Machine Learning
CS 6830

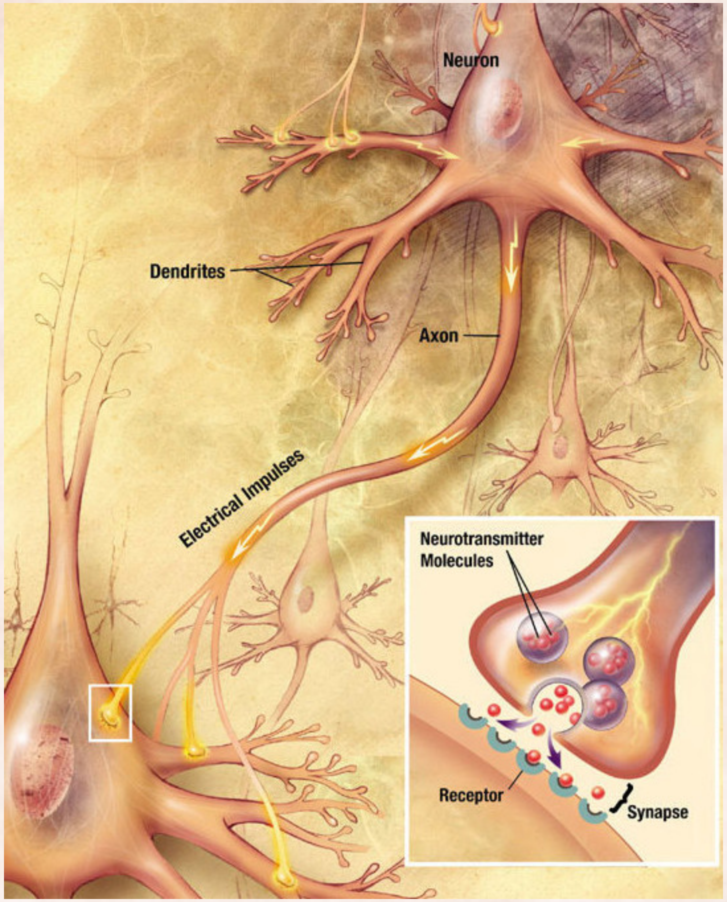
Lecture 03b

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

Neurons



Soma is the central part of the neuron:

- *where the input signals are combined.*

Dendrites are cellular extensions:

- *where majority of the input occurs.*

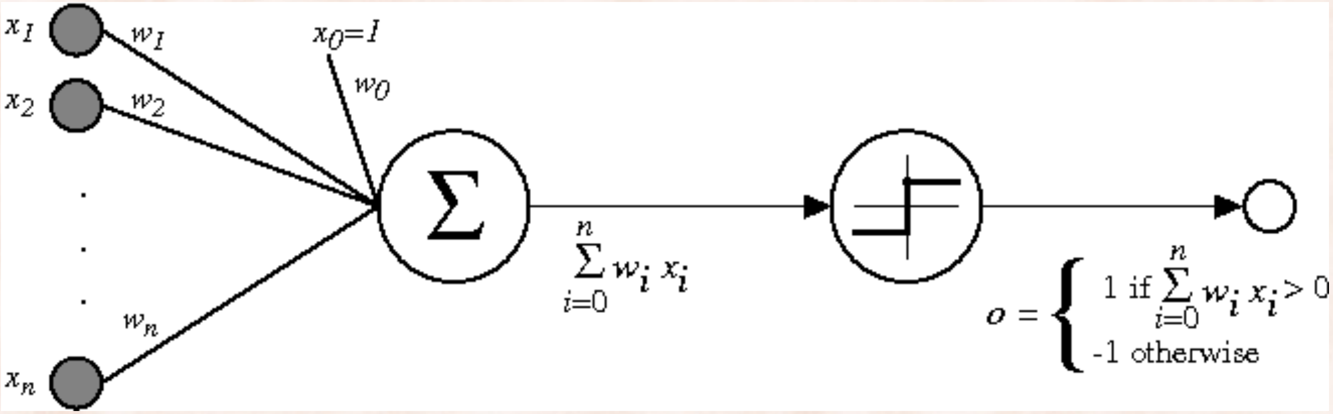
Axon is a fine, long projection:

- *carries nerve signals to other neurons.*

Synapses are molecular structures between axon terminals and other neurons:

- *where the communication thakes place.*

Neurons & Perceptrons



- Biological Interpretation:
  - The output of the neuron is a linear combination of inputs from other neurons, rescaled by the synaptic weights.
    - It is often transformed through a monotonic function such as *signum*, or *sigmoid*
  - Weights w_i correspond to the synaptic weights (activating or inhibiting).
  - Summation corresponds to combination of signals in the soma.

The Perceptron Algorithm: Two Classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $i = 1 \ldots n$
3. $\quad y_i = sgn(\mathbf{w}^T \varphi(\mathbf{x}_i))$
4. $\quad$ **if** $y_i \neq t_i$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + t_i \varphi(\mathbf{x}_i)$

Repeat:
a) until convergence.
b) for a number of epochs E.

Theorem [Rosenblatt, 1962]:

If the training dataset is **linearly separable**, the perceptron learning algorithm is guaranteed to find a solution in a finite number of steps.

- see Theorem 1 (Block, Novikoff) in [Freund & Schapire, 1999].

Motivation: Error function minimization

- Error: total number of misclassified patterns?
  - piecewise constant function of **w** with discontinuities.
  - cannot use gradient methods (gradient zero almost everywhere).

- The *Perceptron Criterion*:
  - Assume classes $T = \{c_1, c_2\} = \{-1,+1\}$.
  - Want $\mathbf{w}^T\varphi(\mathbf{x}_n) \geq 0$ for $t_n = +1$, and $\mathbf{w}^T\varphi(\mathbf{x}_n) < 0$ for $t_n = -1$.
  - $\Rightarrow$ would like to have $\mathbf{w}^T\varphi(\mathbf{x}_n)\, t_n > 0$ for all patterns.
  - $\Rightarrow$ want to minimize $-\mathbf{w}^T\varphi(\mathbf{x}_n)\, t_n$ for all misclassified patterns.

$$\Rightarrow \text{minimize } E_P(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}^T \varphi(\mathbf{x}_n) t_n$$

The Perceptron Algorithm as Stochastic Gradient Descent

- Update parameters $\mathbf{w}$ sequentially:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}^{(\tau)}, x_n)$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \varphi(x_n) t_n$$

- The magnitude of $\mathbf{w}$ is inconsequential => $\eta = 1$.

The Perceptron Algorithm: K classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $i = 1 \ldots n$
3. $\quad y_i = \arg\max_{t \in T} \mathbf{w}^T \varphi(\mathbf{x}_i, t)$
4. $\quad$ **if** $y_i \neq t_i$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + \varphi(\mathbf{x}_i, t_i) - \varphi(\mathbf{x}_i, y_i)$

Repeat:

a) until convergence.

b) for a number of epochs E.

During testing:

$$t^* = \arg\max_{t \in T} \mathbf{w}^T \phi(\mathbf{x}, t)$$

Averaged Perceptron

1. **initialize** parameters $\mathbf{w} = 0$, $\tau = 1$, $\overline{\mathbf{w}} = 0$
2. **for** $i = 1 \ldots n$
3. $\quad y_i = sgn(\mathbf{w}^T \varphi(\mathbf{x}_i))$
4. $\quad$ **if** $y_i \neq t_i$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + t_i \varphi(\mathbf{x}_i)$
6. $\quad \overline{\mathbf{w}} = \overline{\mathbf{w}} + \mathbf{w}$
7. $\quad \tau = \tau + 1$
8. **return** $\overline{\mathbf{w}} / \tau$

Repeat:
a) until convergence.
b) for a number of epochs E.

During testing: $t^* = \operatorname{sgn} \overline{\mathbf{w}}^T \phi(\mathbf{x})$

Averaged Perceptron: K classes

1. **initialize** parameters $\mathbf{w} = 0$, $\tau = 1$, $\overline{\mathbf{w}} = 0$
2. **for** $i = 1 \ldots n$
3. $\quad y_{\mathrm{i}} = \arg\max_{t \in T} \mathbf{w}^T \varphi(\mathbf{x}_i, t)$
4. $\quad$ **if** $y_{\mathrm{i}} \neq t_i$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + \varphi(\mathbf{x}_{\mathrm{i}}, t_i) - \varphi(\mathbf{x}_{\mathrm{i}}, y_{\mathrm{i}})$
6. $\quad \overline{\mathbf{w}} = \overline{\mathbf{w}} + \mathbf{w}$
7. $\quad \tau = \tau + 1$
8. **return** $\overline{\mathbf{w}} / \tau$

Repeat:
a) until convergence.
b) for a number of epochs E.

During testing: $t^* = \arg\max_{t \in T} \overline{\mathbf{w}}^T \varphi(\mathbf{x}, t)$

The Perceptron Algorithm: Two Classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $i = 1 \ldots n$
3. $\quad y_{\mathrm{i}} = sgn(\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_{\mathrm{i}}))$
4. $\quad$ **if** $y_{\mathrm{i}} \neq t_i$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + t_i\varphi(\mathbf{x}_{\mathrm{i}})$

Repeat:

a) until convergence.

b) for a number of epochs E.

Loop invariant: **w** is a weighted sum of training vectors:

$$\mathbf{w} = \sum_i \alpha_i t_i \phi(\mathbf{x}_i) \quad \Rightarrow \quad \mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i t_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

Kernel Perceptron: Two Classes

1. **define** $f(\mathbf{x}) = \sum_j \alpha_j t_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}) = \sum_j \alpha_j t_j K(\mathbf{x}_j, \mathbf{x})$
2. **initialize** dual parameters $\alpha_i = 0$
3. **for** $i = 1 \ldots n$
4. $\quad y_i = sgn\, f(\mathbf{x}_i)$
5. $\quad$ **if** $y_i \neq t_i$ **then**
6. $\quad\quad \alpha_i = \alpha_i + 1$

During testing: $t = sgn\, f(\mathbf{x})$

Kernel Perceptron: Two Classes

1. **define** $f(\mathbf{x}) = \sum_j \alpha_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}) = \sum_j \alpha_j K(\mathbf{x}_j, \mathbf{x})$
2. **initialize** dual parameters $\alpha_i = 0$
3. **for** $i = 1 \ldots n$
4. $\quad y_i = sgn\, f(\mathbf{x}_\mathrm{i})$
5. $\quad$ **if** $y_i \neq t_i$ **then**
6. $\quad\quad \alpha_i = \alpha_i + t_i$

During testing: $t = sgn\, f(\mathbf{x})$

The Perceptron Algorithm: K classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $i = 1 \ldots n$
3. $\quad c_j = \arg\max_{t \in T} \mathbf{w}^T \varphi(\mathbf{x}_i, t)$
4. $\quad$ **if** $c_j \neq t_i$ **then**
5. $\quad\quad \mathbf{w} = \mathbf{w} + \varphi(\mathbf{x}_i, t_i) - \varphi(\mathbf{x}_i, c_j)$

Repeat:
a) until convergence.
b) for a number of epochs E.

Loop invariant: **w** is a weighted sum of training vectors:

$$\mathbf{w} = \sum_{i,j} \alpha_{ij} (\phi(\mathbf{x}_i, t_i) - \phi(\mathbf{x}_i, c_j))$$

$$\Rightarrow \quad \mathbf{w}^T \phi(\mathbf{x}, t) = \sum_{i,j} \alpha_{ij} (\phi(\mathbf{x}_i, t_i)^T \phi(\mathbf{x}, t) - \phi(\mathbf{x}_i, c_j)^T \phi(\mathbf{x}, t))$$

Kernel Perceptron: K classes

1. **define** $f(\mathbf{x},t) = \sum_{i,j} \alpha_{ij}(\phi(\mathbf{x}_i, t_i)^T \phi(\mathbf{x},t) - \phi(\mathbf{x}_i, c_j)^T \phi(\mathbf{x},t))$
2. **initialize** dual parameters $\alpha_{ij} = 0$
3. **for** $i = 1 \ldots n$
4. $\quad c_j = \arg\max_{t \in T} f(\mathbf{x}_i, t)$
5. $\quad$ **if** $y_i \neq t_i$ **then**
6. $\quad\quad \alpha_{ij} = \alpha_{ij} + 1$

Repeat:
a) until convergence.
b) for a number of epochs E.

During testing:

$$t^* = \arg\max_{t \in T} f(\mathbf{x}, t)$$

Kernel Perceptron: K classes

- Discriminant function:

$$f(\mathbf{x},t) = \sum_{i,j} \alpha_{i,j} (\phi(\mathbf{x}_i, t_i)^T \phi(\mathbf{x},t) - \phi(\mathbf{x}_i, c_j)^T \phi(\mathbf{x},t))$$

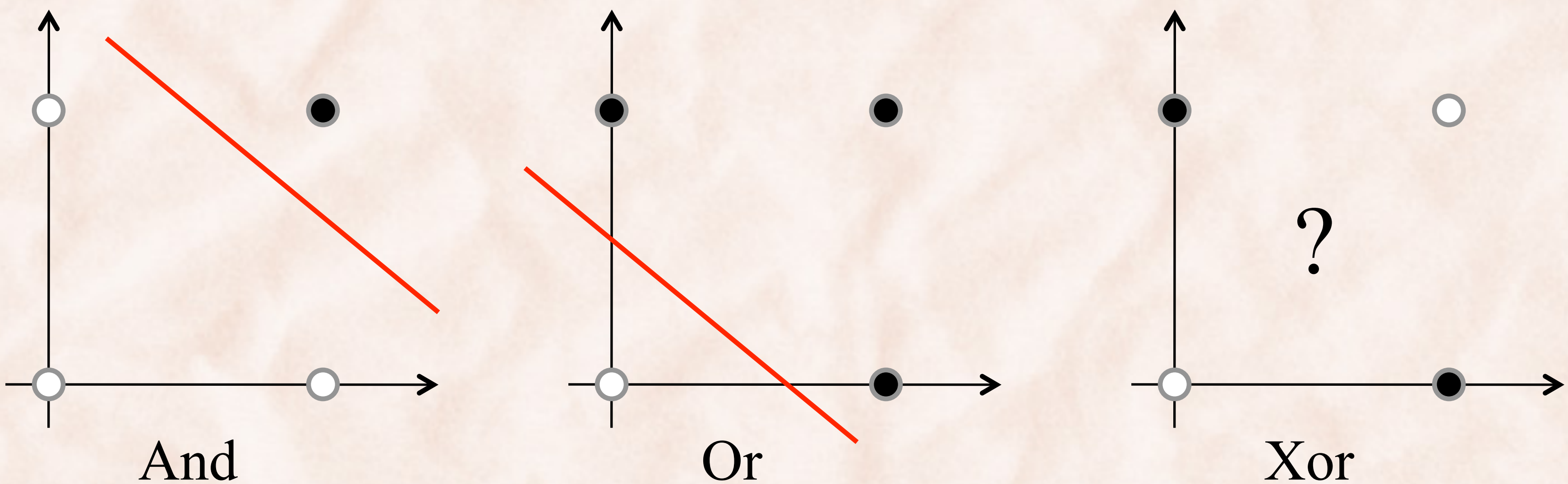
$$= \sum_{i,j} \alpha_{ij} (K(\mathbf{x}_i, t_i, \mathbf{x}, t) - K(\mathbf{x}_i, c_j, \mathbf{x}, t))$$

where:

$$K(\mathbf{x}_i, t_i, \mathbf{x}, t) = \varphi^T(\mathbf{x}_i, t_i) \varphi(\mathbf{x}, t)$$

$$K(\mathbf{x}_i, y_i, \mathbf{x}, t) = \phi^T(\mathbf{x}_i, y_i) \phi(\mathbf{x}, t)$$

The Perceptron vs. Boolean Functions



$$\left.\begin{aligned}\varphi(\mathbf{x}) &= [1, x_1, x_2]^T \\ \mathbf{w} &= [w_0, w_1, w_2]^T\end{aligned}\right\} \Rightarrow \mathbf{w}^T\varphi(\mathbf{x}) = [w_1, w_2]^T[x_1, x_2] + w_0$$

Perceptron with Quadratic Kernel

- Discriminant function:

$$f(\mathbf{x}) = \sum_i \alpha_i t_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) = \sum_i \alpha_i t_i K(\mathbf{x}_i, \mathbf{x})$$

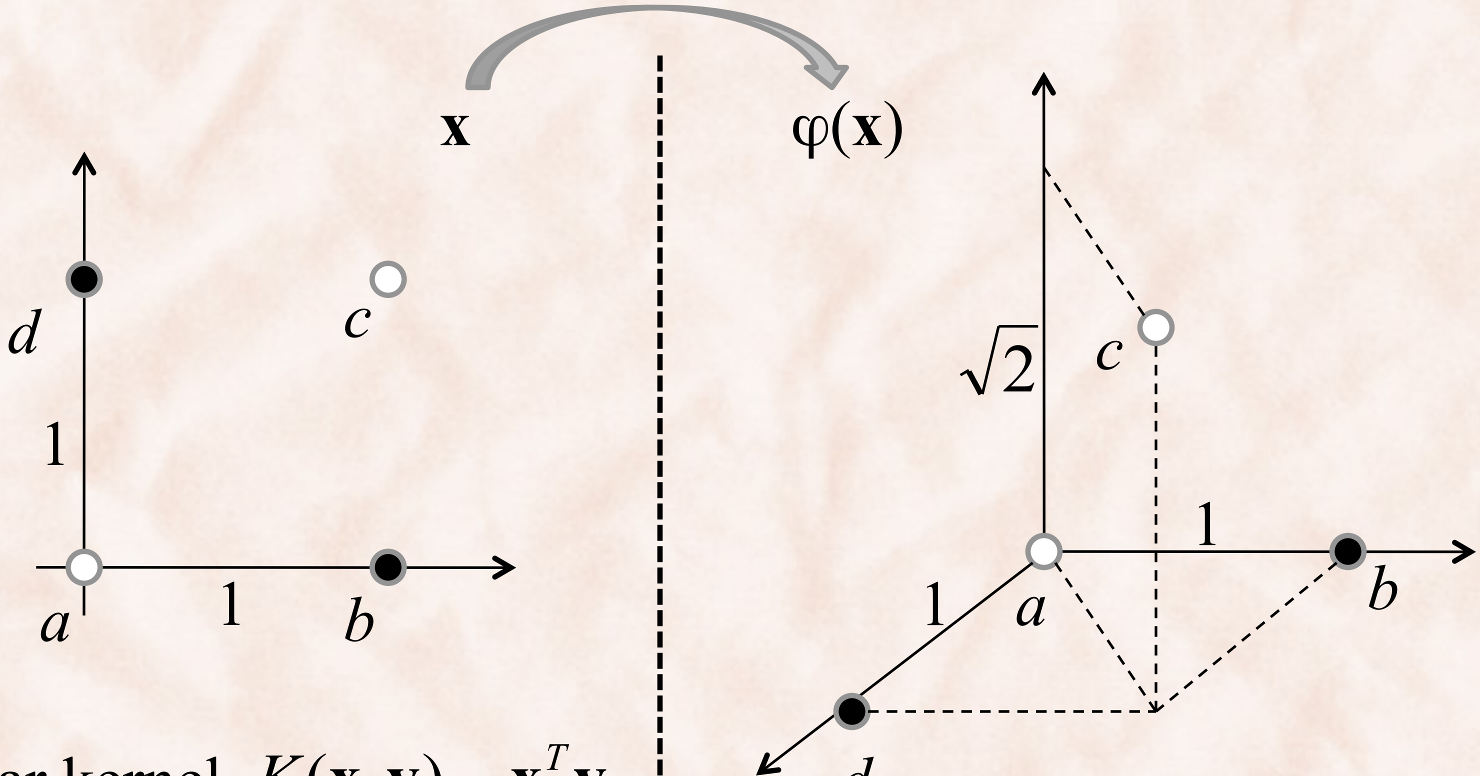
- Quadratic kernel:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2 = (x_1 y_1 + x_2 y_2)^2$$

⇒ corresponding feature space $\varphi(\mathbf{x}) = ?$

conjunctions of two atomic features

Perceptron with Quadratic Kernel



Linear kernel $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$

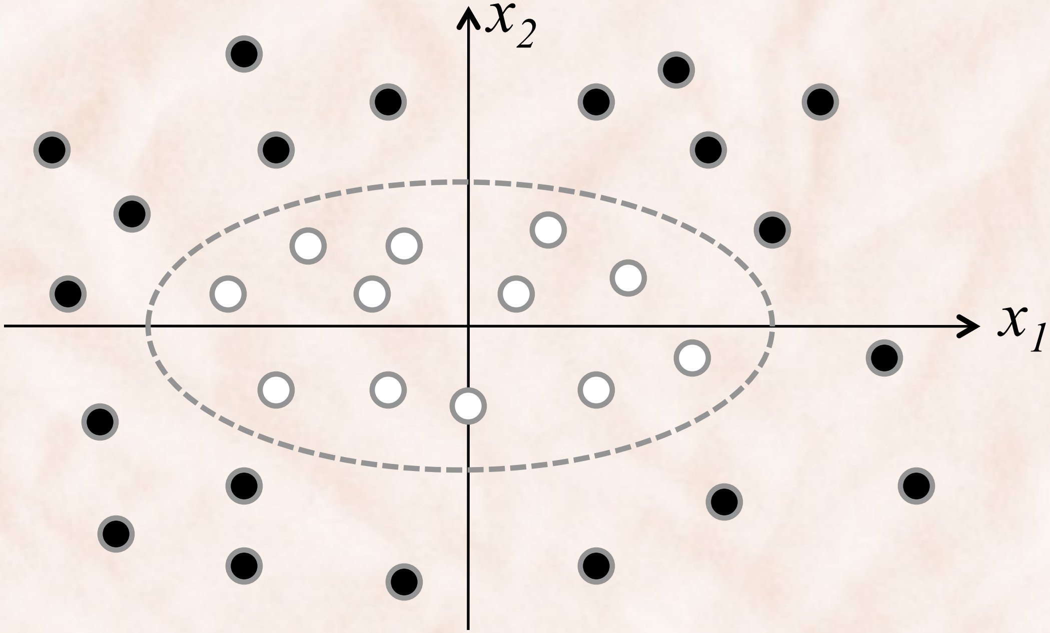
Quadratic kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2$

Quadratic Kernels

- Circles, hyperbolas, and ellipses as separating surfaces:

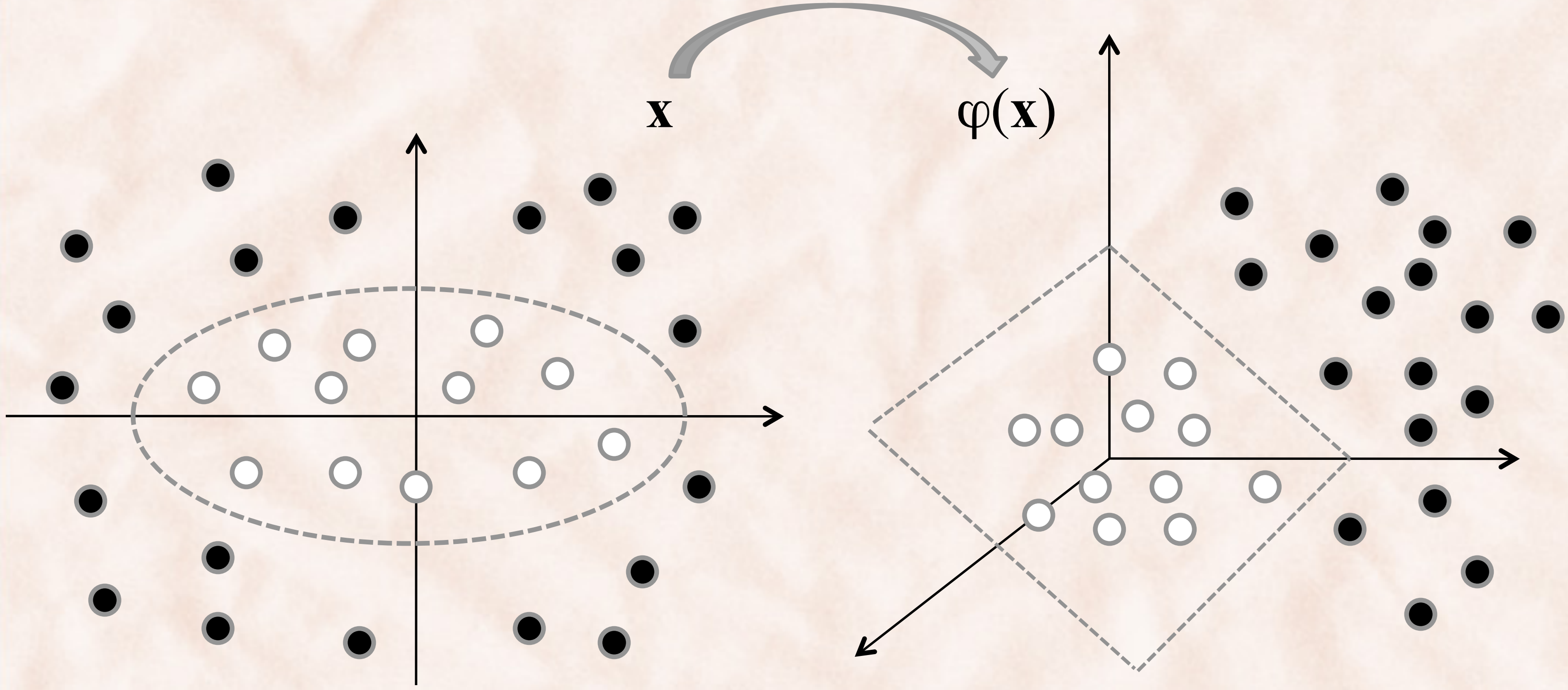
$$K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^T\mathbf{y})^2 = \varphi(x)^T\varphi(y)$$

$$\varphi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$$



Quadratic Kernels

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2 = \varphi(\mathbf{x})^T\varphi(\mathbf{y})$$



Explicit Features vs. Kernels

- Explicitly enumerating features can be prohibitive:
  - 1,000 basic features for $\mathbf{x}^T\mathbf{y}$ => 500,500 quadratic features for $(\mathbf{x}^T\mathbf{y})^2$
  - Much worse for higher order features.

- Solution:
  - Do not compute the feature vectors, compute kernels instead (i.e. compute dot products between implicit feature vectors).
    - $(\mathbf{x}^T\mathbf{y})^2$ takes 1001 multiplications.
    - $\varphi(\mathbf{x})^T \varphi(\mathbf{y})$ in feature space takes 500,500 multiplications.

Kernel Functions

- Definition:

  A function $k : X \times X \rightarrow R$ is a kernel function if there exists a feature mapping $\varphi : X \rightarrow R^n$ such that:

  $$k(\mathbf{x},\mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y})$$

- Theorem:

  $k : X \times X \rightarrow R$ is a valid kernel $\Leftrightarrow$ the Gram matrix K whose elements are given by $k(\mathbf{x}_n,\mathbf{x}_m)$ is positive semidefinite for all possible choices of the set $\{\mathbf{x}_n\}$.

Kernel Examples

- Linear kernel: $K(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{y}$
- Quadratic kernel: $K(\mathbf{x},\mathbf{y}) = (c + \mathbf{x}^T\mathbf{y})^2$
  - contains constant, linear terms and terms of order two (c > 0).
- Polynomial kernel: $K(\mathbf{x},\mathbf{y}) = (c + \mathbf{x}^T\mathbf{y})^M$
  - contains all terms up to degree M (c > 0).
- Gaussian kernel: $K(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2)$
  - corresponding feature space has infinite dimensionality.

Kernels over Discrete Structures

- Subsequence Kernels [Lodhi et al., JMLR 2002]:
  - Σ is a finite alphabet (set of symbols).
  - $\mathbf{x},\mathbf{y}\in\Sigma^*$ are two sequences of symbols with lengths $|\mathbf{x}|$ and $|\mathbf{y}|$
  - $k(\mathbf{x},\mathbf{y})$ is defined as the number of common substrings of length n.
  - $k(\mathbf{x},\mathbf{y})$ can be computed in $O(n|\mathbf{x}||\mathbf{y}|)$ time complexity.

- Tree Kernels [Collins and Duffy, NIPS 2001]:
  - T_1 and T_2 are two trees with N_1 and N_2 nodes respectively.
  - $k(T_1, T_2)$ is defined as the nummber of common subtrees.
  - $k(T_1, T_2)$ can be computed in $O(N_1N_2)$ time complexity.
  - in practice, time is linear in the size of the trees.

Reading Assignment

- Chapter 6:
  - Section 6.1 on dual representations for linear regression models.
  - Section 6.2 on techniques for constructing new kernels.