Supervised Learning Methods: SVM, Neural Networks

ECE/CS 498 DS U/G

Lecture 20

Ravi K. Iyer

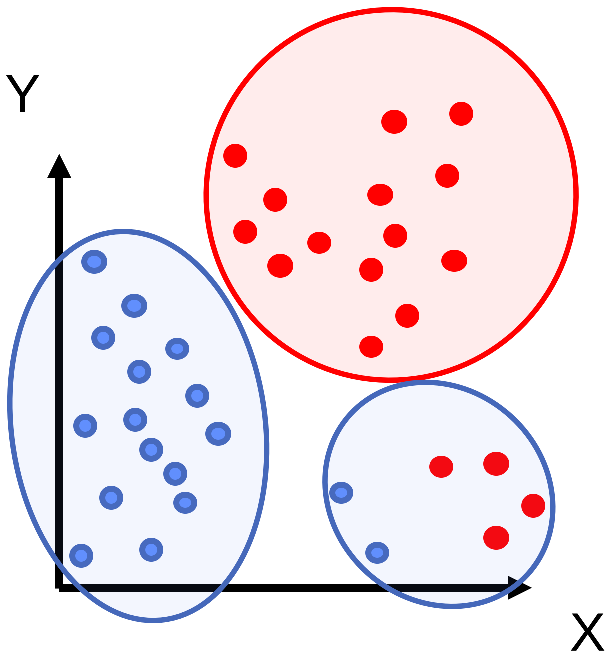
Dept. of Electrical and Computer Engineering

University of Illinois at Urbana Champaign

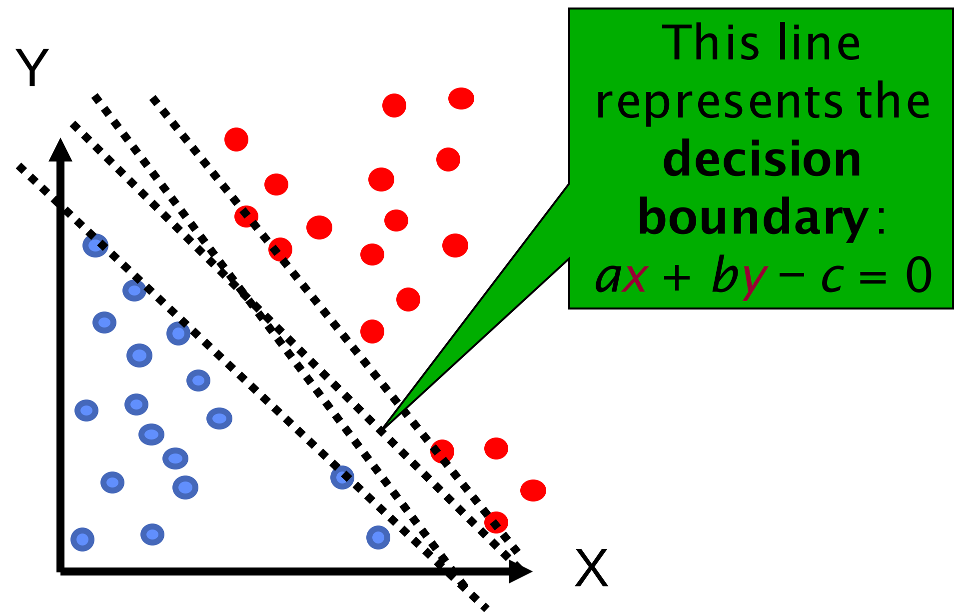
Announcements

- MP3 Checkpoint 2 is due on Wednesday, Apr 17
- HW 4 on Factor graphs and HMM will be released on Wednesday, Apr 17

Unsupervised to Supervised Learning



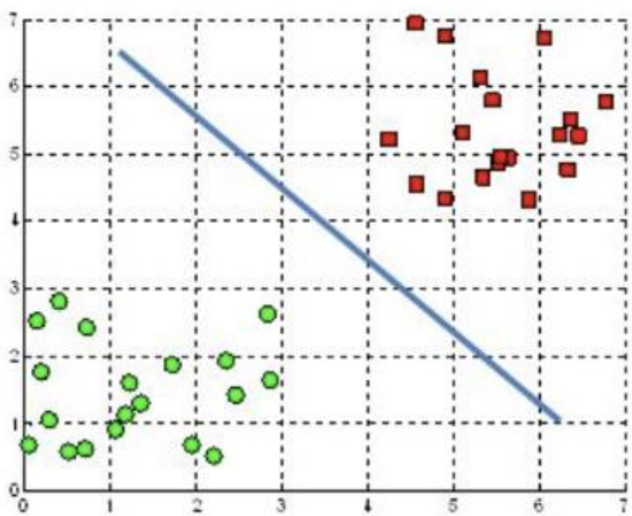
Can we do better? Yes



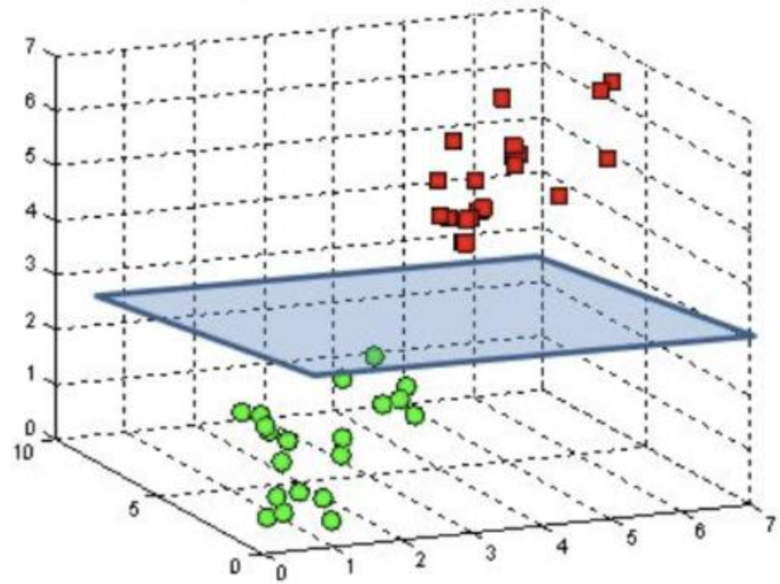
If in a classification task the data is linearly separable, a decision boundary with the same dimensionality as the data can be used to separate the data

Hyperplane as decision boundary

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



Example of decision boundary in 2D space (line) and 3D space (plane).

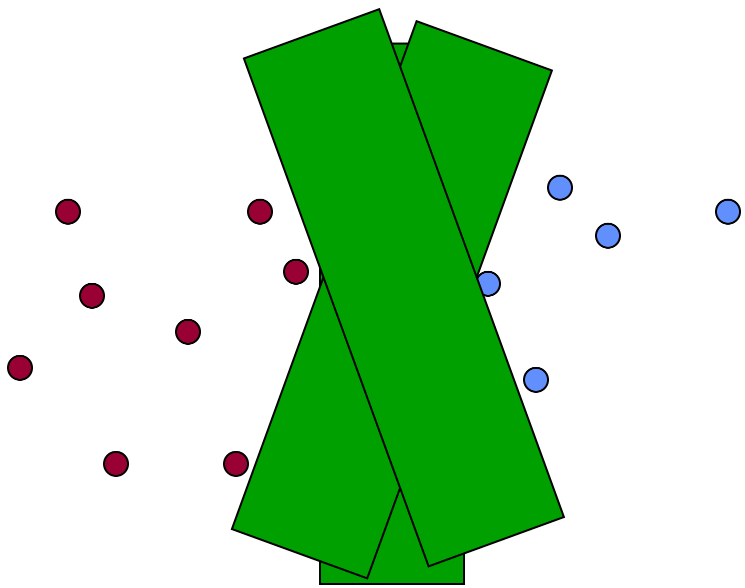
In d dimensional space, hyperplane is given by:

$$a_0 + a_1x_1 + a_2x_2 + \cdots + a_dx_d = 0$$

Image Source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Linear classifiers: Which Hyperplane?

- Lots of possible solutions for *a, b, c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of goodness]
  - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal* solution.
  - Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary
  - SVMs maximize the *margin* around the separating hyperplane.
    - A.k.a. large margin classifiers
  - The decision function is fully specified by a subset of training samples, *the support vectors*.

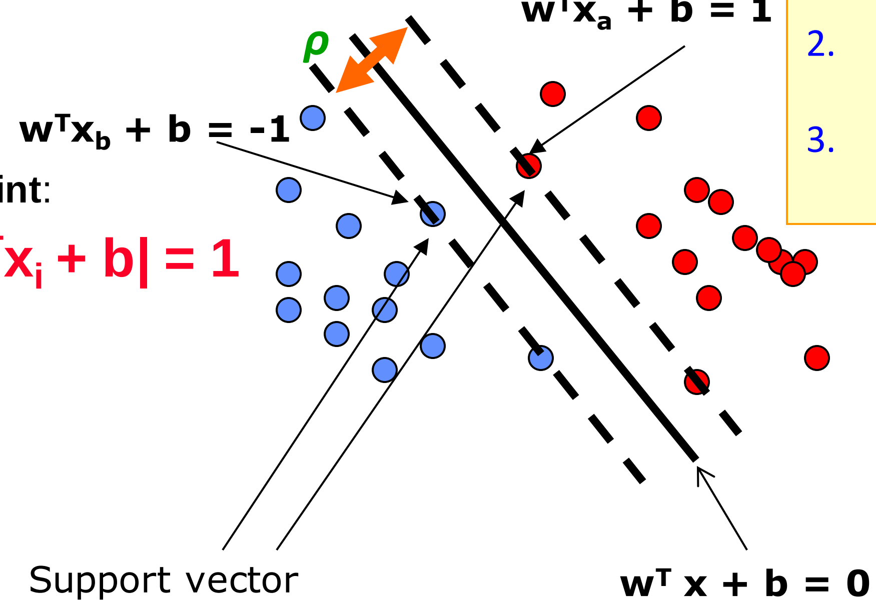


If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased

Linear Support Vector Machine (SVM)

Assumption: Data is linearly separable
Let (x_0, y_0) be a point on $w^Tx+b = 1$
Then its distance to the separating plane $w^T x+b = 0$ is:
$|w^T (x_0\ y_0) +b|/||w||= 1/||w||$

- **Hyperplane**
  $\mathbf{w}^T \mathbf{x} + b = 0$
- **Extra scale constraint**:
  $$\min_{i=1,\ldots,n} |w^Tx_i + b| = 1$$

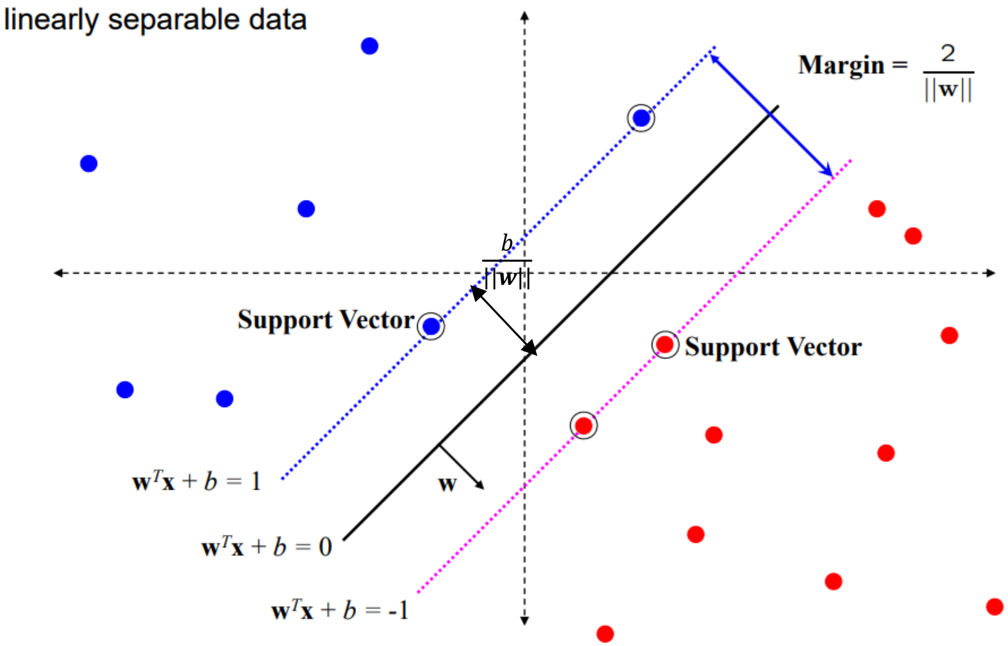


Distance between
$w^T x+b = +1$ and -1 is $\rho = 2 / ||w||$
What we did:

1. Consider all possible w with different angles
2. Scale w such that the constraints are tight
3. Pick the one with largest margin/minimal size

SVM Optimization:
Maximize: ρ or
Minimize: $\frac{1}{2}\ ||w||^2$

SVM : How to find such hyperplane?



Constraint and Objective:

1. Define the hyperplane via the following rule:

$$y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1$$

-- **Constraint**

2. Maximize margin distance:

Margin distance= $\frac{\boldsymbol{w}^T\boldsymbol{x}+b+\mathbf{1} - (\boldsymbol{w}^T\boldsymbol{x}+b-\mathbf{1})}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$

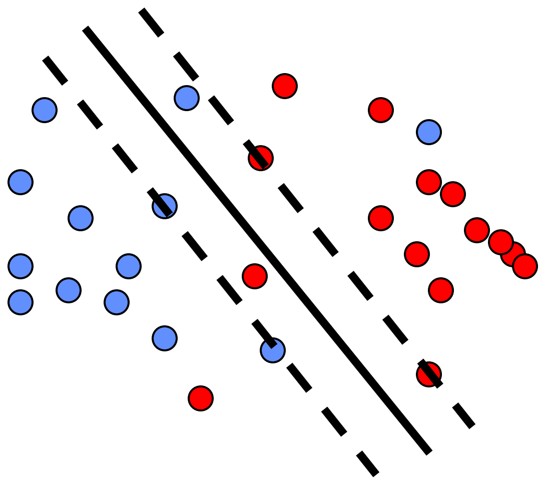
$$\max \frac{2}{||\boldsymbol{w}||} \Rightarrow \min ||\boldsymbol{w}|| \Rightarrow \min \frac{||\boldsymbol{w}||^2}{2}$$

-- **Objective**

Each training point: $\boldsymbol{x_i} \in R^d \ \forall i \in \{1, \ldots, n\}$
Label: $y \in \{-1, 1\}$
$\boldsymbol{w} \in R^d$

Maximum margins linear classifier: $\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{w}||^{\mathbf{2}}$ such that $1 \leq y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \quad \forall i \in \{1, \ldots, N\}$

Dataset with noise



- **Hard Margin:** So far we require all data points be classified correctly

  - No training error

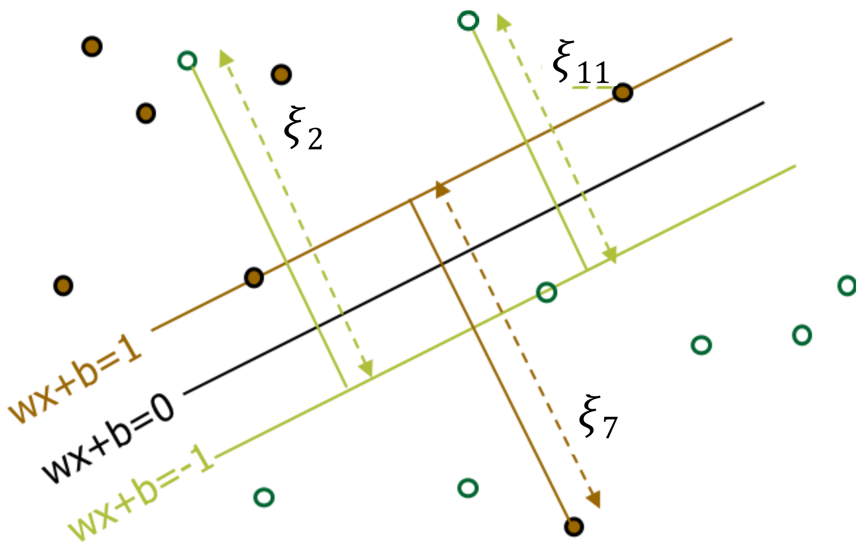
- **What if the training set is noisy?**

  - Solution: Use **Soft Margin**

The two red points and two blue points do not satisfy the criteria $y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1$

Soft Margin Classification

Slack variables $\xi_i \geq 0$ **can be added to allow misclassification of difficult or noisy examples.**



What should our optimization criterion be?

Minimize

$$½\,\mathbf{w}^{\mathrm{T}}\mathbf{w} + \mathrm{C}\sum_{j=1}^{k}\xi_j$$

The slack variables will be zero for the samples which can be classified correctly and a positive value for the remaining points.

Hard Margin v.s. Soft Margin

- **The old formulation:**

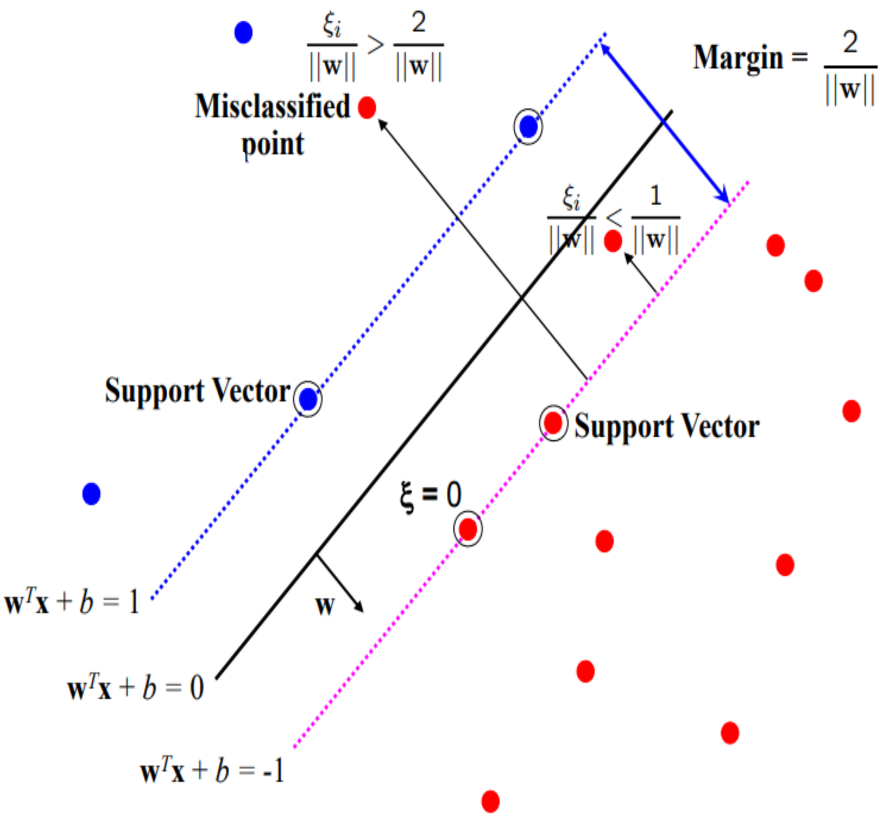
Find $\mathbf{w}$ and b such that
$\frac{1}{2}\,\mathbf{w}^{T}\mathbf{w} = \frac{1}{2}\,\|\mathbf{w}\|^{2}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\frac{1}{2}\,\mathbf{w}^{T}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\boldsymbol{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter *C* can be viewed as a way to control overfitting.**

SVM: Non-separable case



- $\xi_i \geq 0$
- $0 < \xi_i \leq 1$: point is between margin and correct side of hyper plane
  —**margin violation**
- $\xi_i > 1$: point is on the wrong side of margin
  —**misclassification**

Maximum margins linear classifier:

$$\underset{\boldsymbol{w},\xi_I \geq 0}{\text{argmin}} \frac{1}{2}||\boldsymbol{w}||^{\mathbf{2}} + C\sum_i^N \xi_i \text{ such that } 1 - \xi_i \leq y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \quad \forall i \in \{1, \ldots, N\}$$

Transformation to data

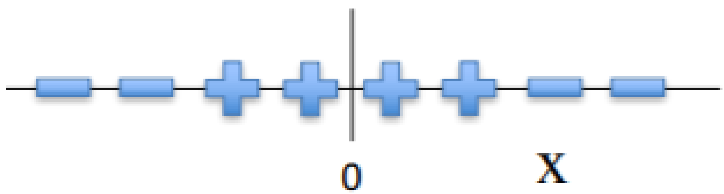


Fig 1

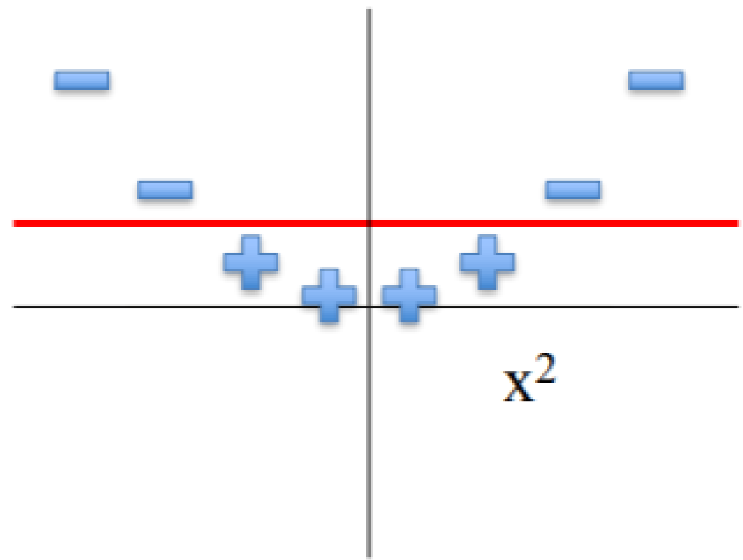
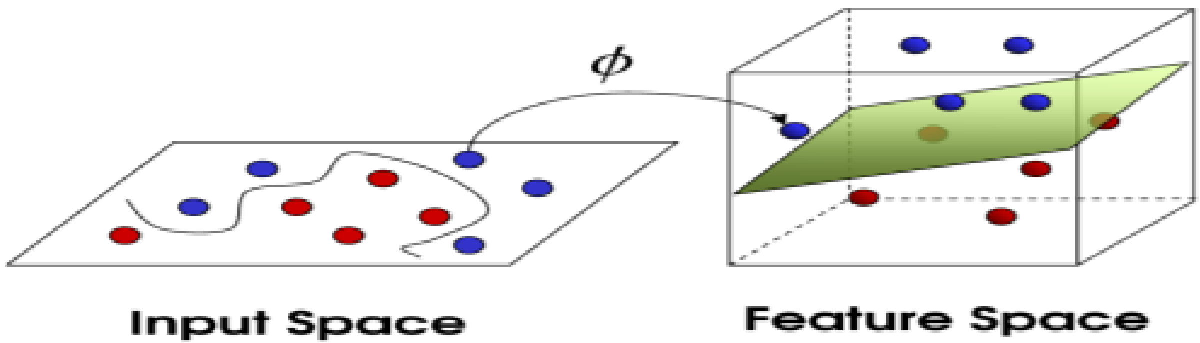


Fig 2

- Data points in Fig 1 cannot be separated using SVM
- Applying **transformation** $\Phi(\mathrm{x}) = \mathrm{x}^2$ gives data points in Fig 2
  - Can be separated by a line
- Apply SVM on transformed data

Mapping data to new feature space



$$\Phi : \mathcal{X} \mapsto \hat{\mathcal{X}} = \Phi(\mathbf{x})$$

- For example, if $x_i = [x_{i1}, x_{i2}]$, i.e., $\boldsymbol{x} \in \mathbb{R}^2$,

$\Phi(x) = [1, x_{i1}, x_{i2}, x_{i1}x_{i2}, x_{i1}^2, x_{i2}^2]$

- Run SVM on $\Phi(x)$ instead of x

Kernels

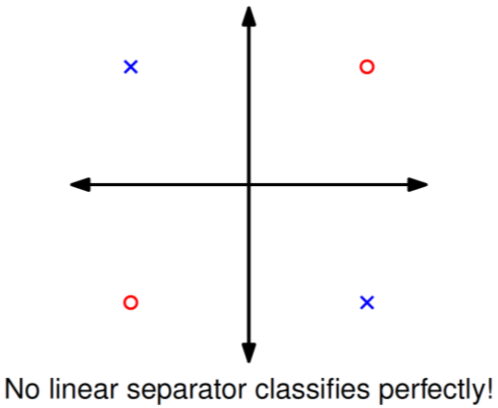
- Define Kernel $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$
- Use $K(x_i, x_j)$ in SVM
- Because of the way the solution of SVM optimization problem is computed, it is easy to use $K(x_i, x_j)$

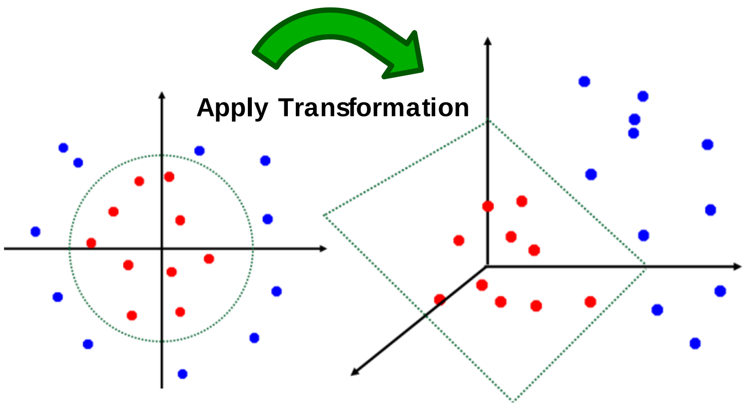
Example of commonly used Kernels ($x_i, x_j \in \mathbb{R}$)

- Polynomial kernel of order 2: $K\left(x_i, x_j\right) = 1 + x_i + x_j + x_i^2 + x_j^2 + 2x_i x_j$
- Radial Basis Function: $K\left(\mathrm{x}_i, x_j\right) = \exp\left(-\frac{\left(\mathrm{x}_i - x_j\right)^2}{2\gamma^2}\right)$

Neural Networks

- Limitations in SVM Models
  - Kernel-SVM can separate data using transformation $\phi(x)$, i.e., x is not linearly separable but $\phi(x)$ is linearly separable; think of $\phi(x)$ as the new feature
  - Identifying the appropriate kernel is difficult
  - Computation time for large problems





- Neural Network
  - Derive features of the data automatically as a constrained optimization problem

Perceptron Model

- The core of the neural network is perceptron model

Forward

$\boldsymbol{x} = [x_1, x_2, x_3]$

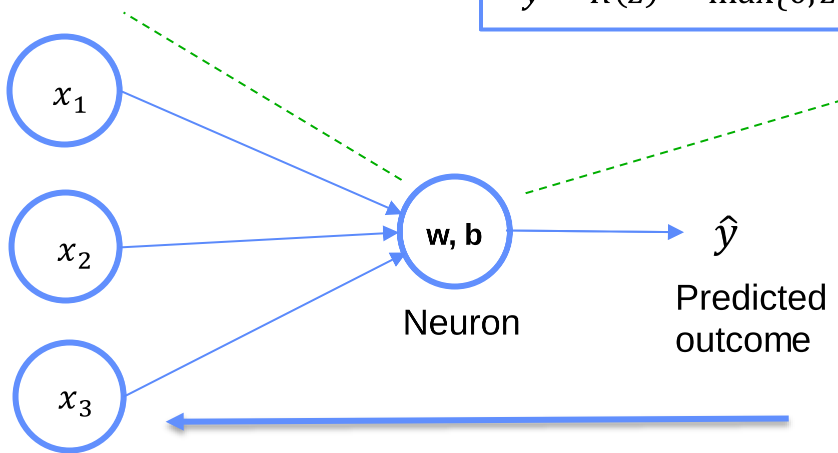
$z = \boldsymbol{w}^T\boldsymbol{x} + \mathrm{b}$

Activation Unit

$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$ (Sigmoid)

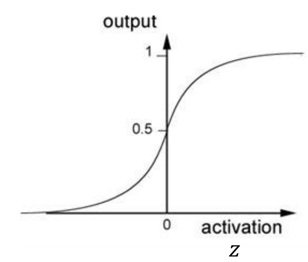
Or

$\hat{y} = R(z) = \max\{0, z\}$ (ReLU)



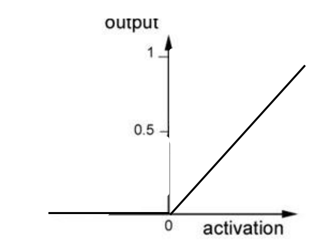
Backward

Sigmoid Function



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

ReLU Function



$$\mathrm{R}(z) = \max\{0, z\}$$

Update Rule (Backward):

$$w_{t+1} = w_t - \eta \nabla J(w_t)$$

η : Learning rate

Loss

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(w.x^{(i)}, y^{(i)})$$

Computing Gradient

$$\nabla J(\mathbf{w}_0) = \left(\frac{\partial J(\mathbf{w})}{\partial w_0}, \frac{\partial J(\mathbf{w})}{\partial w_1}, \cdots, \frac{\partial J(\mathbf{w})}{\partial w_n}\right)_{\mathbf{w}_0}$$

N: number of samples $x^{(i)}$: feature of i^{th} sample

Neural Network (Forward)

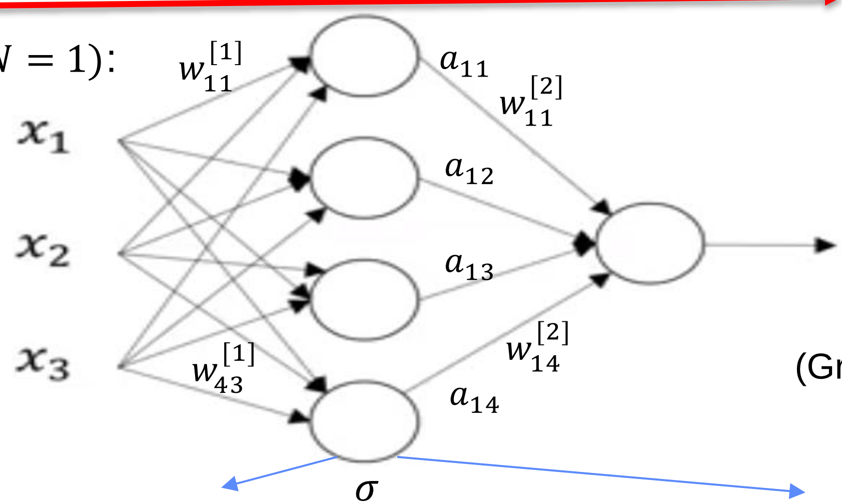
Input layer — Layer 0

Hidden layer — Layer 1

Output layer — Layer 2

Forward

Assume 1 sample $(N = 1)$:



Predicted outcome $\hat{y}$

y (Ground Truth)

MSE Loss Function:

$$L = (\hat{y} - y)^2$$

$$\underbrace{[x_1 \quad x_2 \quad x_3] \cdot \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} & w_{31}^{[1]} & w_{41}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} & w_{32}^{[1]} & w_{42}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} & w_{33}^{[1]} & w_{43}^{[1]} \end{bmatrix} + [b_1^{[1]} \quad b_2^{[1]} \quad b_3^{[1]} \quad b_4^{[1]}] = \boldsymbol{z_1}} \rightarrow \underbrace{\boldsymbol{a_1}}$$

$$\boldsymbol{z_1} = [z_{11} \; z_{12} \; z_{13} \; z_{14}]$$

Where, $\boldsymbol{a_1} = [a_{11} \; a_{12} \; a_{13} \; a_{14}]$ $\qquad \boldsymbol{a_{ij}} = \boldsymbol{\sigma}(\boldsymbol{z_{ij}})$

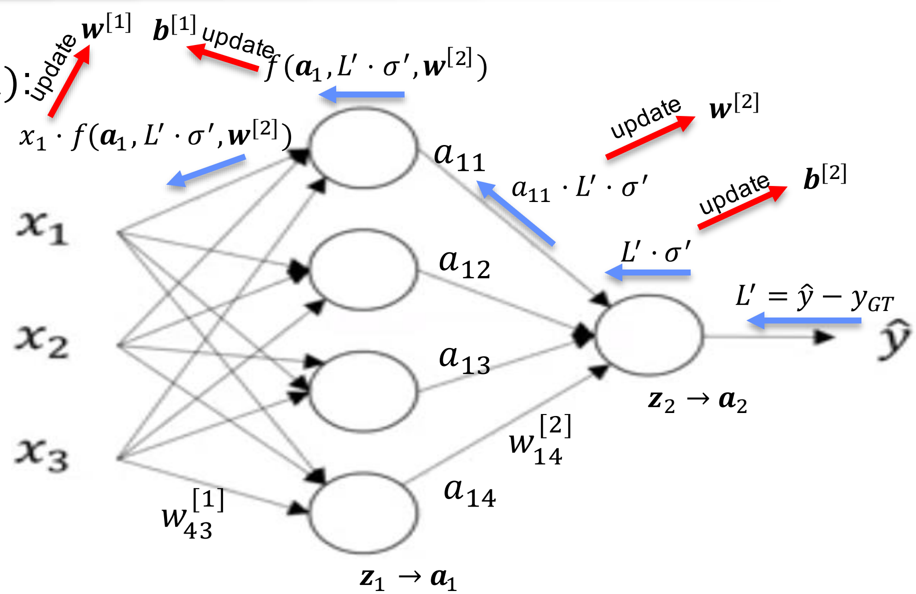
Backpropagation: Gradient Descent

Input layer | Hidden layer | Output layer

Layer 0 | Layer 1 | Layer 2

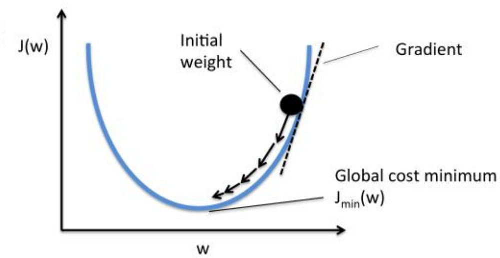
Backward

Assume 1 sample $(N = 1)$:



Derivative of Sigmoid function $\sigma(z)$:

$$\begin{aligned}
\sigma'(z) = \frac{\partial \sigma(z)}{\partial z} &= \frac{\partial}{\partial z} \frac{1}{1+e^{-z}} \\
&= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} \\
&= \sigma(z) - \sigma^2(z) \\
&= \sigma(z)\big(1-\sigma(z)\big)
\end{aligned}$$



Purpose of backpropagation:

Apply *chain rule of derivative* to update parameters: $\boldsymbol{w}^{[1]}, \boldsymbol{b}^{[1]}, \boldsymbol{w}^{[2]}, \boldsymbol{b}^{[2]}$