Machine Learning
ITCS 6156/8156

Support Vector Machines

Razvan C. Bunescu

Department of Computer Science @ CCI

razvan.bunescu@uncc.edu

Max-Margin Classifiers: Separable Case

- Linear model for binary classification:

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

- Training examples:

$$(\mathbf{x}_1, t_1),\ (\mathbf{x}_2, t_2),\ \ldots\ (\mathbf{x}_N, t_N), \text{ where } t_n \in \{+1, -1\}$$

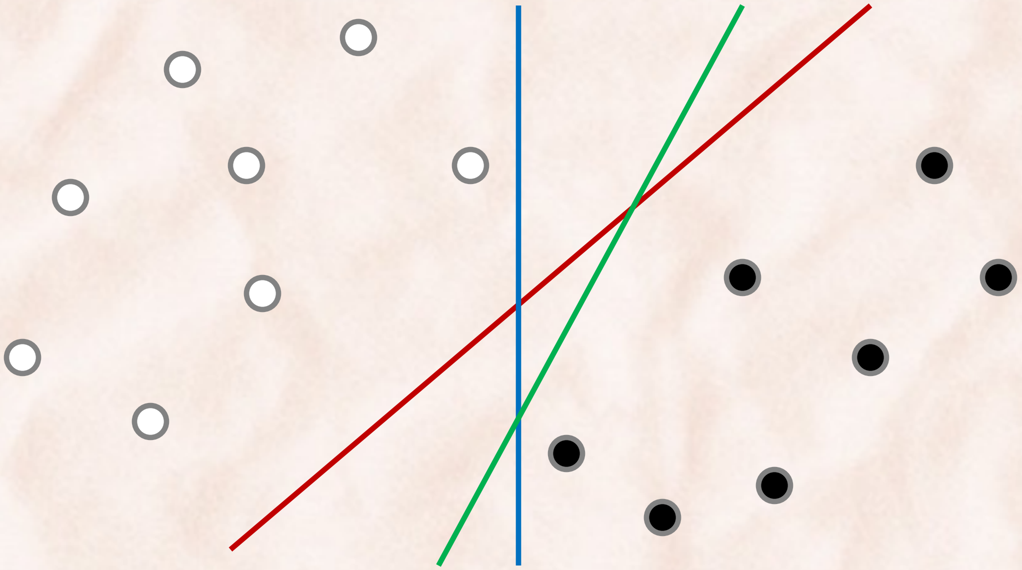
- Assume training data is linearly separable:

$$t_n y(x_n) > 0, \text{ for all } 1 \le n \le N$$

$\Rightarrow$ perceptron solution depends on:

- initial values of **w** and b.
- order of processing of data points.

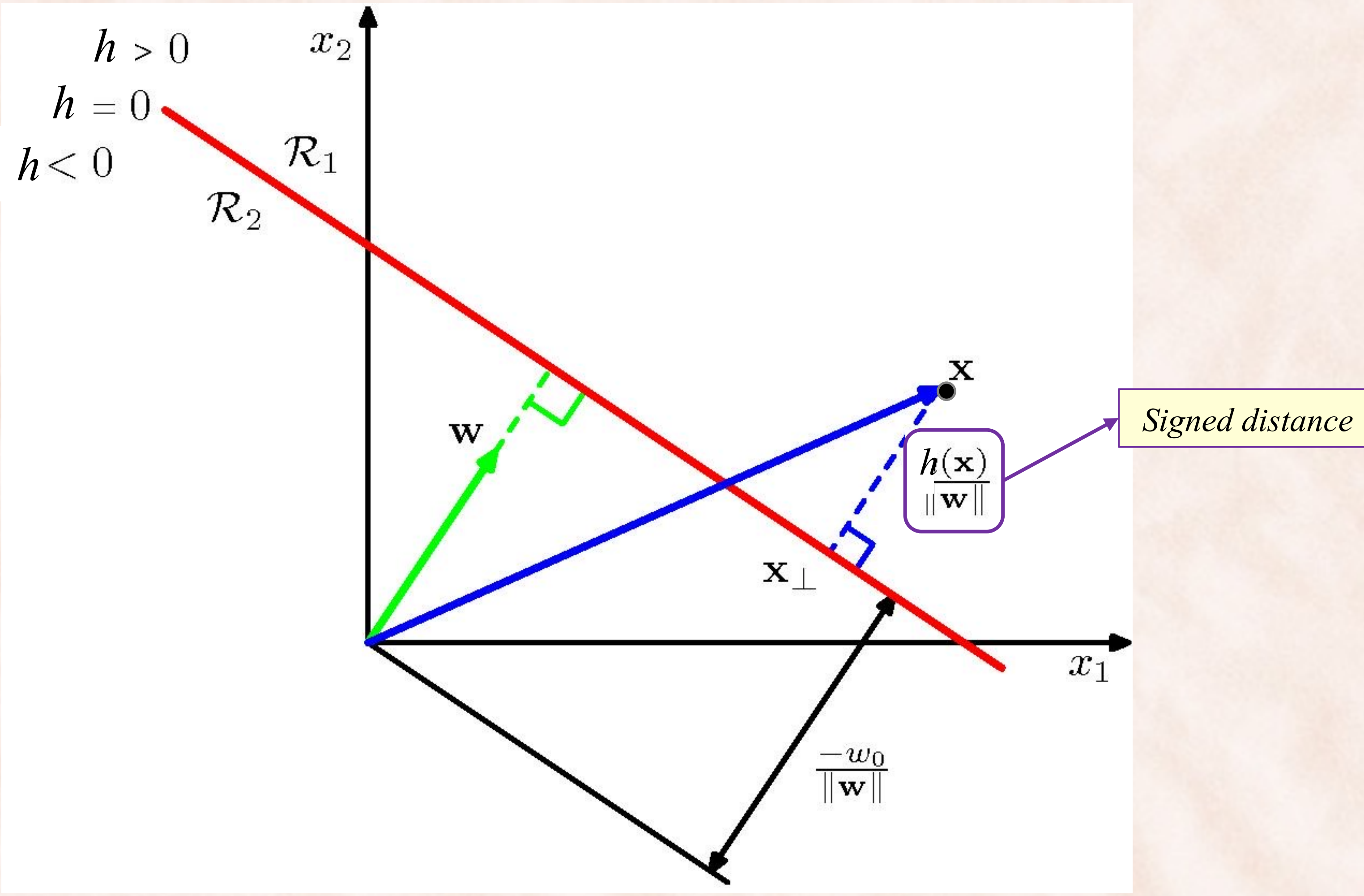
Maximum Margin Classifiers



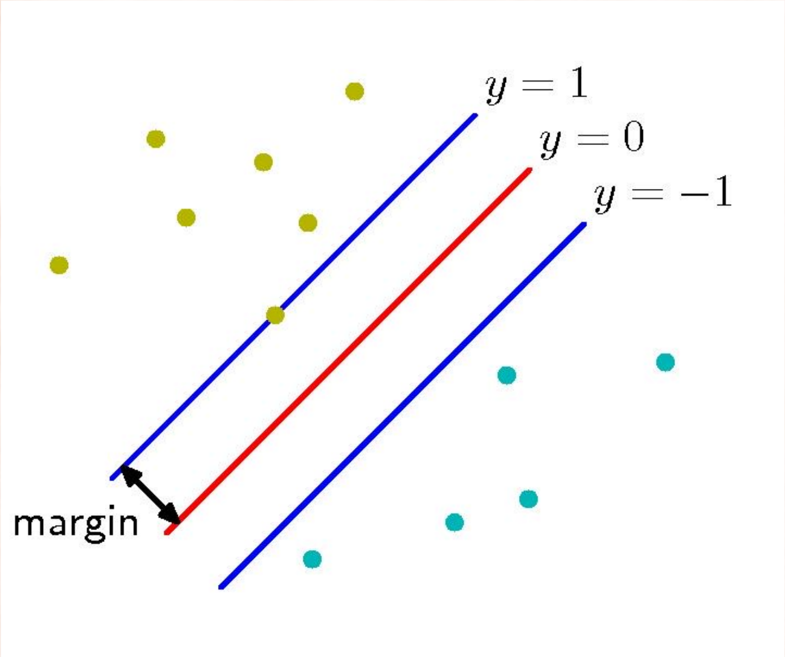
- Which hyperplane has the smallest generalization error?
  - The one that maximizes the margin [Computational Learning Theory]
    - margin = the distance between the decision boundary and the closest sample.

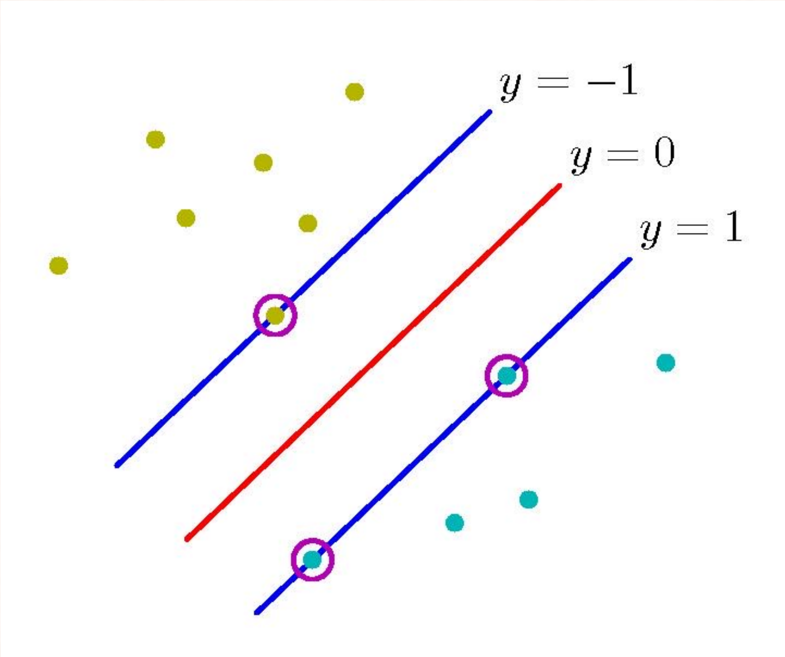
Geometric Interpretation

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$



Maximum Margin Classifiers





- The distance between a point $\mathbf{x}_n$ and a hyperplane $y(\mathbf{x}) = 0$ is:

$$\frac{|y(\mathbf{x}_n)|}{\|\mathbf{w}\|} = \frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

Maximum Margin Classifiers

- **Margin** = the distance between hyperplane $y(\mathbf{x}) = 0$ and closest sample:

$$\min_n \left[\frac{t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \right]$$

- Find parameters **w** and *b* that **maximize the margin**:

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

- Rescaling **w** and *b* does not change distances to the hyperplane:

$\Rightarrow$ for the closest point(s), set $t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) = 1$

$\Rightarrow$ this means $t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1, \quad \forall n \in \{1, \ldots, N\}$

Max-Margin: Quadratic Optimization

- Constrained optimization problem:

minimize:

$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1, \quad \forall n \in \{1, \ldots, N\}$$

- Solved using the technique of Lagrange Multipliers.
  - [*derivation shown at the end of slides, mandatory for 8156*].

Max-Margin: Quadratic Optimization

- Equivalent **dual representation**:

maximize:

$$L_D(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to:

$$\alpha_n \geq 0, \quad n = 1, \ldots, N$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

  - $k(\mathbf{x}_n, \mathbf{x}_m) = \varphi(\mathbf{x}_n)^T \varphi(\mathbf{x}_n)$ is the *kernel* function.
  - where $\boxed{\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \varphi(x_n)}$ and $\sum_{n=1}^{N} \alpha_n t_n = 0$

Exactly like in the Kernel Perceptron!

KKT conditions

1. primal constraints: $t_n y(x_n) - 1 \geq 0$

1. dual constraints: $\alpha_n \geq 0$

2. complementary slackness: $\alpha_n \{ t_n y(x_n) - 1 \} = 0$

$\Rightarrow$ for any data point, either $\alpha_n = 0$ or $t_n y(x_n) = 1$

S = $\{n \mid t_n y(x_n) = 1\}$ is the set of *support vectors*

Max-Margin Solution

- After solving the dual problem $\Rightarrow$ know α_n, for $n = 1\ldots N$

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \varphi(x_n) = \sum_{m \in S} \alpha_m t_m \varphi(x_m)$$

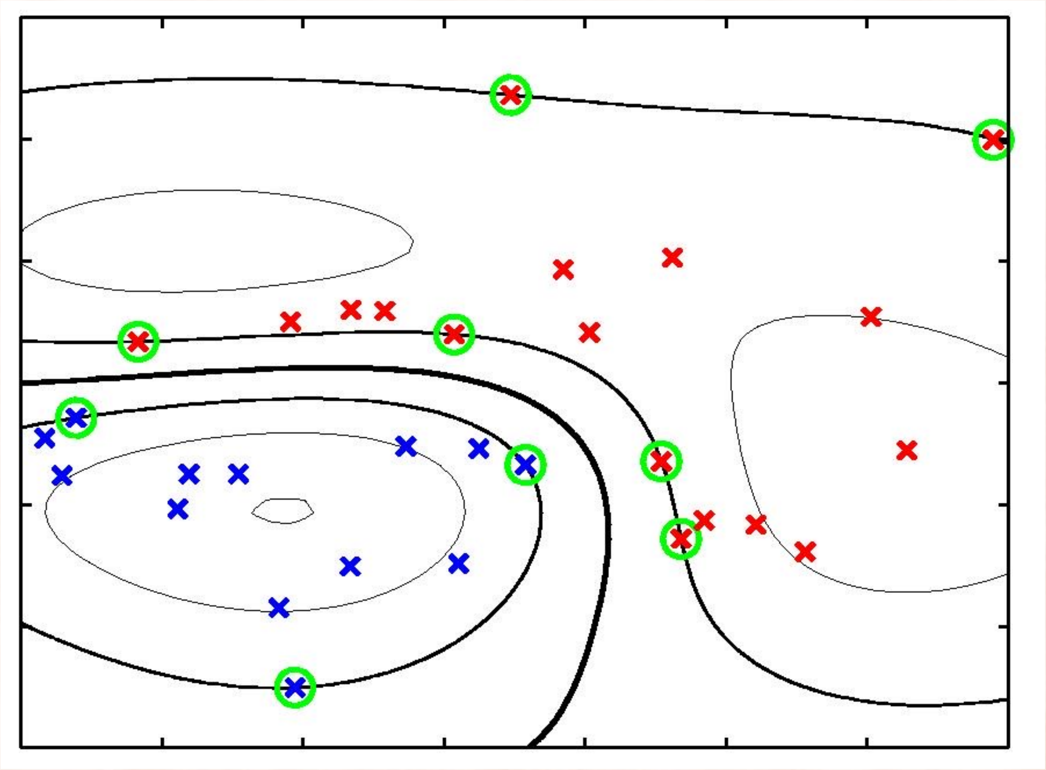
$$b = \frac{1}{|S|} \sum_{n \in S} \left(t_n - \sum_{m \in S} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

- Linear discriminant function becomes:

$$y(x) = \sum_{m \in S} \alpha_m t_m k(x, x_m) + b$$

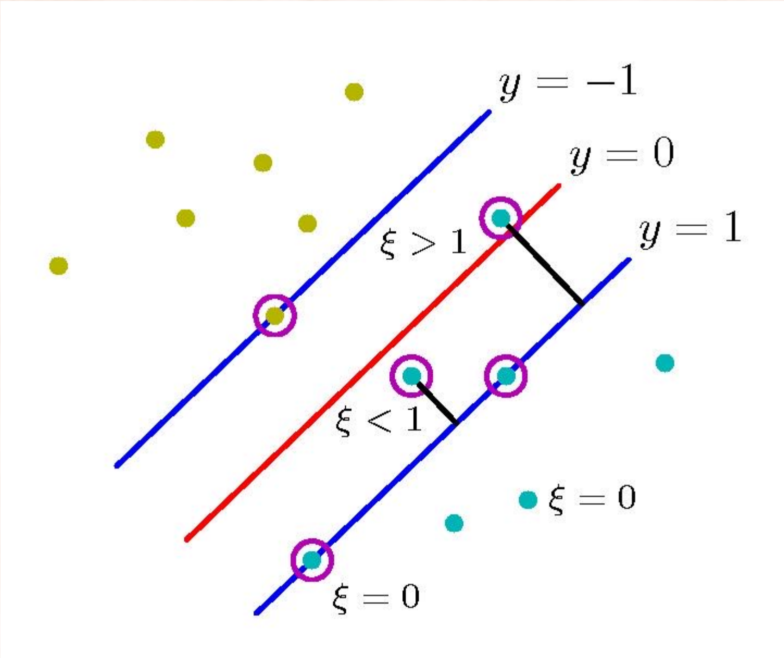
$\Rightarrow$ In both training and testing, examples are used only through the *kernel function*!

An SVM with Gaussian kernel



Max-Margin Classifiers: Non-Separable Case

- Allow data points to be on the wrong side of the margin boundary.
  - Penalty that increases with the distance from the boundary.



Max-Margin: Quadratic Optimization

- Optimization problem:

minimize:

$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n$$

subject to:

$$t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1 - \xi_n, \quad \forall n \in \{1, \ldots, N\}$$

$$\xi_n \geq 0$$

- Solve it using the technique of Lagrange Multipliers.

Max-Margin: Quadratic Optimization

- Dual representation:

maximize:

$$L_D(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to:

$$0 \le \alpha_n \le C, \quad n = 1, \ldots, N$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

- $k(\mathbf{x}_n,\mathbf{x}_m)=\varphi(\mathbf{x}_n)^{T}\varphi(\mathbf{x}_n)$ is the *kernel* function.

(Some of the) KKT conditions

1. primal constraints: $t_n y(x_n) - 1 + \xi_n \geq 0$

1. dual constraints: $0 \leq \alpha_n \leq C$

2. complementary slackness: $\alpha_n \{ t_n y(x_n) - 1 + \xi_n \} = 0$

$\Rightarrow$ for any data point, either $\alpha_n = 0$ or $t_n y(x_n) = 1 - \xi_n$

$S = \{n \mid t_n y(x_n) = 1 - \xi_n\}$ is the set of *support vectors* $(\alpha_n > 0)$

$M = \{n \mid 0 < \alpha_n < \mathrm{C}\}$ is the set of SVs that lie on the margin.

Max-Margin Solution

- After solving the dual problem $\Rightarrow$ know α_n, for $n = 1\ldots N$

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \varphi(x_n) = \sum_{m \in S} \alpha_m t_m \varphi(x_m)$$

$$b = \frac{1}{|M|} \sum_{n \in M} \left(t_n - \sum_{m \in S} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

- Linear discriminant function becomes:

$$y(x) = \sum_{m \in S} \alpha_m t_m k(x, x_m) + b$$

$\Rightarrow$ In both training and testing, examples are used only through the *kernel function*!

Support Vector Machines

- Optimization problem:

upper bound on the **misclassification error** on the training data.

minimize:

$$J(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\boxed{\sum_{n=1}^{N}\xi_n}$$

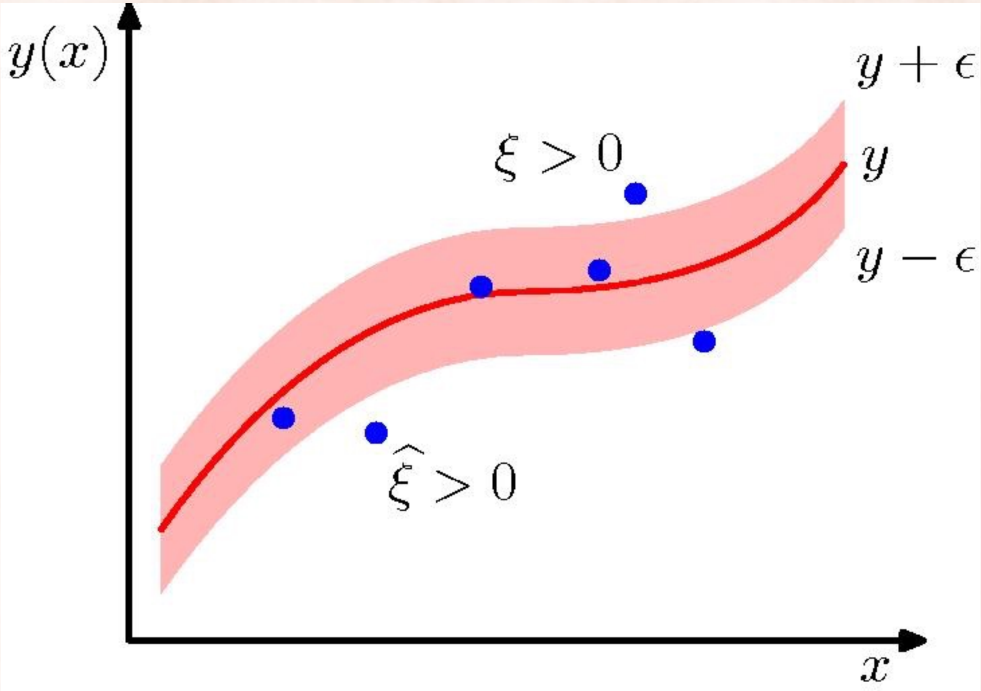
subject to:

$$t_n(\mathbf{w}^T\varphi(\mathbf{x}_n)+b) \geq 1-\xi_n, \quad \forall n \in \{1,\ldots,N\}$$
$$\xi_n \geq 0$$

- Implemented in *sklearn*:
  - https://scikit-learn.org/stable/modules/svm.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

SVMs for Regression

- Use an ε-insensitive error function (ε > 0) to obtain *sparse solutions*.
  - Penalty that increases with the distance from the ε-insensitive “tube”.



SVMs for Regression: Quadratic Optimization

- Optimization problem:

minimize:

$$J(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n)$$

subject to:

$$t_n \leq \mathbf{w}^T\varphi(\mathbf{x}_n) + b + \varepsilon + \xi_n$$

$$t_n \geq \mathbf{w}^T\varphi(\mathbf{x}_n) + b - \varepsilon - \hat{\xi}_n$$

$$\xi_n, \hat{\xi}_n \geq 0, \quad \forall n \in \{1,\ldots,N\}$$

- Solve it using the technique of Lagrange Multipliers.

SVMs for Regression: Sparse Solution

- After solving the dual problem $\Rightarrow$ know $\alpha_n, \hat{\alpha}_n$ for $n = 1 \ldots N$

$$\mathbf{w} = \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n)\varphi(x_n) = \sum_{m \in S} (\alpha_m - \hat{\alpha}_m)\varphi(x_m)$$

- S is the set of *support vectors*:

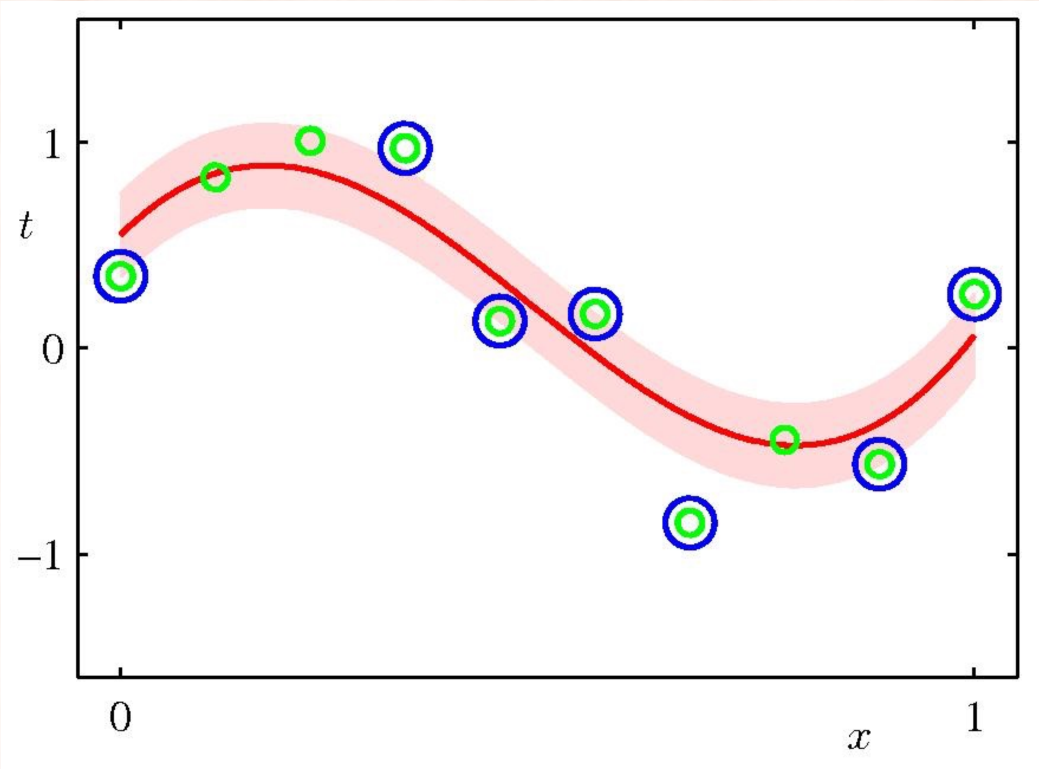
  i.e. points for which either $\alpha_n \neq 0$ or $\hat{\alpha}_n \neq 0$

  $\Rightarrow$ points that lie on the boundary of the ε-insensitive tube or outside the tube

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{m \in S} (\alpha_m - \hat{\alpha}_m) k(x, x_m) + b$$

$\Rightarrow$ In both training and testing, examples are used only through the *kernel function*!

SVMs for Regression: Sparse Solution



SVMs for Ranking [Joachims, KDD'02]

- Problem:
  - For a query q, a search engine returns a set of documents D.
  - Want to rank d_i higher than d_j if d_i is more relevant to q than d_j.

- Solution:
  - Learn a ranking function $f(q,d) = \mathbf{w}^T \varphi(q,d)$
  - Rank d_i higher than d_j if $f(q,d_i) \geq f(q,d_j) \Leftrightarrow \mathbf{w}^T \varphi(q,d_i) \geq \mathbf{w}^T \varphi(q,d_j)$
  - Training data:
    - Set $\{(q_k, d_i, d_j) \mid d_i$ ranked higher than d_j for query $q_k\}$.
    - Relative rankings obtained from clicktrough data.

SVMs for Ranking [Joachims, KDD'02]

- Optimization problem:

minimize:

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_{k,i,j}$$

subject to:

$$\boxed{\mathbf{w}^T \varphi(q_k, d_i) \geq \mathbf{w}^T \varphi(q_k, d_j) + 1 - \xi_{k,i,j}}$$

$$\xi_{k,i,j} \geq 0$$

$$\boxed{\mathbf{w}^T \left(\varphi(q_k, d_i) - \varphi(q_k, d_j)\right) \geq 1 - \xi_{k,i,j}}$$ $\Rightarrow$ equivalent with a classification problem

SVMs for Ranking [Joachims, KDD'02]

- After solving the quadratic problem:

$$\mathbf{w} = \sum_{k,l} \alpha_{k,l} \varphi(q_k, d_l)$$

$$\Rightarrow \; f(q,d) = \mathbf{w}^T \varphi(q,d)$$

$$= \sum_{k,l} \alpha_{k,l} \varphi^T(q_k, d_l) \varphi(q,d)$$

$$= \sum_{k,l} \alpha_{k,l} K(q_k, d_l, q, d)$$

$\Rightarrow$ In both training and testing, examples are used only through the *kernel function*!

Learning Scenarios for SVMs

- **Classification.**
- **Ranking.**
- **Regression.**
- Ordinal Regression.
- One Class Learning.
- Learning with Positive and Unlabeled examples.
- Transductive Learning.
- Semi-Supervised Learning.
- Multiple Instance Learning.
- Structured Outputs.

Practical Issues

- **Data Scaling**:
  - Between [-1,+1] or [0, 1].
  - Use same scaling factors in training and testing!
- **Parameter Tuning**:
  - Most SVM packages specify reasonable default values.
    - Tuning helps, especially with kernels that tend to overfit.
  - Grid search is simple and effective:
    - For RBF kernels, need to tune C and γ:
      - $C \in \{2^{-5}, 2^{-3}, ..., 2^{15}\}$, $\gamma \in \{2^{-15}, 2^{-13}, ..., 2^{3}\}$
- Read LibSVM's "A practical guide to SVM classification".

Conclusion

- SVMs were originally proposed by Boser, Guyon, and Vapnik in 1992.
- Good performance on a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types, e.g. *graphs, trees, sequences*, by designing kernel functions for such data.
  - Also to probability distributions – "Learning from Distributions via Support Measure Machines" [Muandet et al., NIPS 2012]
- Kernel trick has been extended to other methods such as Perceptron, PCA, kNN, etc.
- Popular optimization algorithms for SVMs use decomposition to hill-climb over a subset of α_n's at a time, e.g. SMO [Platt '99].
  - But training and testing with linear SVMs are much faster.
- Read Lin's "Machine Learning Software: Design and Practical Use"

Supplementary Readings (mandatory for 8156)

- PRML, Chapter 7:
  - Most of Section 7.1 on Maximum Margin Classifiers.
- PRML, Appendix E on Langange Multipliers.

Convex Optimization

- Convex optimization problem in standard form (primal):

$$\begin{aligned}&\text{minimize:}\\ &\qquad f_0(\mathbf{x})\\ &\text{subject to:}\\ &\qquad f_i(\mathbf{x}) \le 0, \quad i = 1,\ldots,m\\ &\qquad h_i(\mathbf{x}) = 0, \quad i = 1,\ldots,p\end{aligned}$$

solution $\mathbf{x}^*$

  - f_i : $R^n \rightarrow R$ are all **convex functions**, for $i = 0, \ldots, m$
  - h_i : $R^n \rightarrow R$ are all **afine functions** , for $i = 0, \ldots, p$ (e.g. $h_i(\mathbf{x})=\mathbf{A}\mathbf{x}+\mathbf{b}$)

Lagrange Multipliers

- Define Lagrangian function $L_P : \mathrm{R}^n \times \mathrm{R}^m \times \mathrm{R}^p \to \mathrm{R}$:

$$L_P(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\upsilon}) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \upsilon_i h_i(x)$$

- $\lambda_i \geq 0$, and v_i are the *Lagrange multipliers*.

- Define Lagrange dual function $L_D : \mathrm{R}^m \times \mathrm{R}^p \to \mathrm{R}$:

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\upsilon}) = \inf_{\mathbf{x}} L_P(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\upsilon})$$

Convex Optimization

- Lagrange Dual Problem:

maximize:

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\upsilon})$$

subject to:

$$\lambda_i \geq 0, \quad i = 1, \ldots, m$$

solution $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\upsilon}) = \inf_{\mathbf{x}} L_P(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\upsilon})$$

Strong Duality

minimize:

$$f_0(\mathbf{x})$$

subject to:

$$f_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m$$
$$h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p$$

$\Leftrightarrow$

maximize:

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\upsilon})$$

subject to:

$$\lambda_i \ge 0, \quad i = 1, \ldots, m$$

solution $\mathbf{x}^*$

solution $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$

- Optimum for primal problem = optimum for dual problem:

$$f_0(\mathbf{x}^*) = L_D(\boldsymbol{\lambda}^*, \boldsymbol{\upsilon}^*)$$

Karush–Kuhn–Tucker (KKT) conditions

Assume ($\mathbf{x}$, $\boldsymbol{\lambda}$, $\boldsymbol{\nu}$) are the primal & dual solutions. Then ($\mathbf{x}$, $\boldsymbol{\lambda}$, $\boldsymbol{\nu}$) satisfy the following constraints:

1. primal constraints: $\begin{cases} f_i(\mathbf{x}) \le 0, & i = 1, \ldots, m \\ h_i(\mathbf{x}) = 0, & i = 1, \ldots, p \end{cases}$
2. dual constraints: $\lambda_i \ge 0, \quad i = 1, \ldots, m$
3. complementary slackness: $\lambda_i f_i(\mathbf{x}) = 0, \quad i = 1, \ldots, m$
4. gradient of Lagrangian with respect to **x** vanishes:

$$\nabla L_P(\mathbf{x}) = \nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} \upsilon_i \nabla h_i(x) = 0$$

Max-Margin: Quadratic Optimization

- Constrained optimization problem:

minimize:

$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$t_n(\mathbf{w}^T \varphi(\mathbf{x}_n) + b) \geq 1, \quad \forall n \in \{1, \ldots, N\}$$

- Let's solve it using the technique of Lagrange Multipliers.

Max-Margin: Quadratic Optimization

- Lagrangian function:

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} \alpha_n \left\{ t_n(\mathbf{w}^T \varphi(x_n) + b) - 1 \right\}$$

  - $\alpha_n \geq 0$ are the *Lagrangian multipliers*.

- Lagrangian dual function:

$$L_D(\boldsymbol{\alpha}) = \inf_{\mathbf{w}, b} L_P(\mathbf{w}, b, \boldsymbol{\alpha})$$

- Solve:

$$\left.\begin{array}{l} \dfrac{\partial L_P}{\partial \mathbf{w}} = 0 \\ \dfrac{\partial L_P}{\partial b} = 0 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \varphi(x_n) \\ \sum_{n=1}^{N} \alpha_n t_n = 0 \end{array}\right.$$