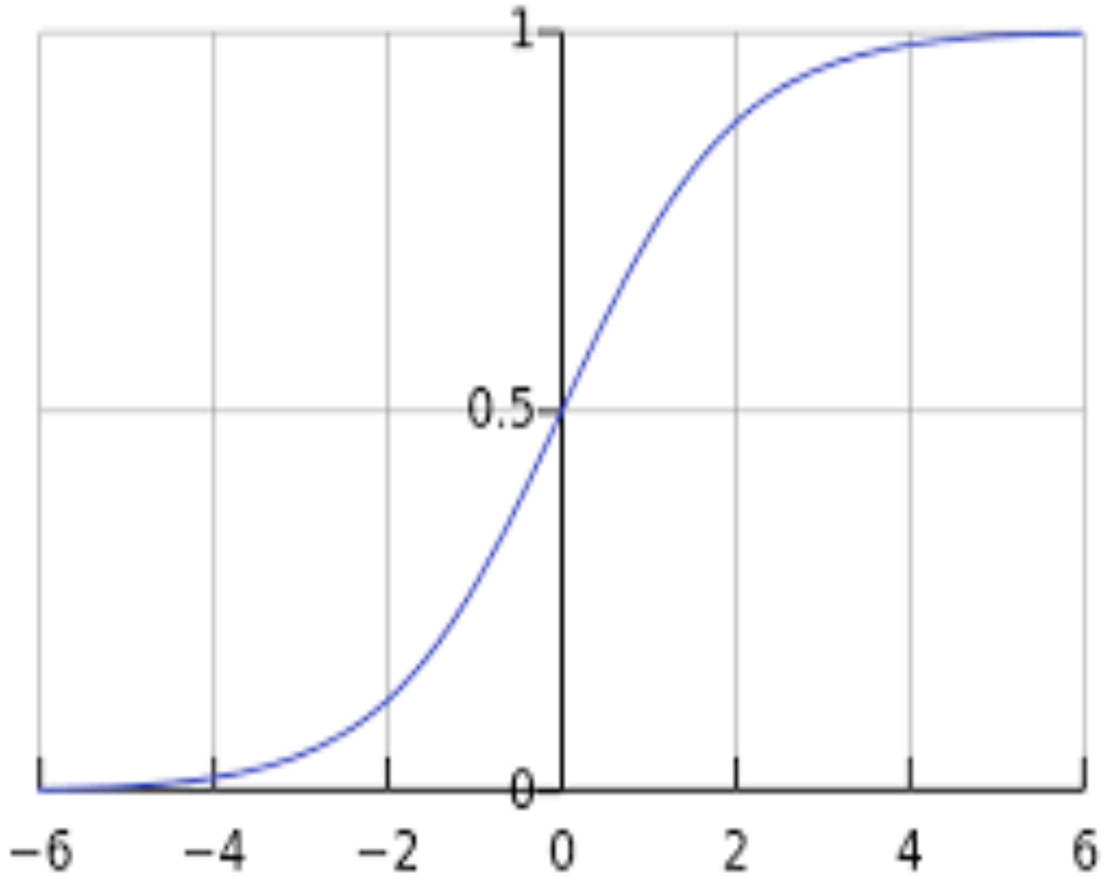
TYPES OF NON-LINEARITIES

Non-linearity in hidden layer

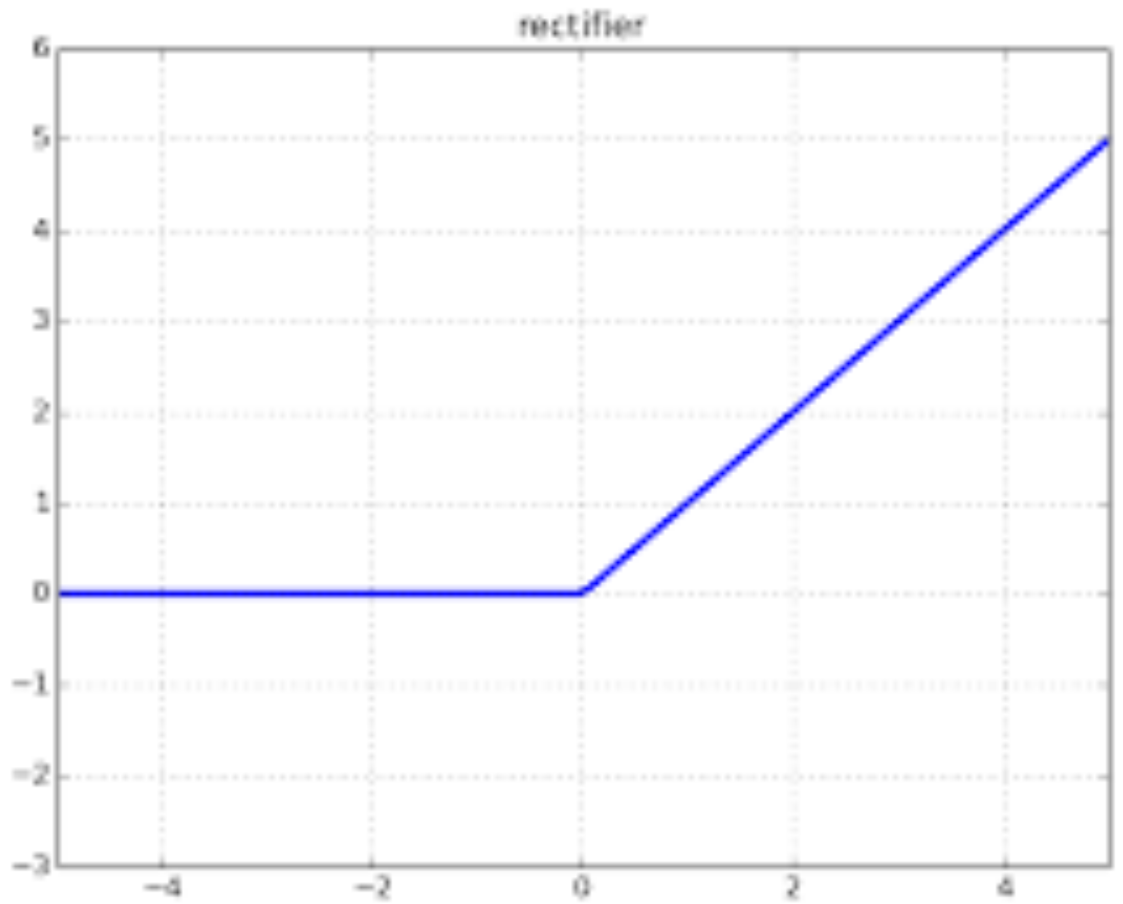
tanh



sigmoid



ReLu



OUTPUT LAYER NON-LINEARITY AND COST FUNCTIONS

- Using a softmax non-linearity
  - error function is cross entropy

$$E_{CE} = -\sum_{n}\sum_{k} t_{nk}\ log(v_{nk})$$

- For regression style tasks - output is linear
  - error function is mean square error

$$E_{MSE} = -\sum_{n}\sum_{k}(t_{nk} - v_{nk})^2$$

FORWARD THROUGH THE MODEL PROPAGATION LEARNING

- Computations in the forward direction

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{x} + \mathbf{b}^1$$
$$\mathbf{z}^1 = \sigma(\mathbf{a}^1)$$
$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{z}^1 + \mathbf{b}^2$$
$$\mathbf{y} = softmax(\mathbf{a}^2)$$

- Loss function

$$E_{CE} = -\sum_n \sum_k t_{nk} \; log(v_{nk})$$

$$\boldsymbol{\Theta} = \{\mathbf{W}^1 \;,\; \mathbf{b}^1 \;,\; \mathbf{W}^2 \;,\; \mathbf{b}^2\}$$

- Parameters in the model

Need to be updated based on the gradients w.r.t. the error

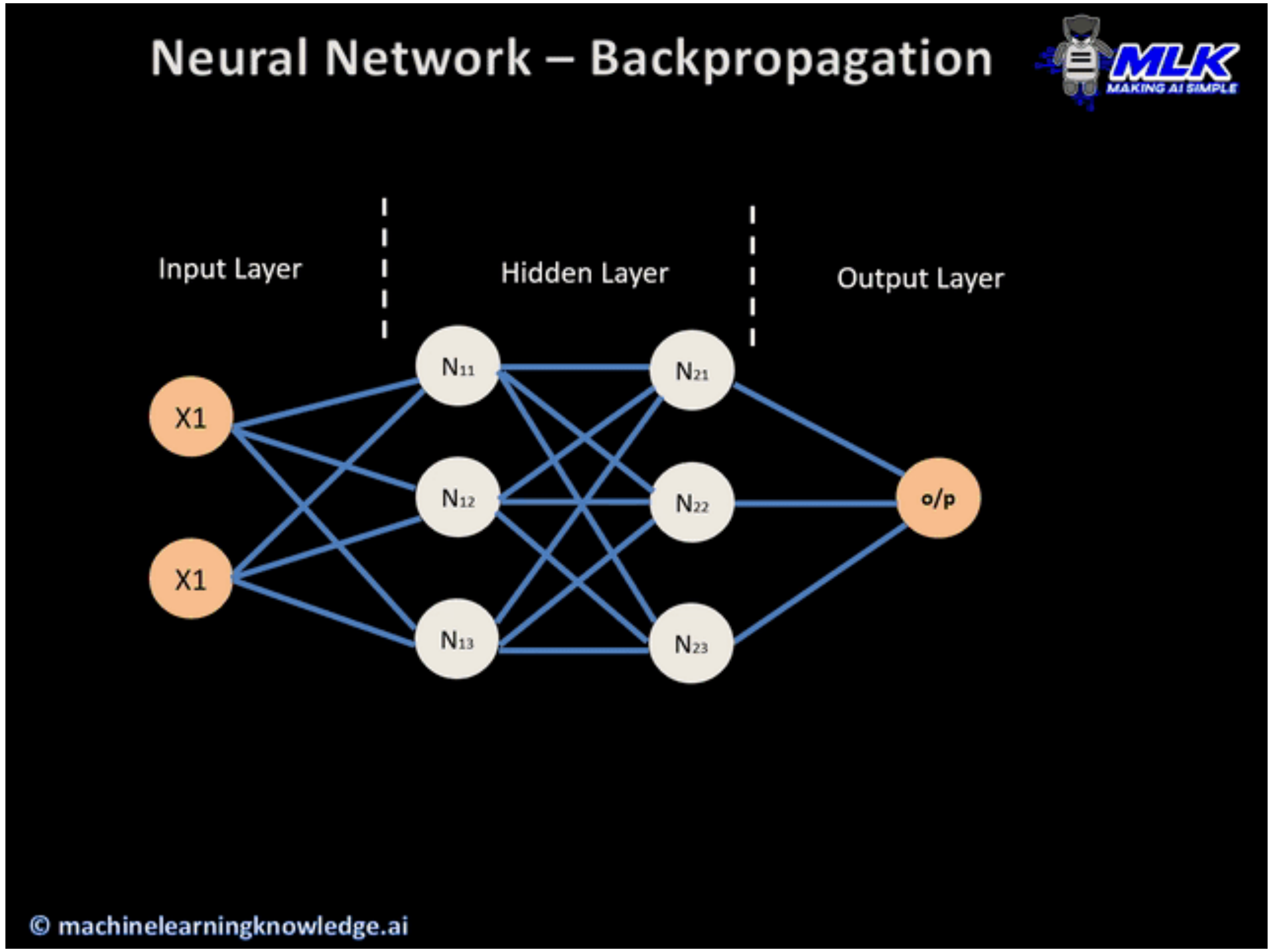
GRADIENT COMPUTATION IN THE MODEL

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{x} + \mathbf{b}^1$$
$$\mathbf{z}^1 = \sigma(\mathbf{a}^1)$$
$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{z}^1 + \mathbf{b}^2$$
$$\mathbf{y} = softmax(\mathbf{a}^2)$$

$$E_{CE} = -\sum_n \sum_k t_{nk}\ log(v_{nk})$$

- When computing the gradients
  - Order of computations
    - The derivative of the loss function w.r.t output layer
    - The derivative of the loss function w.r.t output activation
    - The derivative of the loss function w.r.t hidden layer outputs
    - The derivative of the loss function w.r.t. hidden layer activations

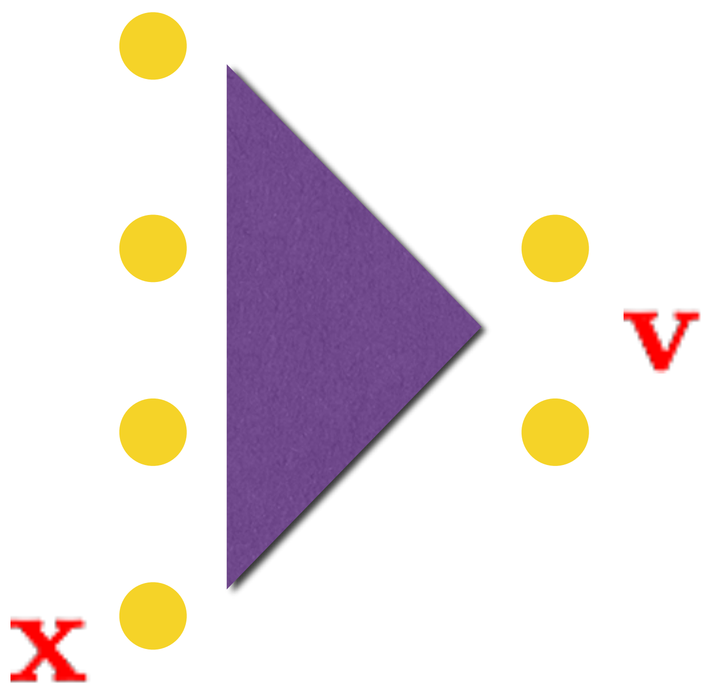
BACK PROPAGATION LEARNING



LEAP

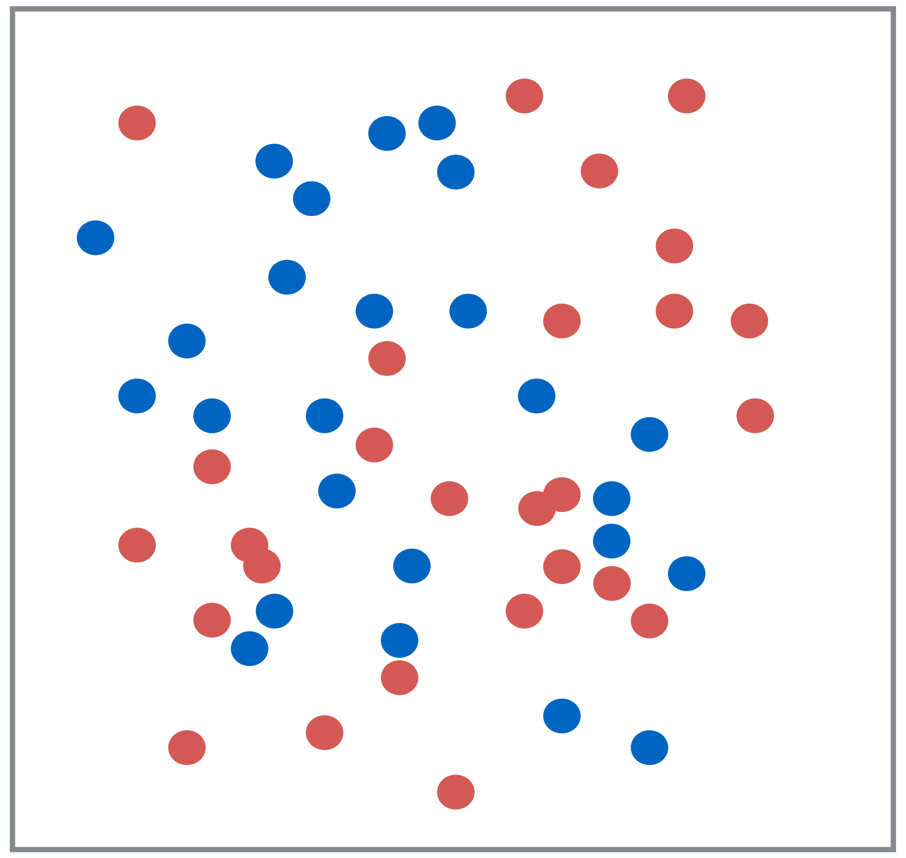
PERCEPTRON ALGORITHM

Perceptron Model [McCulloch, 1943, Rosenblatt, 1957]



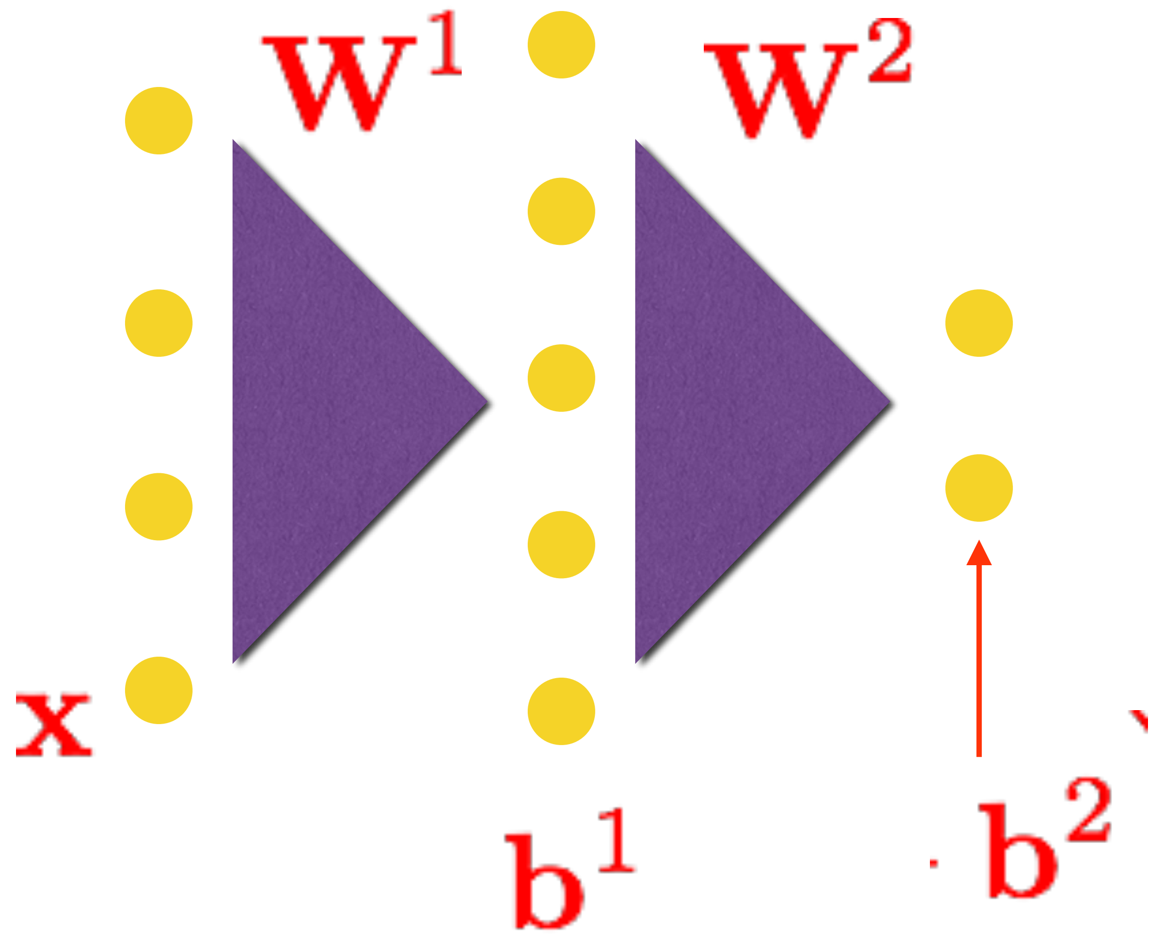
Targets are binary classes [-1,1]

What if the data is not linearly separable



Multi-layer Perceptron

Multi-layer Perceptron [Hopfield, 1982]



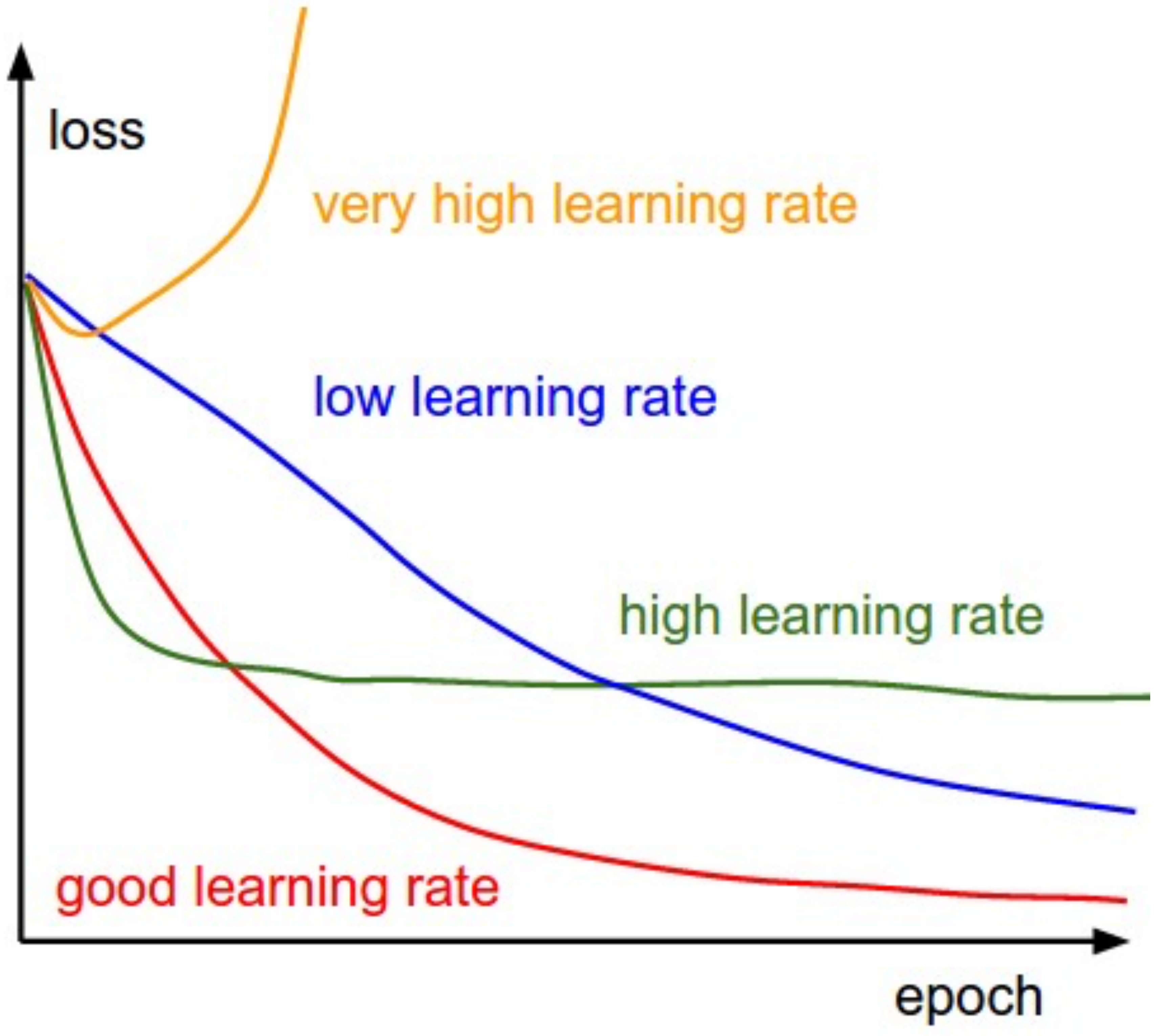
$$\mathbf{v}^2 = \psi\left(\mathbf{W}^2\phi(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2\right)$$

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

ψ thresholding function

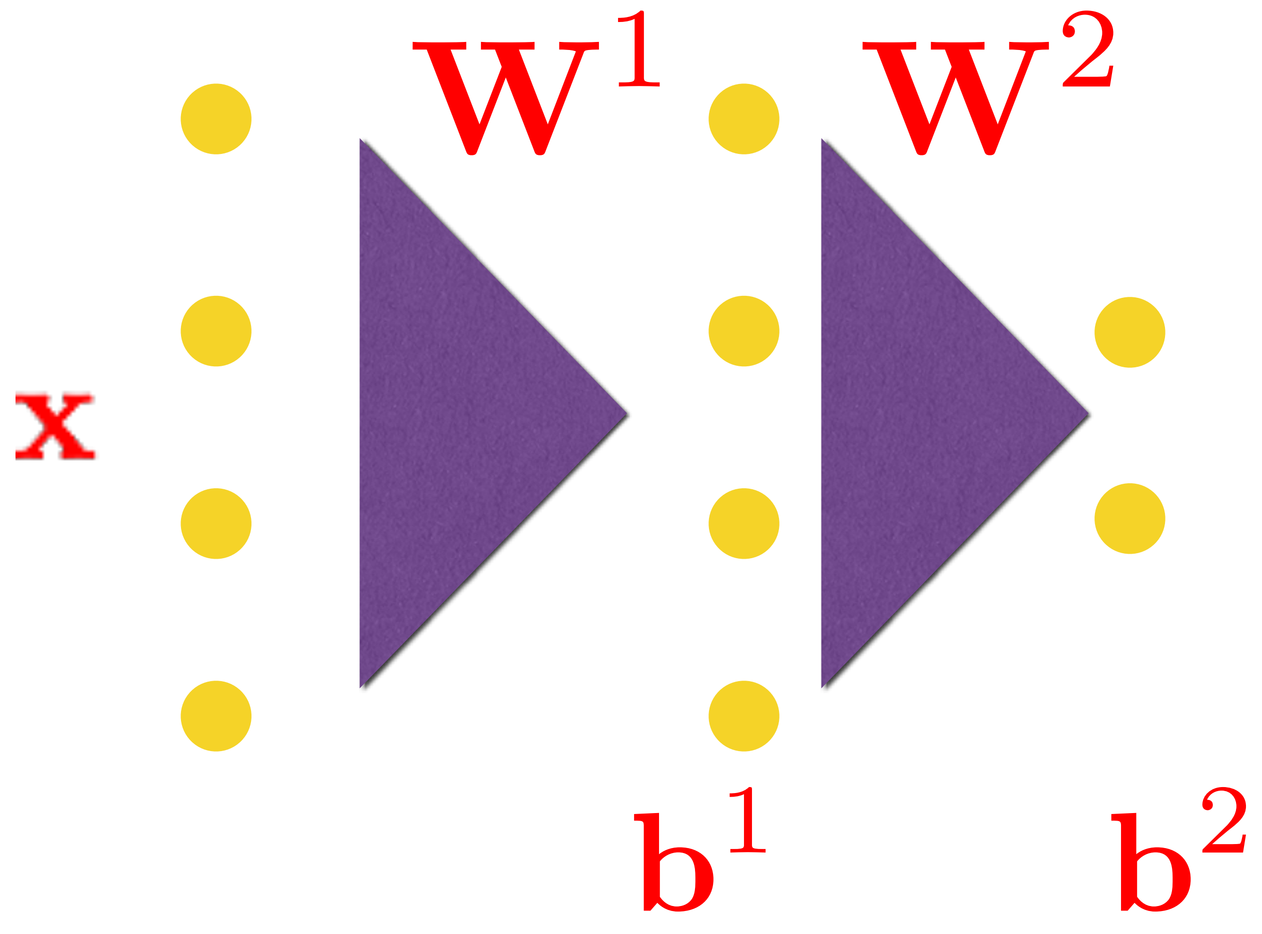
MULTI-LAYER PERCEPTRON

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

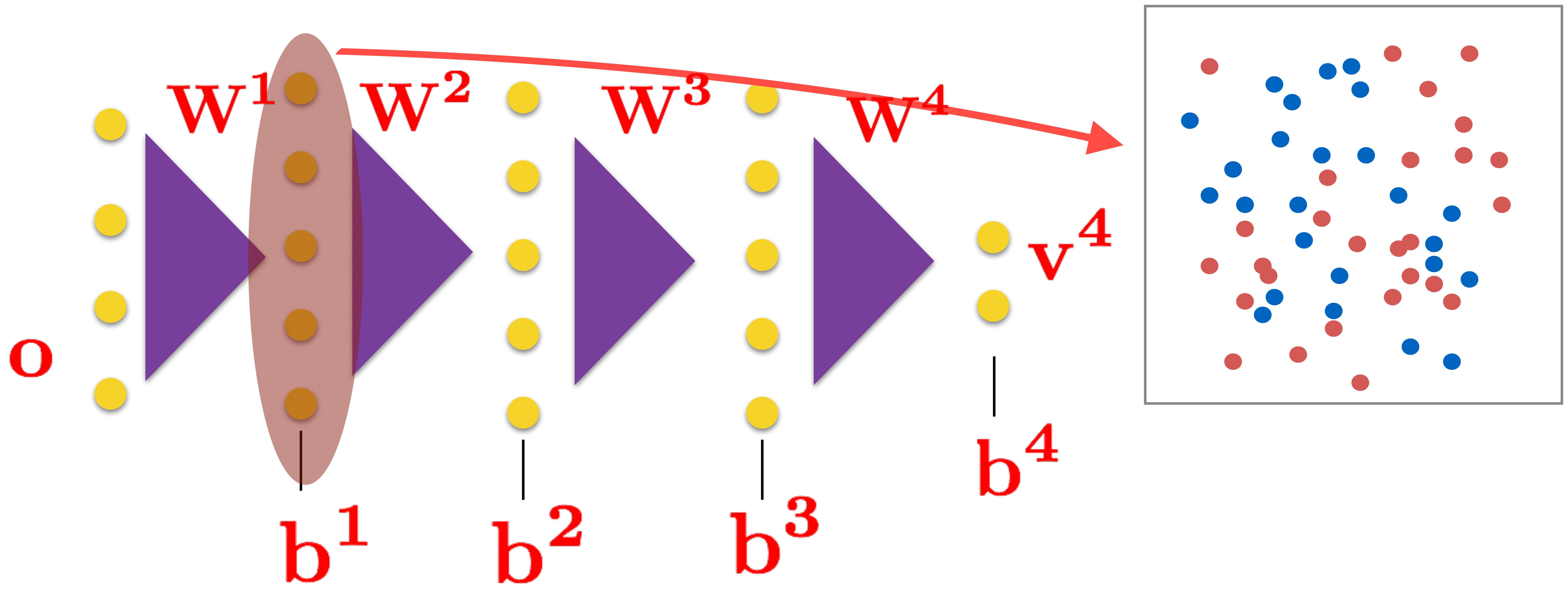


NEURAL NETWORKS - 1 HIDDEN LAYER

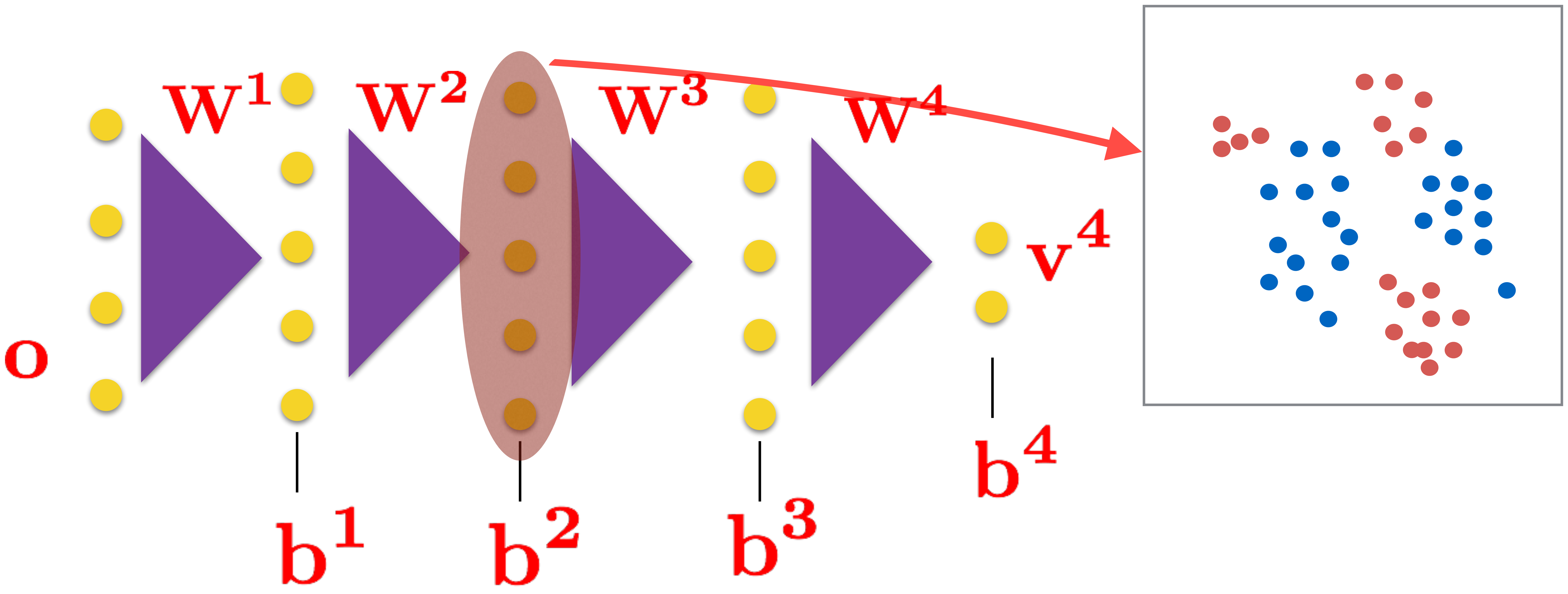
- For complex problems in audio/image/text
  - Single hidden layer may be too restrictive in learning the model parameters
  - May not scale up with availability of big data.



Neural Networks — Deep

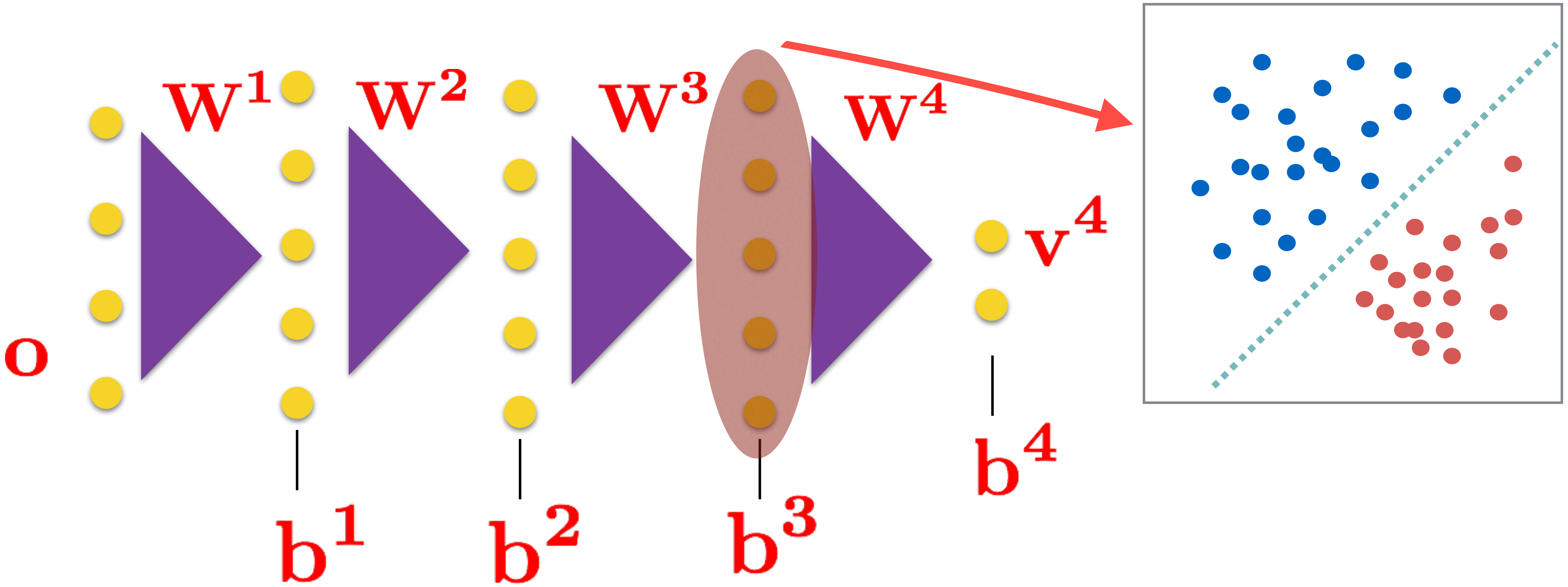


Neural networks with multiple hidden layers - Deep networks



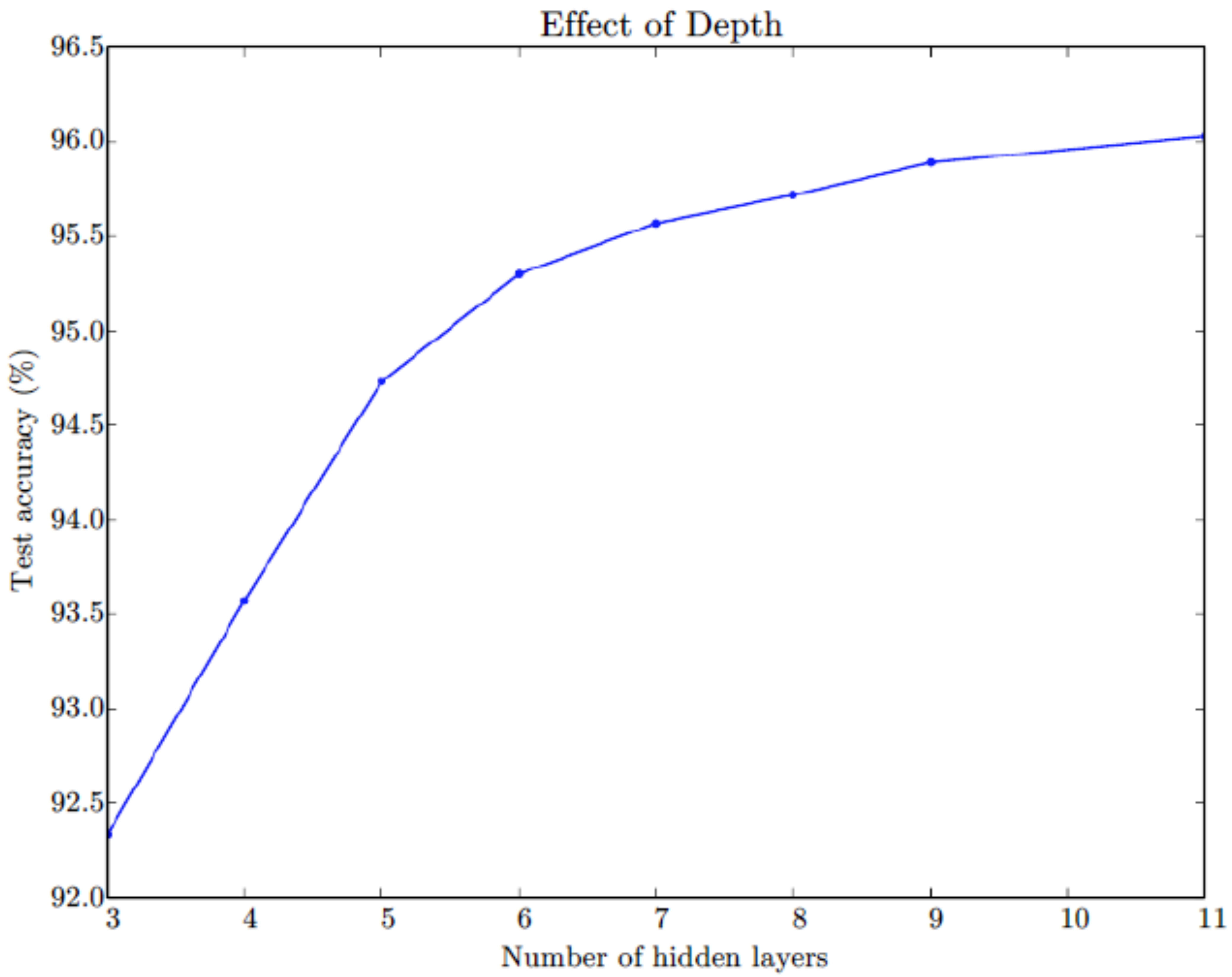
DEEP NEURAL NETWORKS

Neural networks with multiple hidden layers - Deep networks



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

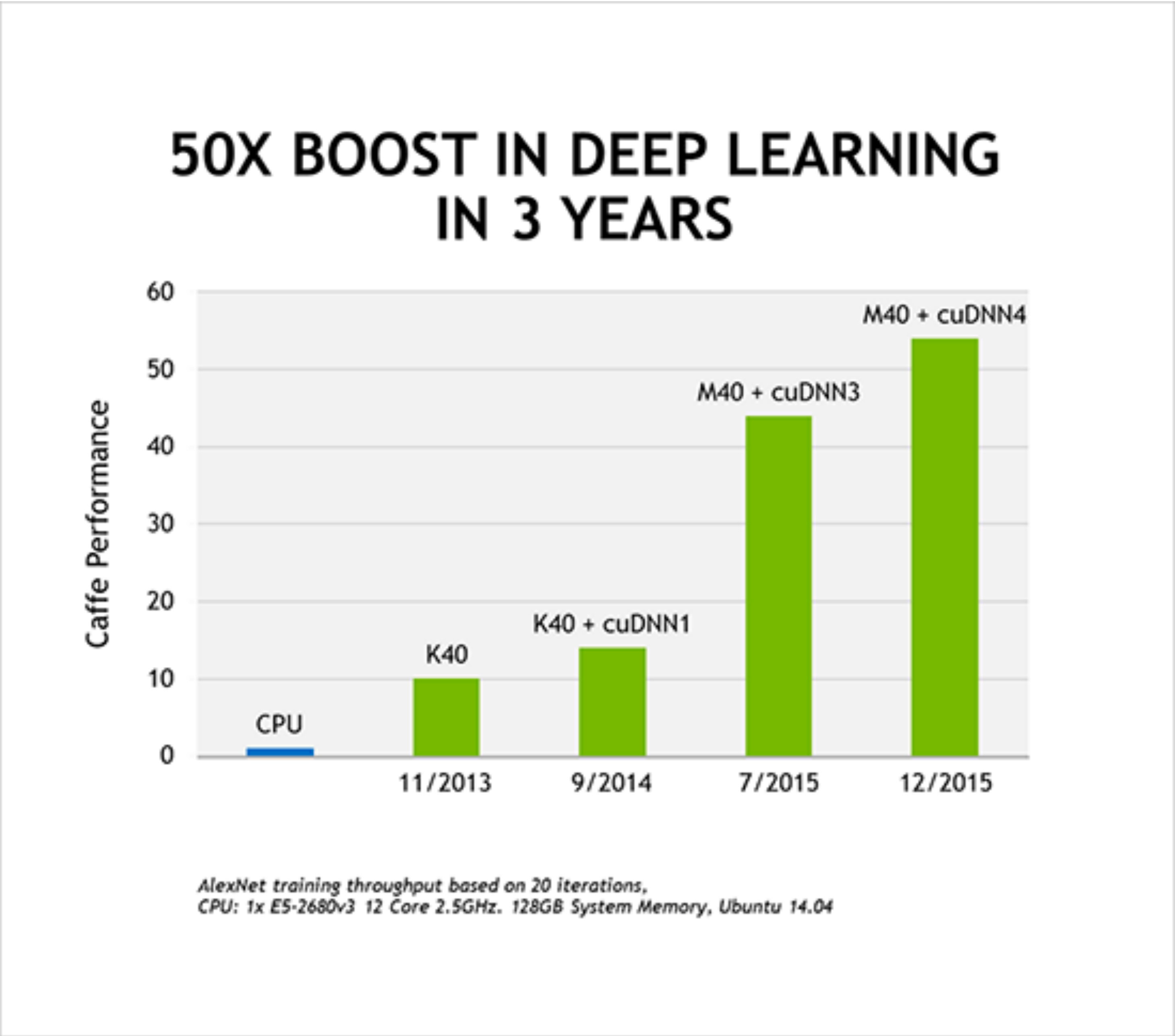
Need for Depth



$$\boldsymbol{h}^{(1)} = g^{(1)}\left(\boldsymbol{W}^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right)$$

$$\boldsymbol{h}^{(2)} = g^{(2)}\left(\boldsymbol{W}^{(2)\top}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right)$$

DEEP NEURAL 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

DEEP NEURAL NETWORKS

- Will the networks generalize with deep networks
  - DNNs are quite data hungry and performance improves by increasing the data.
  - Generalization problem is tackled by providing training data from all possible conditions.
    - Many artificial data augmentation methods have been successfully deployed
  - Providing the state-of-art performance in several real world applications.

Representation Learning in Deep Networks

- The input data representation is one of most important components of any machine learning system.
  - Extract factors that enable classification while suppressing factors which are susceptible to noise.

- Finding the right representation for real world applications - substantially challenging.
  - Deep learning solution - build complex representations from simpler representations.
  - The dependencies between these hierarchical representations are refined by the target.

THANK YOU

Sriram Ganapathy and TA team
LEAP lab, C328, EE, IISc
sriramg@iisc.ac.in