# Data Classification (b)

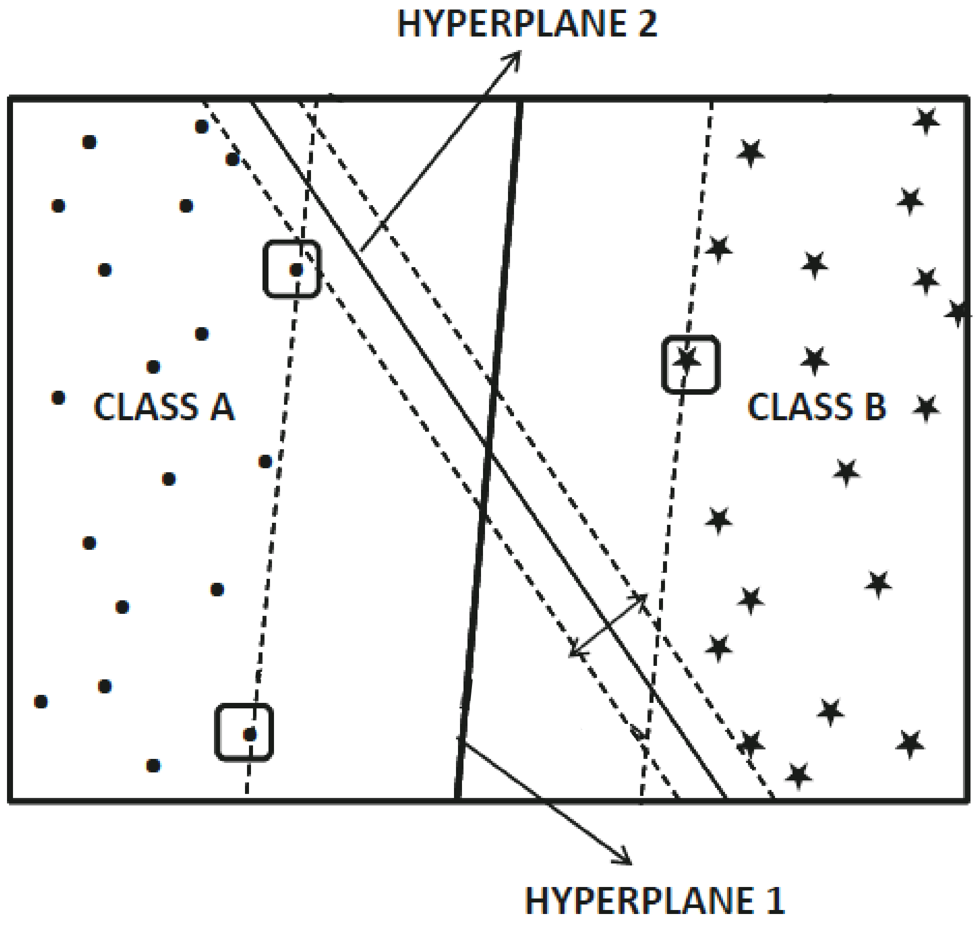
Lijun Zhang
zlj@nju.edu.cn
http://cs.nju.edu.cn/zlj

## Outline

- **Support Vector Machines**
- Neural Networks
- Instance-Based Learning
- Classifier Evaluation
- Summary

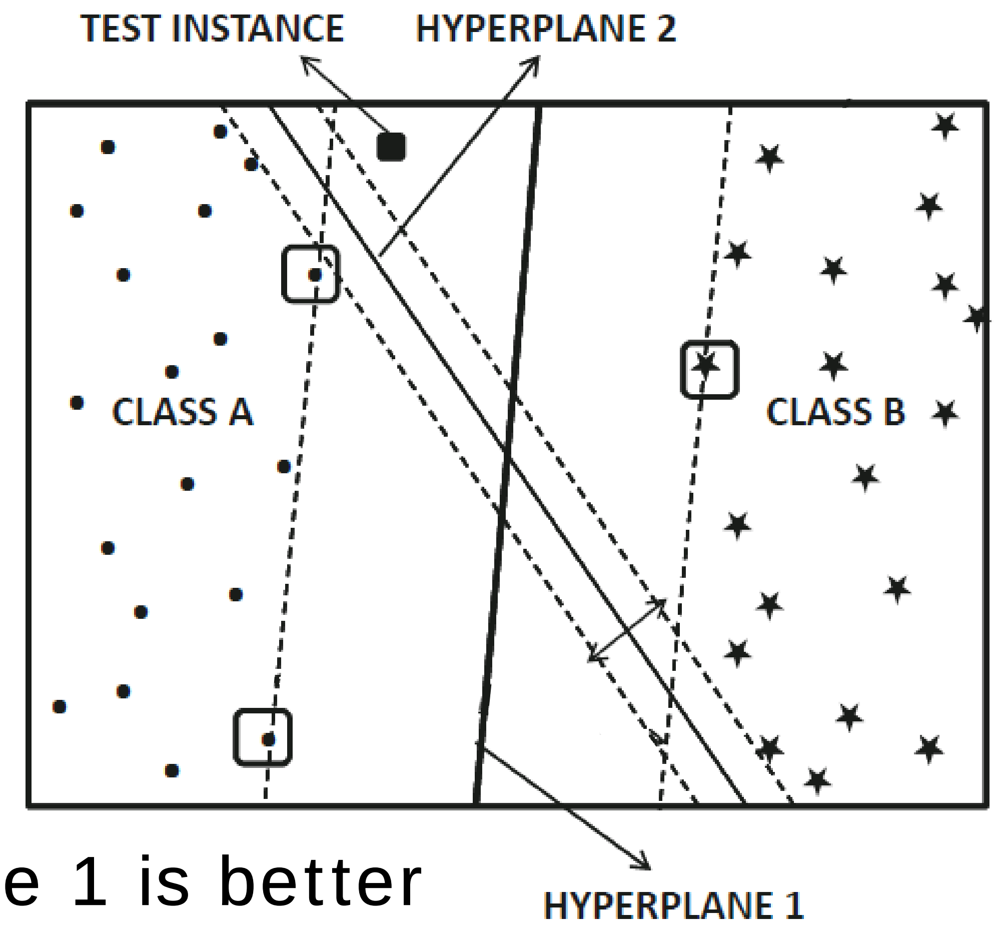
# SVM for Linearly Separable Data (1)

- Hyperplane 1 v.s. Hyperplane 2



## SVM for Linearly Separable Data (2)

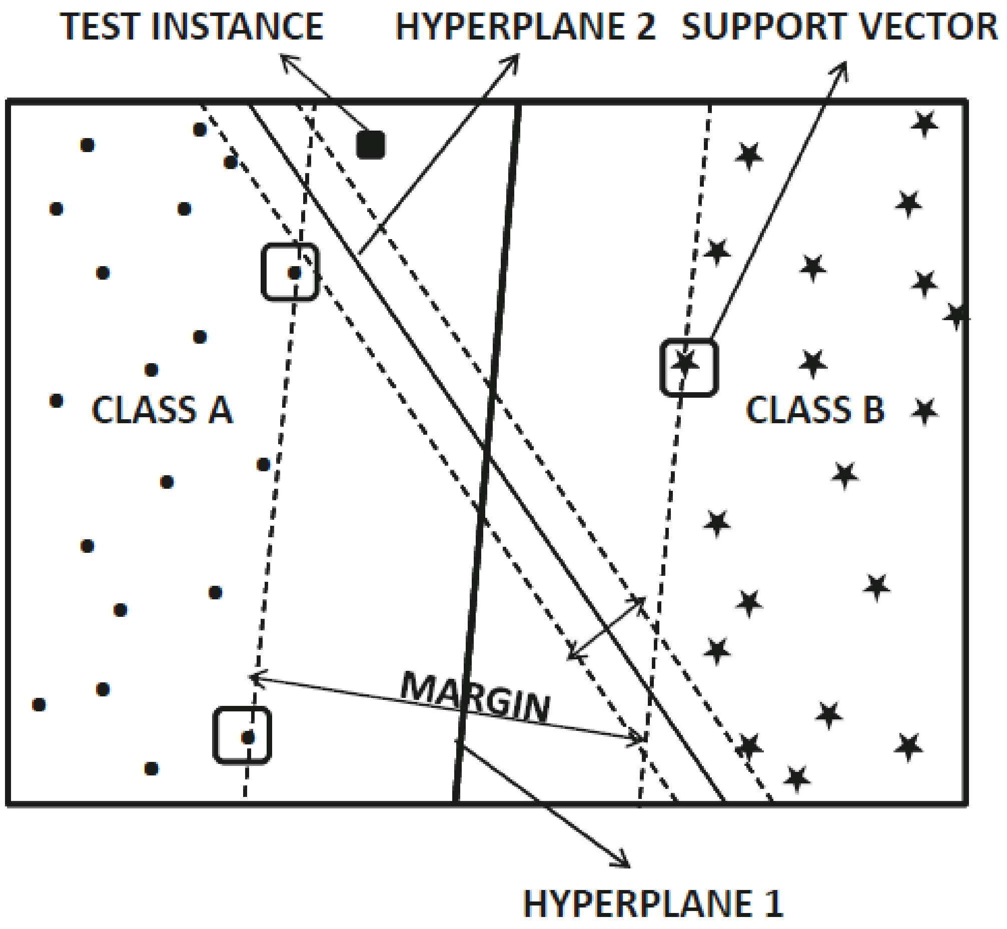
- Hyperplane 1 v.s. Hyperplane 2



  - Hyperplane 1 is better

# SVM for Linearly Separable Data (3)

- Hyperplane 1 v.s. Hyperplane 2
  - Margin
  - Support vectors



## SVM for Linearly Separable Data (4)

- The Observation
  - Larger margin provides better generalization power
- The Goal of SVM
  - Find the maximum margin hyperplane
- Notations
  - Training set $\mathcal{D} = \{(\overline{X_1}, y_1), \ldots, (\overline{X_n}, y_n)\}$, where $\overline{X}_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$
  - The separating hyperplane

$$\overline{W} \cdot \overline{X} + b = 0.$$

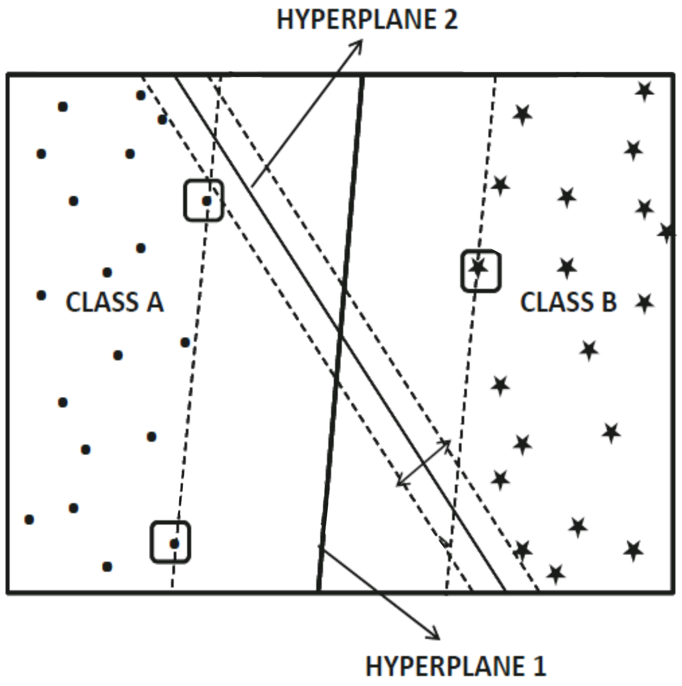
Where $\overline{W} = (w_1 \ldots w_d)$

# Training (1)

- Basic Constraints

$$\overline{W} \cdot \overline{X_i} + b \geq 0 \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \leq 0 \quad \forall i : y_i = -1$$

  - There may be infinite solutions



## Training (1)

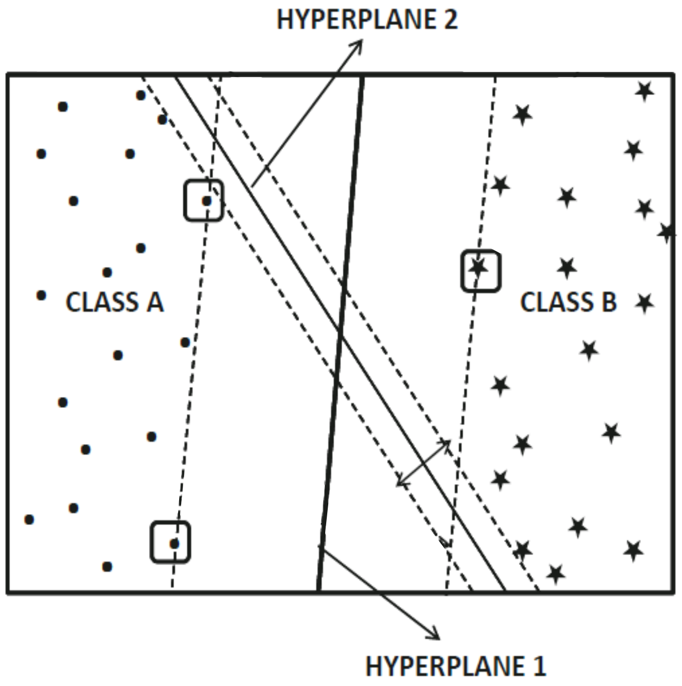
- Basic Constraints

$$\overline{W}\cdot\overline{X_i}+b\geq 0 \quad \forall i: y_i=+1$$
$$\overline{W}\cdot\overline{X_i}+b\leq 0 \quad \forall i: y_i=-1$$

  - There may be infinite solutions

- Margin Constraints
  - $\bar{W}\cdot\bar{X}+b=0$ is in the center
  - Two hyperplanes

$$\begin{aligned}\overline{W}\cdot\overline{X}+b&=+c\\ \overline{W}\cdot\overline{X}+b&=-c\end{aligned} \Rightarrow \begin{aligned}\overline{W}\cdot\overline{X}+b&=+1\\ \overline{W}\cdot\overline{X}+b&=-1.\end{aligned}$$



## Training (2)

- Basic Constraints

$$\overline{W}\cdot\overline{X_i}+b\geq 0 \quad \forall i: y_i=+1$$
$$\overline{W}\cdot\overline{X_i}+b\leq 0 \quad \forall i: y_i=-1$$

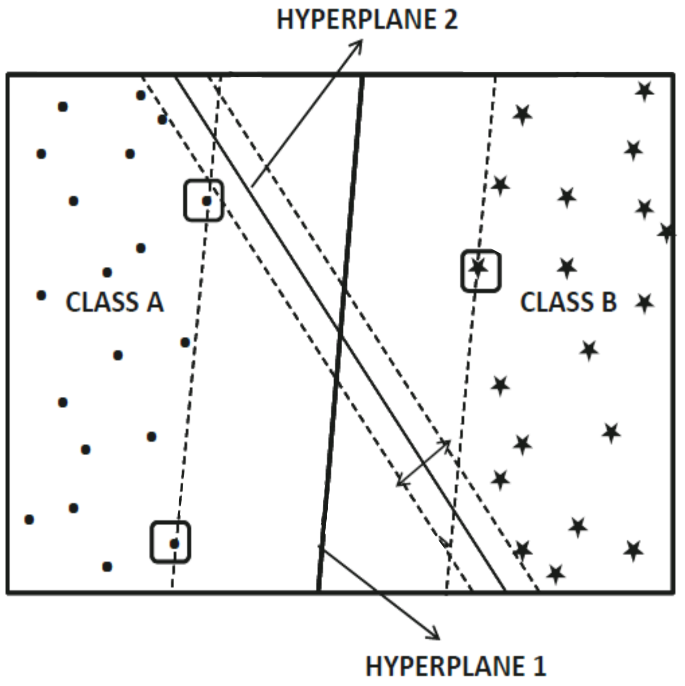
  - There may be infinite solutions

- Margin Constraints

$$\overline{W}\cdot\overline{X_i}+b\geq +1 \quad \forall i: y_i=+1$$
$$\overline{W}\cdot\overline{X_i}+b\leq -1 \quad \forall i: y_i=-1.$$

$$\Downarrow$$

$$y_i(\overline{W}\cdot\overline{X_i}+b)\geq +1 \quad \forall i.$$

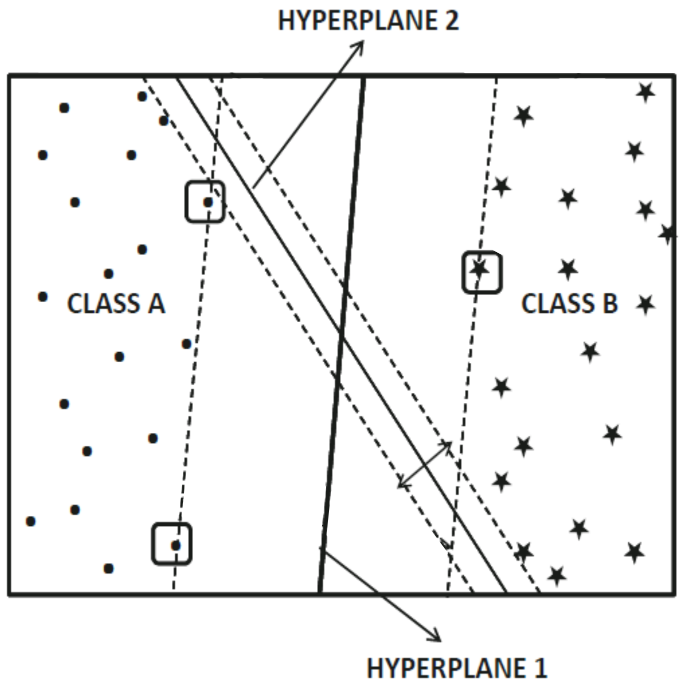


## Training (3)

- Distance between Two Hyperplanes

$$\frac{2}{\|\bar{W}\|_2}$$

https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_plane



- The Problem

$$\max_{\bar{W}\in\mathbb{R}^d, b\in\mathbb{R}} \quad \frac{2}{\|\bar{W}\|_2}$$

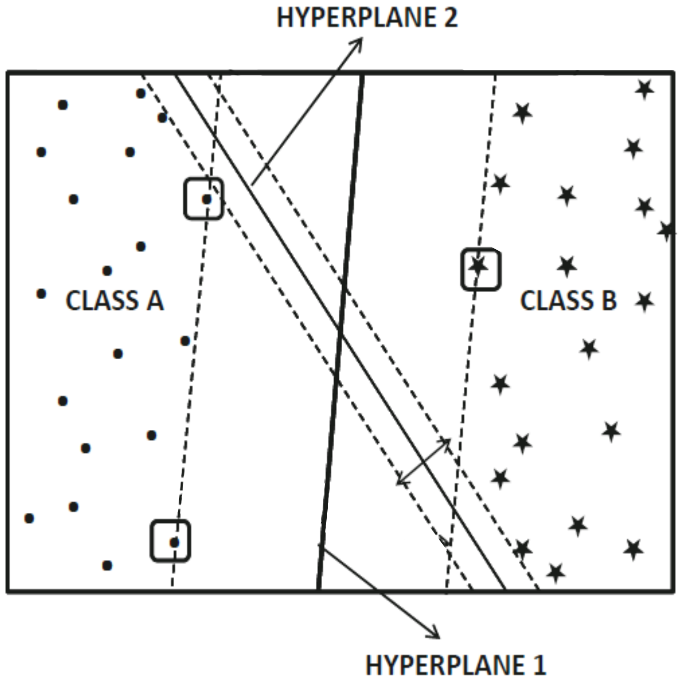
$$\text{s.t.} \quad y_i\langle \bar{W}\cdot \bar{X}_i + b\rangle \geq 1, \forall i$$

## Training (4)

- Distance between Two Hyperplanes

$$\frac{2}{\|\bar{W}\|_2}$$

https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_plane



- Reformulation

$$\min_{\bar{W}\in\mathbb{R}^d, b\in\mathbb{R}} \quad \frac{\|\bar{W}\|_2^2}{2}$$

$$\text{s.t.} \quad y_i\langle \bar{W}\cdot\bar{X}_i + b\rangle \geq 1, \forall i$$

## Optimization (1)

- Lagrangian Relaxation

$$L_P = \frac{||\overline{W}||^2}{2} - \sum_{i=1}^{n} \lambda_i \left[ y_i(\overline{W} \cdot \overline{X_i} + b) - 1 \right]$$

  - Introduce a $\lambda_i \geq 0$ for $y_i \langle \overline{W} \cdot \overline{X}_i + b \rangle \geq 1$

- Lagrange dual function

$$\min_{\overline{W},b} L_P = \frac{||\overline{W}||^2}{2} - \sum_{i=1}^{n} \lambda_i \left[ y_i(\overline{W} \cdot \overline{X_i} + b) - 1 \right]$$

  - The minimization problem is unconstrained

# Optimization (2)

- Lagrange dual function

$$\min_{\overline{W},b} L_P = \frac{||\overline{W}||^2}{2} - \sum_{i=1}^{n} \lambda_i \left[ y_i(\overline{W} \cdot \overline{X_i} + b) - 1 \right]$$

  - Closed form solution for $\overline{W}$

$$\nabla L_P = \nabla \frac{||\overline{W}||^2}{2} - \nabla \sum_{i=1}^{n} \lambda_i \left[ y_i(\overline{W} \cdot \overline{X_i} + b) - 1 \right] = 0$$

$$\overline{W} - \sum_{i=1}^{n} \lambda_i y_i \overline{X_i} = 0.$$

  - For $b$, we obtain

$$\textstyle\sum_{i=1}^{n} \lambda_i y_i = 0.$$

# Optimization (3)

- The Dual Problem

$$\max_{\lambda_1,\dots,\lambda_n \in \mathbb{R}} \quad L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j}.$$

$$\text{s.t.} \quad \lambda_i \geq 0 \ and \ \sum_{i=1}^{n} \lambda_i y_i = 0$$

  - We only need the inner product
- Recovering the Primal Solution
  - The strong duality holds
  - $\bar{W}$ can be recovered directly

$$\overline{W} = \sum_{i=1}^{n} \lambda_i y_i \overline{X_i}$$

If $\lambda_i \neq 0$, then $\bar{X}_i$ is a support vector

## Optimization (4)

- The Dual Problem

$$\max_{\lambda_1,\ldots,\lambda_n \in \mathbb{R}} \quad L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j}.$$

$$\text{s.t.} \quad \lambda_i \geq 0 \ and \ \textstyle\sum_{i=1}^{n} \lambda_i y_i = 0$$

  - We only need the inner product
- Recovering the Primal Solution
  - The strong duality holds
  - $\overline{W}$ can be recovered directly
  - We need the KKT conditions to recover $b$

$$\lambda_i \left[ y_i(\overline{W} \cdot \overline{X_i} + b) - 1 \right] = 0 \Rightarrow y_r \left[ \overline{W} \cdot \overline{X_r} + b \right] = +1 \quad \forall r : \lambda_r > 0$$

## Optimization (4)

- The Dual Problem

$$\max_{\lambda_1,\dots,\lambda_n\in\mathbb{R}} \quad L_D = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \overline{X_i}\cdot\overline{X_j}.$$

$$\text{s.t.} \quad \lambda_i \ge 0 \ and \ \sum_{i=1}^{n}\lambda_i y_i = 0$$

  - We only need the inner product
- Recovering the Primal Solution
  - The strong duality holds
  - $\bar{W}$ can be recovered directly
  - We need the KKT conditions to recover $b$

$$\lambda_i\left[y_i(\overline{W}\cdot\overline{X_i}+b)-1\right]=0 \Rightarrow y_r\left[\left(\sum_{i=1}^{n}\lambda_i y_i \overline{X_i}\cdot\overline{X_r}\right)+b\right]=+1 \ \forall r:\lambda_r>0$$

## Testing

- For a Test Instance $\bar{Z}$
  - The First Approach

$$F(\overline{Z}) = \text{sign}\{\overline{W} \cdot \overline{Z} + b\}$$

  - The Second Approach

$$F(\overline{Z}) = \text{sign}\{\overline{W} \cdot \overline{Z} + b\} = \text{sign}\{(\sum_{i=1}^{n} \lambda_i y_i \overline{X_i} \cdot \overline{Z}) + b\}$$

    - ✓ We only need the inner product
    - ✓ We only need to save $\lambda_i$ and $b$

## Solving the Dual Problem

- A Quadratic Optimization Problem

$$\max_{\lambda_1,\ldots,\lambda_n\in\mathbb{R}} \quad L_D = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \overline{X_i}\cdot\overline{X_j}.$$

$$\text{s.t.} \quad \lambda_i \geq 0 \ and \sum_{i=1}^{n}\lambda_i y_i = 0$$

  - Gradient Ascent

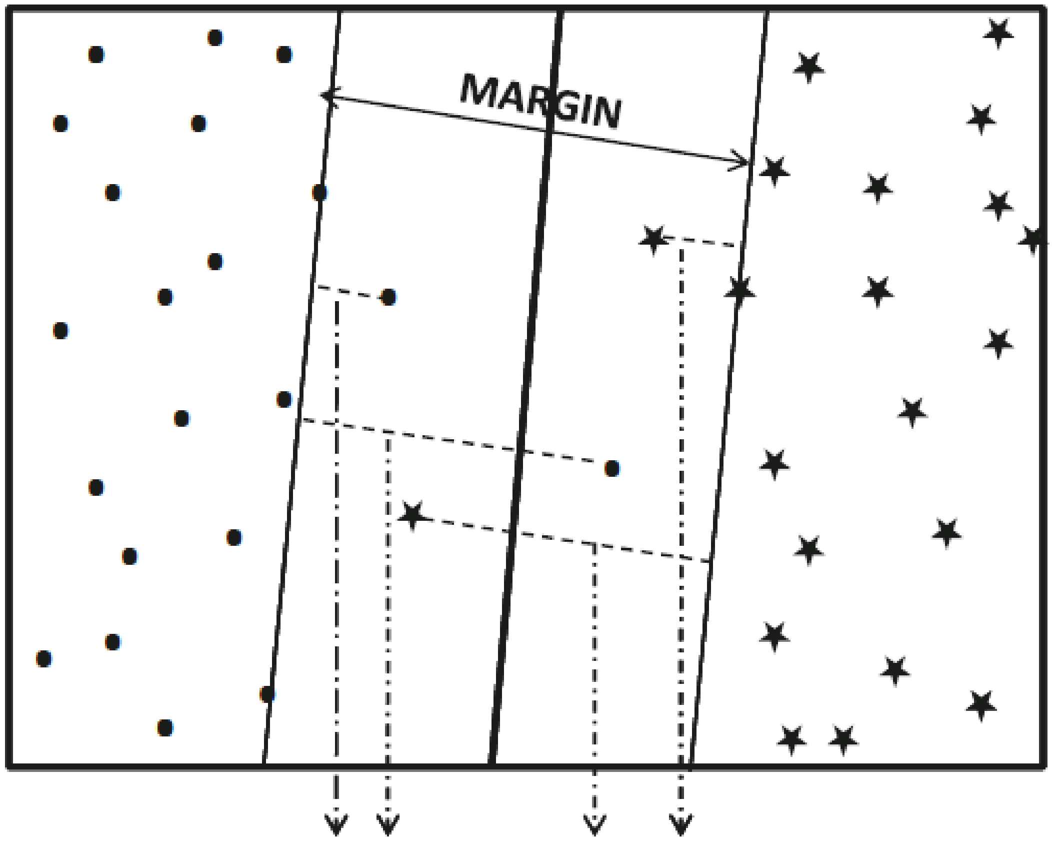
$$\frac{\partial L_D}{\partial \lambda_i} = 1 - y_i\sum_{j=1}^{n} y_j\lambda_j\overline{X_i}\cdot\overline{X_j}$$

$$(\lambda_1\ldots\lambda_n) \leftarrow (\lambda_1\ldots\lambda_n) + \alpha\left(\frac{\partial L_D}{\partial\lambda_1}\cdots\frac{\partial L_D}{\partial\lambda_n}\right)$$

  - The constraints may be violated
    - ✓ Projection before/after updating

## SVM with Soft Margin for Nonseparable Data (1)

- A Nonseparable Case



MARGIN VIOLATION WITH PENALTY-BASED SLACK VARIABLES

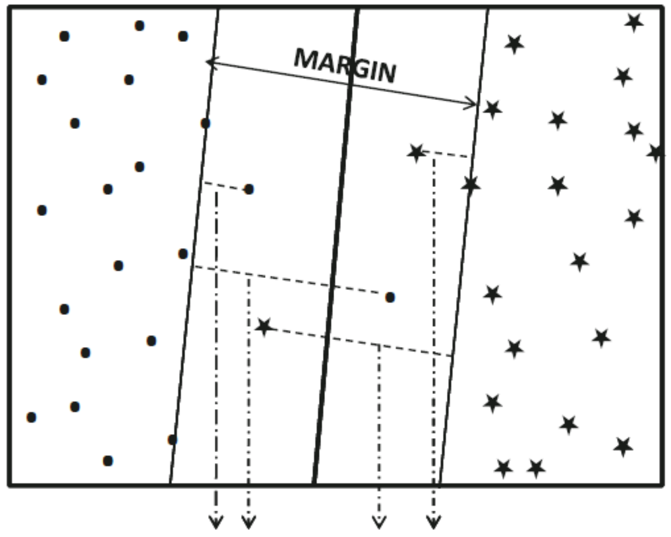
# SVM with Soft Margin for Nonseparable Data (2)

- Hard Margin Constraints

$$\overline{W} \cdot \overline{X_i} + b \geq +1 \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \leq -1 \quad \forall i : y_i = -1.$$

- Soft Margin Constraints

$$\overline{W} \cdot \overline{X_i} + b \geq +1 - \xi_i \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \leq -1 + \xi_i \quad \forall i : y_i = -1$$
$$\xi_i \geq 0 \quad \forall i.$$



MARGIN VIOLATION WITH PENALTY-BASED SLACK VARIABLES

- The Objective

$$O = \frac{||\overline{W}||^2}{2} + C \sum_{i=1}^{n} \xi_i.$$

## SVM with Soft Margin for Nonseparable Data (3)

- The Problem

$$\min_{\overline{W}\in\mathbb{R}^d,\xi_1,\ldots,\xi_n,b\in\mathbb{R}} \quad O = \frac{||\overline{W}||^2}{2} + C\sum_{i=1}^{n}\xi_i.$$

$$\text{s.t.} \quad \begin{aligned} \overline{W}\cdot\overline{X_i} + b &\geq +1 - \xi_i \quad \forall i : y_i = +1 \\ \overline{W}\cdot\overline{X_i} + b &\leq -1 + \xi_i \quad \forall i : y_i = -1 \\ \xi_i &\geq 0 \quad \forall i. \end{aligned}$$

- Lagrangian Relaxation

$$L_P = \frac{||\overline{W}||^2}{2} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i\left[y_i(\overline{W}\cdot\overline{X_i} + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i.$$

## SVM with Soft Margin for Nonseparable Data (4)

- A More Popular Formulation

$$\min_{\overline{W}\in\mathbb{R}^d, b\in\mathbb{R}} \quad O = \frac{||\overline{W}||^2}{2} + C\sum_{i=1}^{n} \max\{0, 1 - y_i[\overline{W}\cdot\overline{X_i} + b]\}.$$

  - Unconstrained but non-smooth
  - $\ell(z, y_i) = \max(0,1 - y_i z)$ is called hinge loss

# SVM with Soft Margin for Nonseparable Data (4)

- A More Popular Formulation

$$\min_{\overline{W}\in\mathbb{R}^d, b\in\mathbb{R}} \quad O = \frac{||\overline{W}||^2}{2} + C\sum_{i=1}^{n} \max\{0, 1 - y_i[\overline{W}\cdot\overline{X_i} + b]\}.$$

  - Unconstrained but non-smooth
  - $\ell(z, y_i) = \max(0,1 - y_i z)$ is called hinge loss

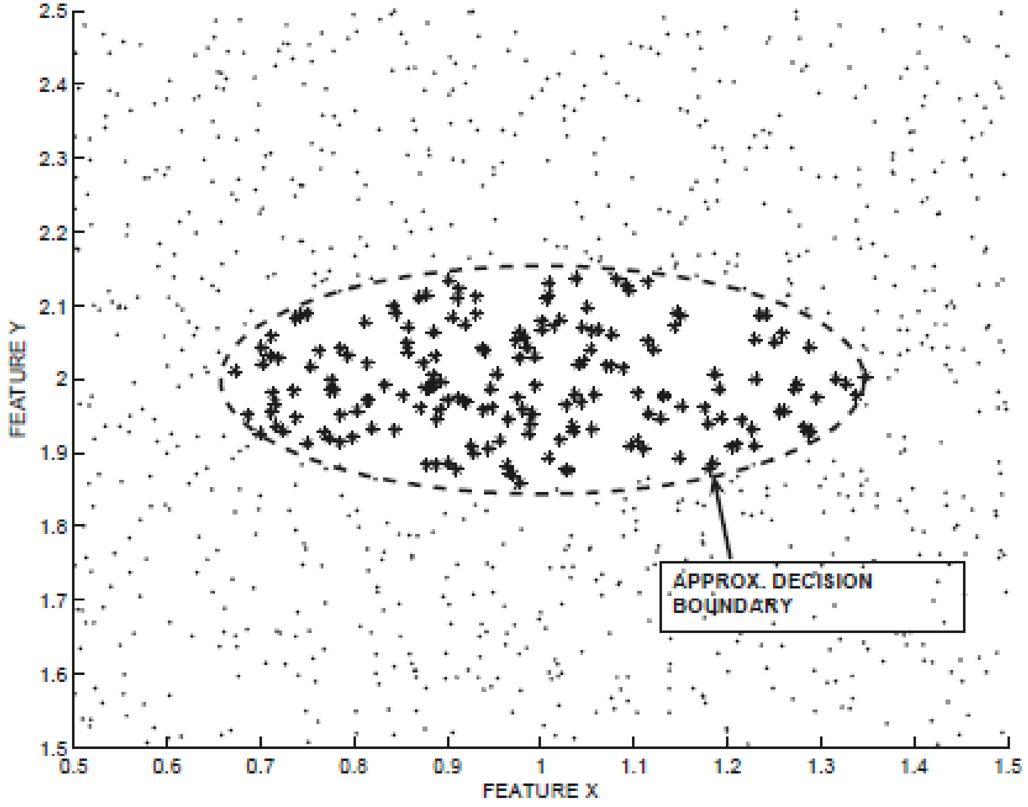
- Logistic Regression

$$\min_{\overline{W}\in\mathbb{R}^d, b\in\mathbb{R}} \quad O = \frac{\|\overline{W}\|^2}{2} + C\sum_{i=1}^{n} \log\left(1 + e^{-y_i[\overline{W}\cdot\overline{X_i}+b]}\right)$$

  - Unconstrained and smooth
  - $\ell(z, y_i) = \log(1 + e^{-y_i z})$ is called logit loss

## Nonlinear SVM

- An Example



## The Kernel Trick (1)

- Replace inner product with kernel functions
  - A Mapping: $\bar{X} \rightarrow \Phi(\bar{X})$
  - Kernel function

$$K(\overline{X_i}, \overline{X_j}) = \Phi(\overline{X_i}) \cdot \Phi(\overline{X_j})$$

- Training

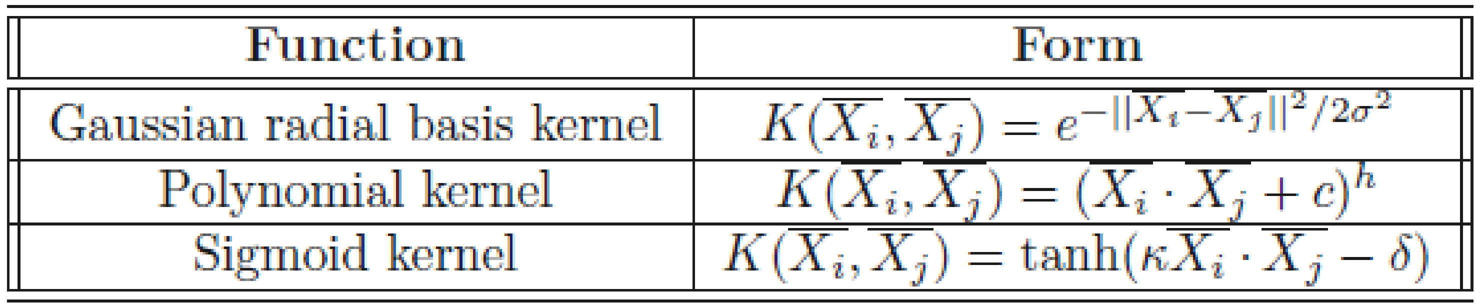
$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j K(\overline{X_i}, \overline{X_j})$$

- Testing

$$F(\overline{Z}) = \text{sign}\{(\sum_{i=1}^{n} \lambda_i y_i K(\overline{X_i}, \overline{Z})) + b\}$$

## The Kernel Trick (2)

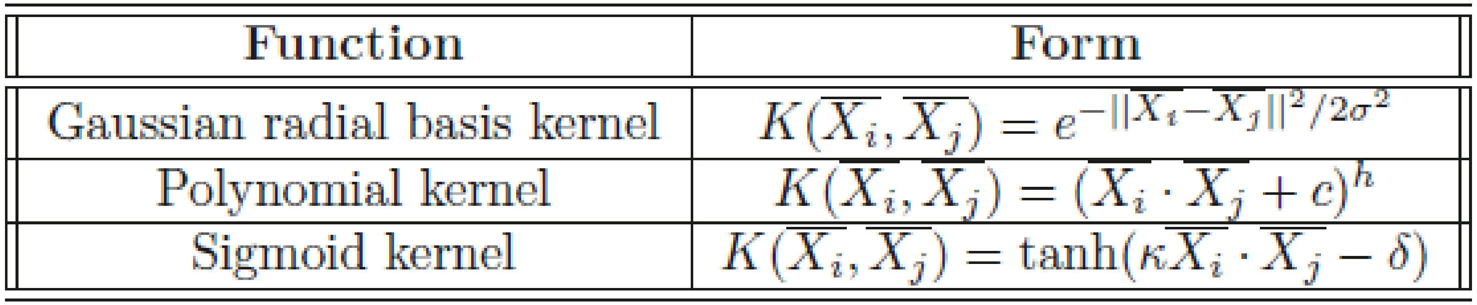
- Kernel Functions



| Function | Form |
|---|---|
| Gaussian radial basis kernel | $K(\overline{X_i}, \overline{X_j}) = e^{-\|\overline{X_i}-\overline{X_j}\|^2/2\sigma^2}$ |
| Polynomial kernel | $K(\overline{X_i}, \overline{X_j}) = (\overline{X_i} \cdot \overline{X_j} + c)^h$ |
| Sigmoid kernel | $K(\overline{X_i}, \overline{X_j}) = \tanh(\kappa\overline{X_i} \cdot \overline{X_j} - \delta)$ |

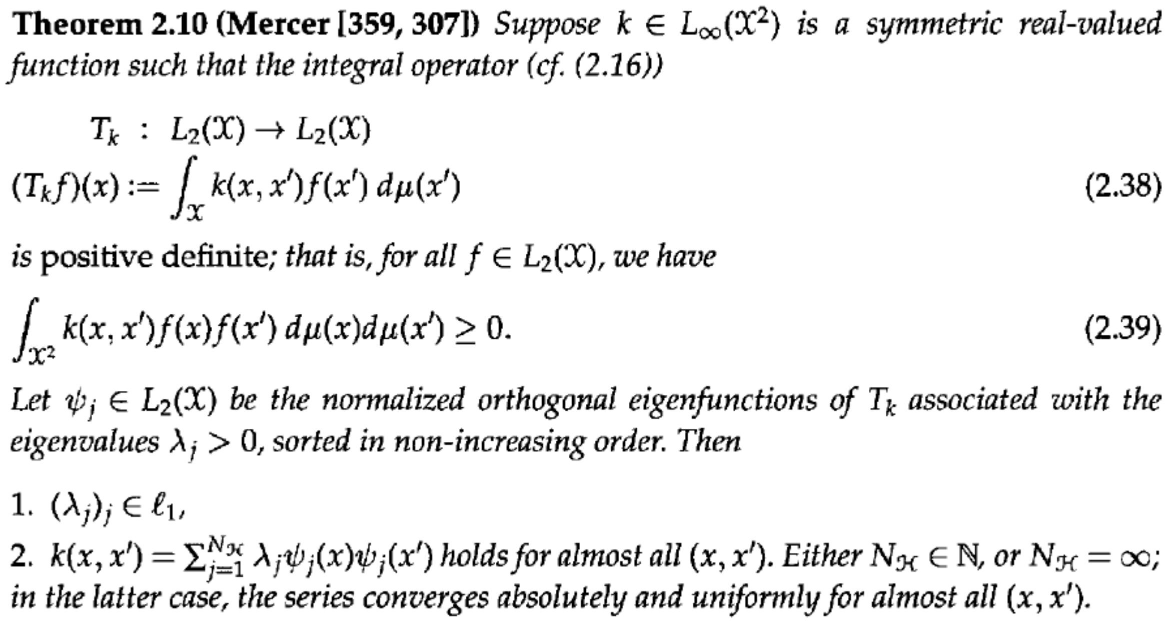
## The Kernel Trick (2)

### Kernel Functions



| Function | Form |
|---|---|
| Gaussian radial basis kernel | $K(\overline{X_i}, \overline{X_j}) = e^{-\|\overline{X_i}-\overline{X_j}\|^2/2\sigma^2}$ |
| Polynomial kernel | $K(\overline{X_i}, \overline{X_j}) = (\overline{X_i} \cdot \overline{X_j} + c)^h$ |
| Sigmoid kernel | $K(\overline{X_i}, \overline{X_j}) = \tanh(\kappa \overline{X_i} \cdot \overline{X_j} - \delta)$ |

### Mercer's Theorem



**Theorem 2.10 (Mercer [359, 307])** *Suppose $k \in L_\infty(\mathcal{X}^2)$ is a symmetric real-valued function such that the integral operator (cf. (2.16))*

$$T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$$

$$(T_k f)(x) := \int_{\mathcal{X}} k(x,x') f(x')\, d\mu(x') \qquad (2.38)$$

*is* positive definite*; that is, for all $f \in L_2(\mathcal{X})$, we have*

$$\int_{\mathcal{X}^2} k(x,x') f(x) f(x')\, d\mu(x) d\mu(x') \geq 0. \qquad (2.39)$$

*Let $\psi_j \in L_2(\mathcal{X})$ be the normalized orthogonal eigenfunctions of $T_k$ associated with the eigenvalues $\lambda_j > 0$, sorted in non-increasing order. Then*

1. $(\lambda_j)_j \in \ell_1$,
2. $k(x,x') = \sum_{j=1}^{N_{\mathcal{H}}} \lambda_j \psi_j(x) \psi_j(x')$ *holds for almost all $(x,x')$. Either $N_{\mathcal{H}} \in \mathbb{N}$, or $N_{\mathcal{H}} = \infty$; in the latter case, the series converges absolutely and uniformly for almost all $(x,x')$.*

Scholkopf and . Smola
Learning with Kernels
The MIT Press, 2011.

## Outline

- Support Vector Machines
- **Neural Networks**
- Instance-Based Learning
- Classifier Evaluation
- Summary

# Neural Networks

- Neural networks are a model of simulation of the human nervous system

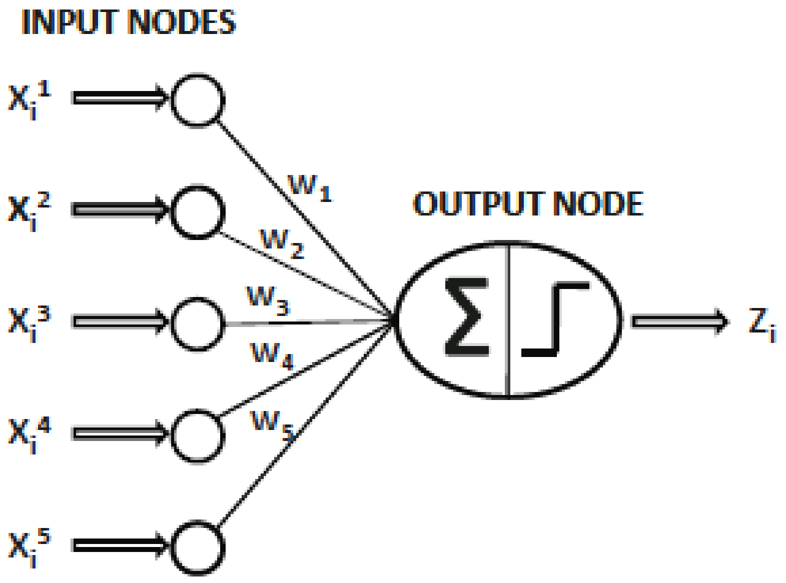
- The human nervous system is composed of cells, referred to as neurons
- Biological neurons are connected to one another at contact points, which are referred to as synapses
- Learning is performed in living organisms by changing the strength of synaptic connections between neurons
  - Typically, the strength of these connections change in response to external stimuli

## Artificial Neural Networks

- The individual nodes in artificial neural networks are referred to as neurons
- The computation function at a neuron is defined by the weights on the input connections to that neuron
  - This weight can be viewed as analogous to the strength of a synaptic connection
- The “external stimulus” in artificial neural networks for learning these weights is provided by the training data

## Single-Layer Neural Network: The Perceptron

- Architecture
  - Input nodes
    - ✓ One for each feature
    - ✓ No computation
  - A output node
    - ✓ Activation function



(a) Perceptron

$$
\begin{aligned}
z_i &= \operatorname{sign}\{\sum_{j=1}^{d} w_j x_i^j + b\} \\
&= \operatorname{sign}\{\overline{W} \cdot \overline{X_i} + b\}
\end{aligned}
$$

## Training of Perceptron

- Prediction Error $(z_i - y_i)$
  - $-2, 0, 2$

## Training of Perceptron

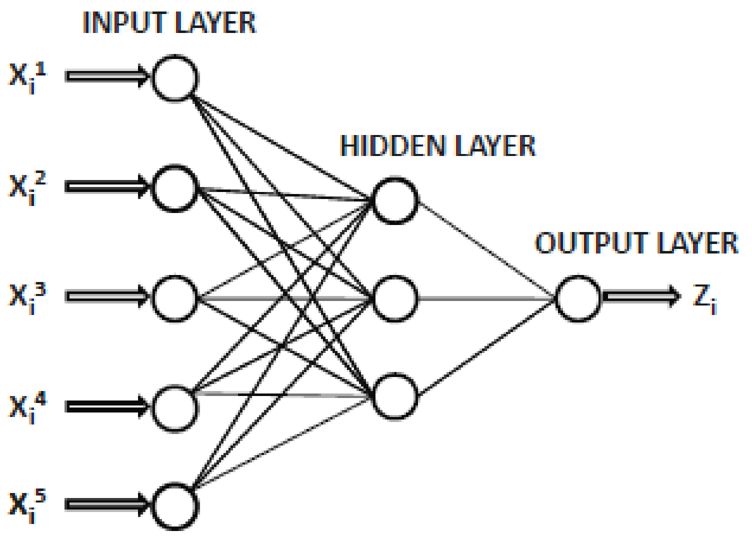
- Prediction Error $(z_i - y_i)$
  - $-2, 0, 2$
- Algorithm
  - Start with a random vector
  - Feed $\bar{X}_i$ into the neural network one by one

$$\overline{W}^{t+1} = \overline{W}^{t} + \eta(y_i - z_i)\overline{X}_i.$$

    - ✓ $\eta$ is a step size or learning rate
    - ✓ $(y_i - z_i) \in \{-2,0,2\}$
    - ✓ An approximation of gradient descent for square loss $(y_i - z_i)^2 = \left(y_i - \mathrm{sign}(\bar{W} \cdot \bar{X}_i - b)\right)^2$

# Multilayer Neural Networks

- Architecture
  - Input Layer
    - ✓ One node for each feature
    - ✓ No computation
  - Hidden Layer
    - ✓ Maybe multiple layers
  - Output layer



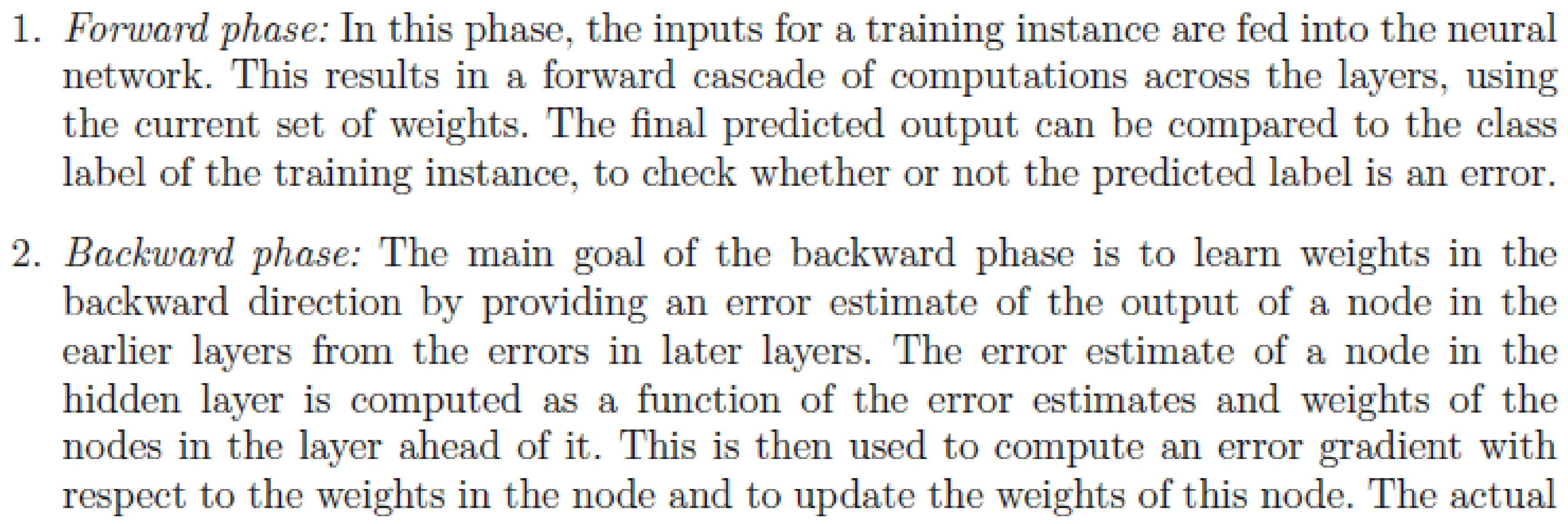
(b) Multilayer

  - Functions at hidden and output layers

$$z_i = \sum_{j=1}^{d} w_j \frac{1}{1+e^{-x_i^j}} + b.$$

## Training

- The Challenge
  - The ground-truth of hidden layer nodes are unknown
- Backpropagation


1. *Forward phase:* In this phase, the inputs for a training instance are fed into the neural network. This results in a forward cascade of computations across the layers, using the current set of weights. The final predicted output can be compared to the class label of the training instance, to check whether or not the predicted label is an error.

2. *Backward phase:* The main goal of the backward phase is to learn weights in the backward direction by providing an error estimate of the output of a node in the earlier layers from the errors in later layers. The error estimate of a node in the hidden layer is computed as a function of the error estimates and weights of the nodes in the layer ahead of it. This is then used to compute an error gradient with respect to the weights in the node and to update the weights of this node. The actual

## Discussions

- A multilayer neural network is more powerful than a kernel SVM
  - Capture decision boundaries of arbitrary shapes
  - Capture noncontiguous class distributions with different decision boundaries in different regions of the data
- Challenges
  - Design of the topology of the network
    - ✓ Overfitting, Deep learning
  - Convergence rate is slow or unclear

# Outline

- Support Vector Machines
- Neural Networks
- **Instance-Based Learning**
- Classifier Evaluation
- Summary

## Instance based learning

- Eager Learner
  - The classification model is constructed *up front* and then used to classify a specific test instance
  - SVM, Neural Networks
- Lazy Learner
  - The training is *delayed* until the last step of classification
  - Instance based learning
    - ✓ Similar instances have similar class labels

## Nearest-neighbor Classifiers

- Given a Test Instance
  - Determine the closest $m$ training examples
  - Use the dominant label among these $m$ training examples
    - ✓ Weighted voting

$$f(\delta) = e^{-\delta^2/t^2}$$

- Challenges
  - Decide the value of $m$
  - Measure the distance

## Design Variations of Nearest Neighbor Classifiers

- The Standard Approach
  - The Euclidean function
  - Cannot reflect the distribution of data

- A More General Formulation

$$Dist(\overline{X},\overline{Y}) = \sqrt{(\overline{X}-\overline{Y})A(\overline{X}-\overline{Y})^T}$$

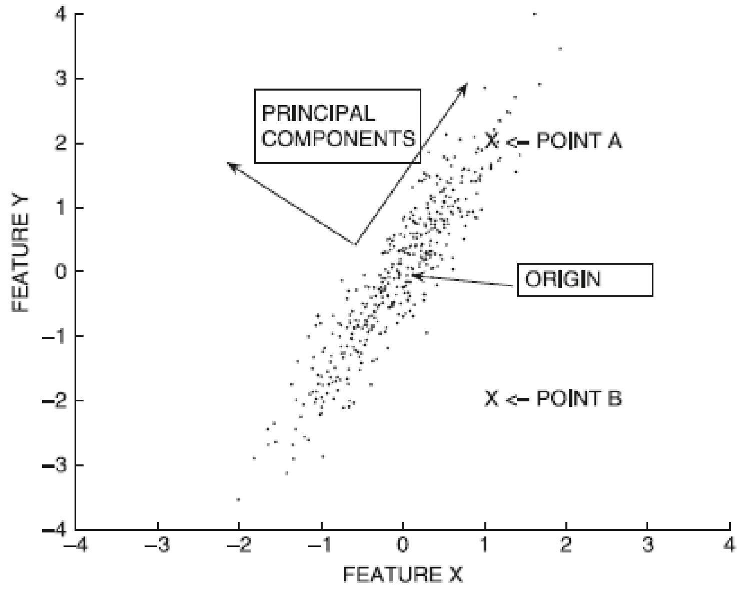
  - $A$ is a positive semidefinite (PSD) matrix

## Unsupervised Mahalanobis Metric

- The Mahalanobis Distance

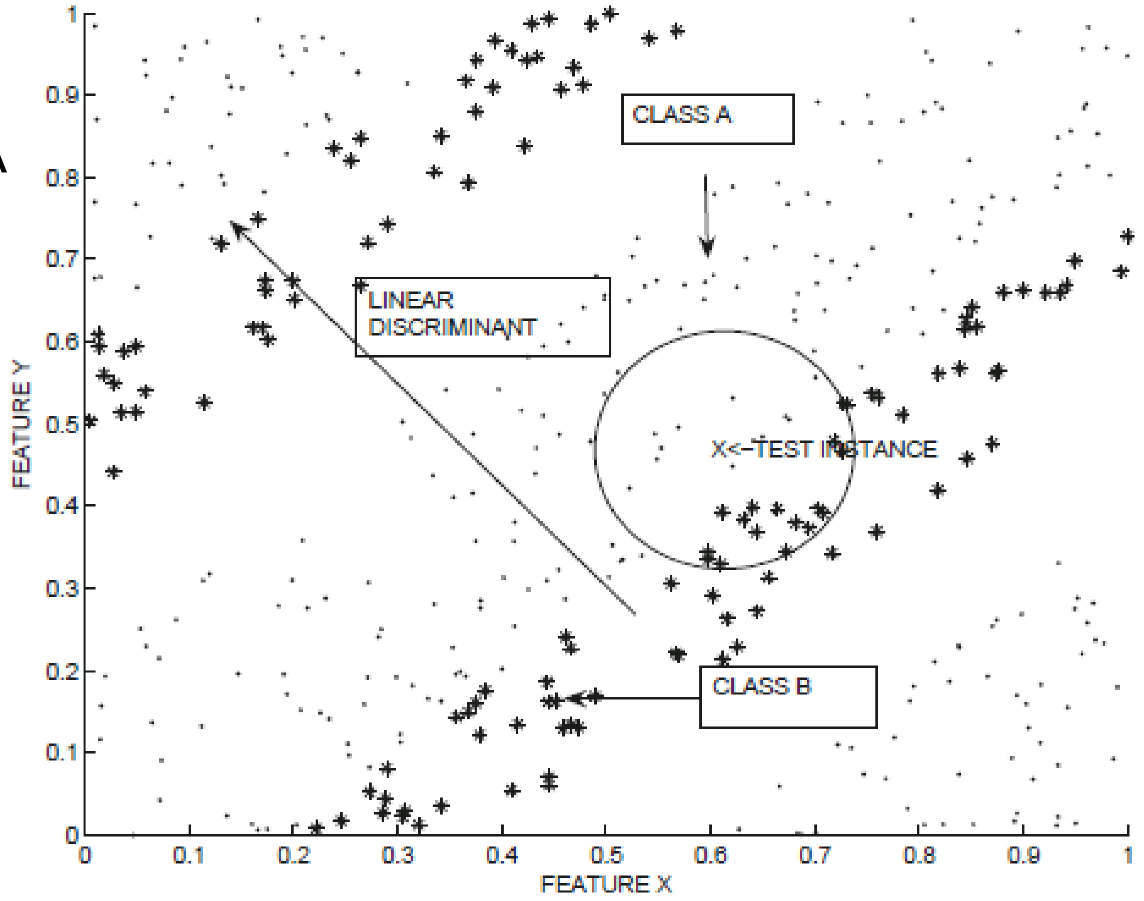
$$Dist(\overline{X},\overline{Y}) = \sqrt{(\overline{X}-\overline{Y})\Sigma^{-1}(\overline{X}-\overline{Y})^T}.$$

  - Σ is the covariance matrix



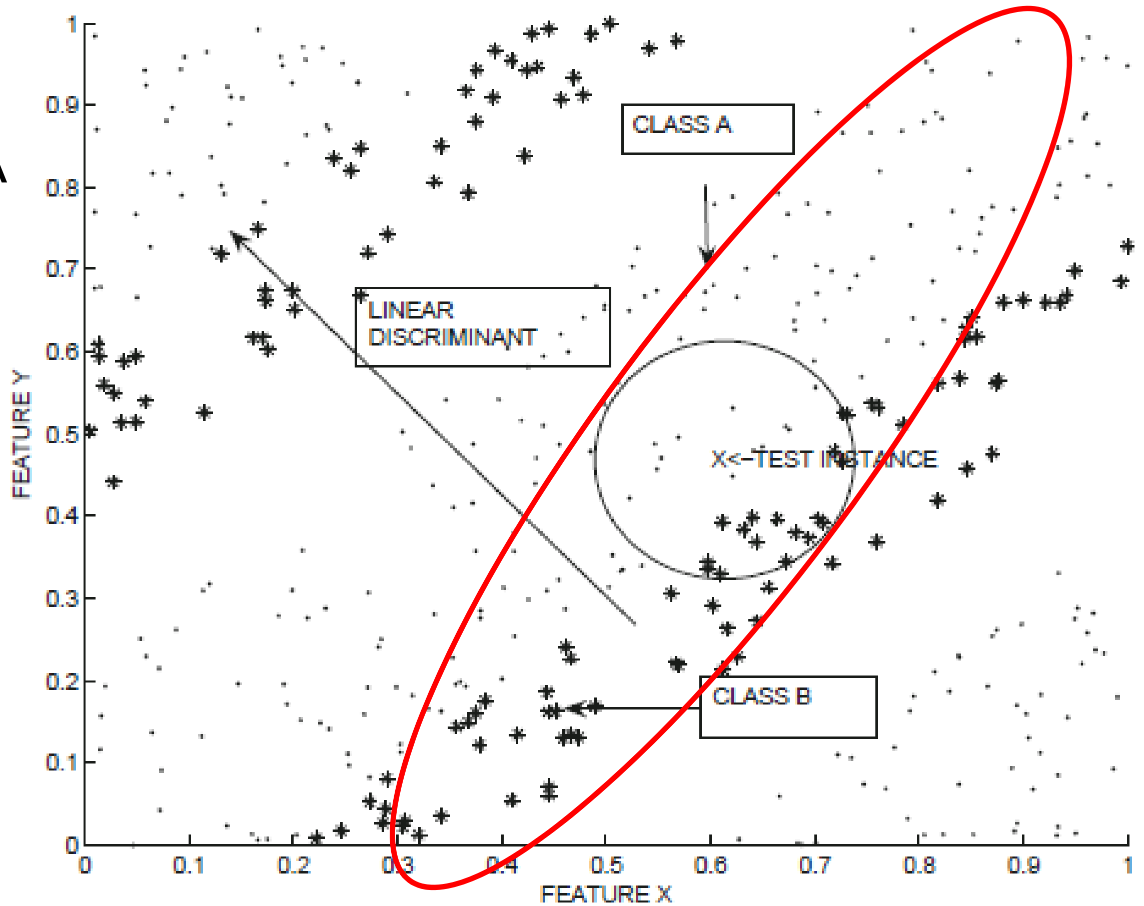
## Nearest Neighbors with Linear Discriminant Analysis (1)

- An Example
  - The circle include more points from class B than class A



## Nearest Neighbors with Linear Discriminant Analysis (2)

- An Example
  - The circle include more points from class B than class A
  - "Elongate" the neighborhoods along the less discriminative directions
  - "Shrink" the neighborhoods along the more discriminative directions



# Nearest Neighbors with Linear Discriminant Analysis (3)

- The Procedure (LDA)
  - $\mathcal{D}$ is the full data set
  - $\bar{\mu}$ is the mean of $\mathcal{D}$
  - $\mathcal{D}_i$ is the set of data belonging to class $i$
  - $p_i = |\mathcal{D}_i|/|\mathcal{D}|$ is the fraction of data in class $i$
  - $\mu_i$ is the mean of $\mathcal{D}_i$
  - $\Sigma_i$ is the covariance matrix of $\mathcal{D}_i$

$$S_w = \sum_{i=1}^{k} p_i \Sigma_i . \qquad S_b = \sum_{i=1}^{k} p_i (\overline{\mu_i} - \overline{\mu})^T (\overline{\mu_i} - \overline{\mu}).$$

$$A = S_w^{-1} S_b S_w^{-1}.$$

## Outline

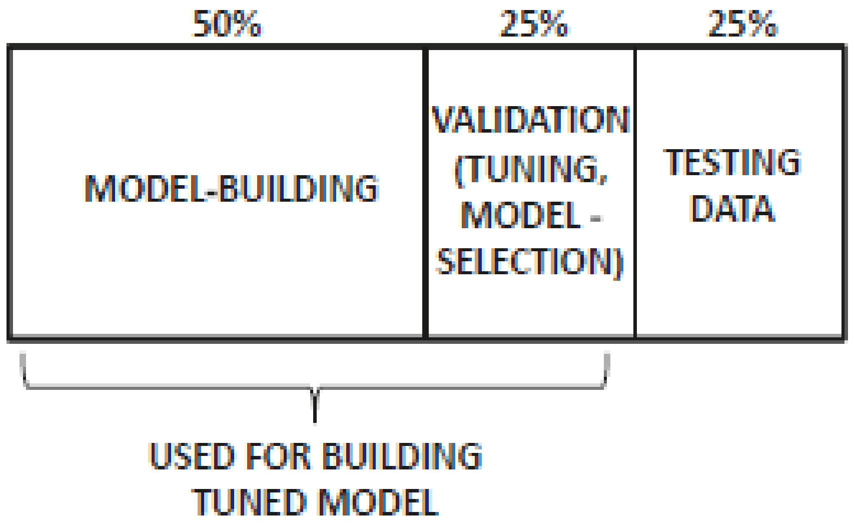
- Support Vector Machines
- Neural Networks
- Instance-Based Learning
- **Classifier Evaluation**
- Summary

## Classifier Evaluation

- Methodological issues
  - Dividing the labeled data appropriately into training and test segments for evaluation
- Quantification issues
  - Providing a numerical measure for the quality of the method after a specific methodology for evaluation has been selected

## Methodological issues

- Training
  - Model-Building
  - Validation
    - ✓ For parameter tuning or mode selection
- Testing
  - Measure the performance


50%
25%
25%
MODEL-BUILDING
VALIDATION (TUNING, MODEL - SELECTION)
TESTING DATA
USED FOR BUILDING TUNED MODEL

## Holdout

- Randomly divided into two disjoint sets
  - A majority is used as the training data
  - Remaining is used as the test data
- Repeating the process over $b$ different holdout samples

- When the classes are imbalanced
  - Implement the holdout method by independently sampling the two classes at the same level (Stratified sampling)

## Cross-Validation

- Data is divided into $m$ disjoint subsets of equal size $n/m$
- One of the $m$ segments is used for testing, and the other $(m-1)$ segments are used for training

- Leave-one-out cross-validation $m = n$

- Repeating the process over $b$ different random $m$-way partitions of the data

## Bootstrap

- The labeled data is *sampled uniformly with replacement*, to create a training data set
  - possibly contain duplicates
- The probability that the data point is not included in $n$ samples

$$\left(1-\frac{1}{n}\right)^n \approx \frac{1}{e}$$

- The fraction of the labeled data points included at least once

$$1-\frac{1}{e} \approx 0.632$$

## Quantification Issues (1)

- Output as Class Labels
  - Accuracy: fraction of test instances in which the predicted value is right
  - Cost-sensitive accuracy
    - ✓ $c_1 \ldots c_k$ be cost of misclassification of each class
    - ✓ $n_1 \ldots n_k$ be the number of test instances belong to each class
    - ✓ $a_1 \ldots a_k$ be the accuracy for each class

$$A = \frac{\sum_{i=1}^{k} c_i n_i a_i}{\sum_{i=1}^{k} c_i n_i}$$

- Significant Test

## Quantification Issues (2)

- Output as Numerical Score
  - The output of the classification algorithm is a numerical score associated with each test instance and label value.
  - Provide more flexibility in evaluating the overall trade-off
  - Similar to outlier validity

## Two Classes

- For any threshold $t$ on the predicted positive-class score
  - $S(t)$ is the declared positive class set
  - $\mathcal{G}$ is the true set of positive instances
  - The Precision

$$Precision(t) = 100 * \frac{|\mathcal{S}(t) \cap \mathcal{G}|}{|\mathcal{S}(t)|}$$

  - The Recall

$$Recall(t) = 100 * \frac{|\mathcal{S}(t) \cap \mathcal{G}|}{|\mathcal{G}|}$$

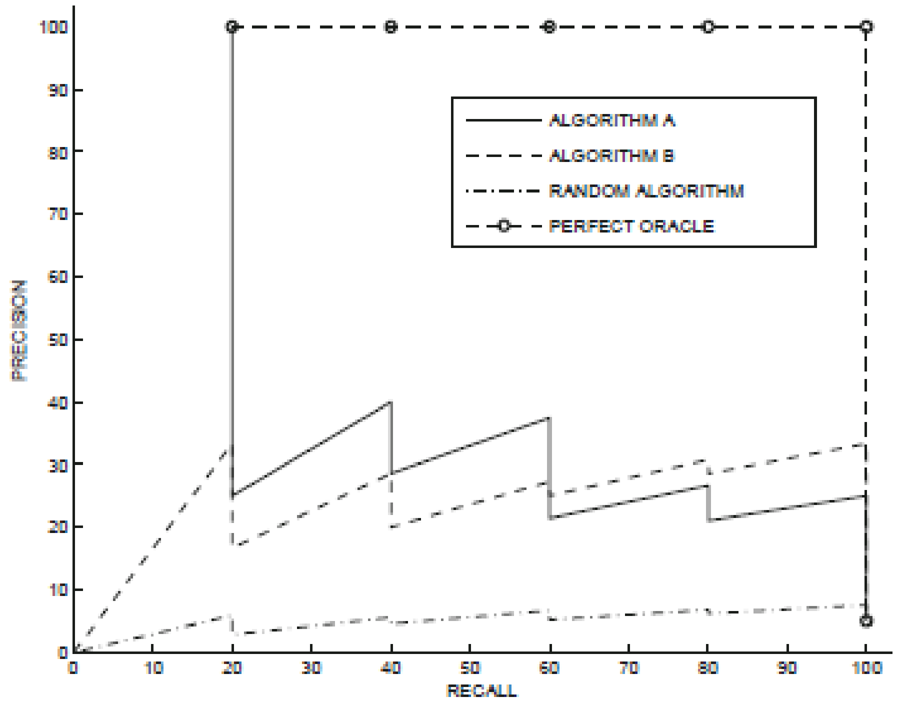
  - $F_1$-measure

$$F_1(t) = \frac{2 \cdot Precision(t) \cdot Recall(t)}{Precision(t) + Recall(t)}$$

# Precision-Recall Curve

Table 10.2: Rank of ground-truth positive instances

| Algorithm | Rank of positive class instances |
|---|---|
| Algorithm A | 1, 5, 8, 15, 20 |
| Algorithm B | 3, 7, 11, 13, 15 |
| Random Algorithm | 17, 36, 45, 59, 66 |
| Perfect Oracle | 1, 2, 3, 4, 5 |



## ROC curve (1)

- True-positive rate (recall)

$$TPR(t) = Recall(t) = 100 * \frac{|\mathcal{S}(t) \cap \mathcal{G}|}{|\mathcal{G}|}$$

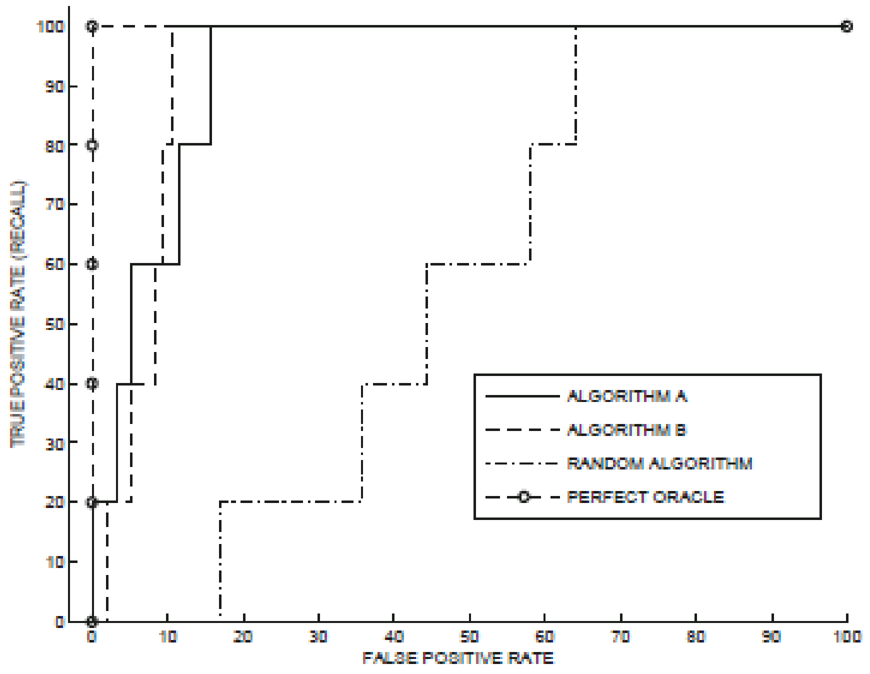
- False-positive rate

$$FPR(t) = 100 * \frac{|\mathcal{S}(t) - \mathcal{G}|}{|\mathcal{D} - \mathcal{G}|}$$

## ROC curve (2)

Table 10.2: Rank of ground-truth positive instances

| Algorithm | Rank of positive class instances |
|---|---|
| Algorithm A | 1, 5, 8, 15, 20 |
| Algorithm B | 3, 7, 11, 13, 15 |
| Random Algorithm | 17, 36, 45, 59, 66 |
| Perfect Oracle | 1, 2, 3, 4, 5 |



## Outline

- Support Vector Machines
- Neural Networks
- Instance-Based Learning
- Classifier Evaluation
- **Summary**

## Summary

- Support Vector Machines
  - Linearly Separable, Nonseparable
  - Dual, Kernel Trick
- Neural Networks
  - Single-Layer, Multilayer
- Instance-Based Learning
  - Nearest-neighbor classifiers
- Classifier Evaluation
  - Holdout, Cross-Validation, Bootstrap
  - Accuracy, precision–recall, ROC curve