CS 2750 Machine Learning
Lecture 15

Support vector machines for regression

Milos Hauskrecht

milos@cs.pitt.edu

5329 Sennott Square

Support vector machines

- **The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- **The decision:**

$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

- **(!!):**
- Decision on a new $\mathbf{x}$ requires to compute the inner product between the examples $(\mathbf{x}_i^T \mathbf{x})$
- Similarly, the optimization depends on $(\mathbf{x}_i^T \mathbf{x}_j)$

$$J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

Nonlinear case

- The linear case requires to compute $(\mathbf{x}_i^T \mathbf{x})$
- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$$

- It is possible to use SVM formalism on feature vectors

$$\boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Kernel function**

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Crucial idea:** If we choose the kernel function wisely we can compute linear separation in the feature space implicitly such that we keep working in the original input space !!!!

Kernel function example

- Assume $\mathbf{x} = [x_1, x_2]^T$ and a feature mapping that maps the input into a quadratic feature set

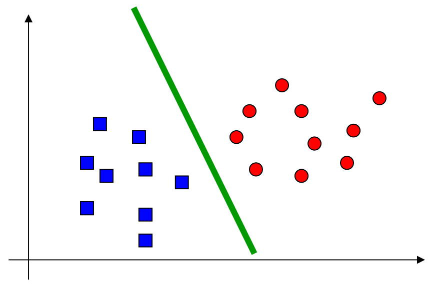
$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

- Kernel function for the feature space:

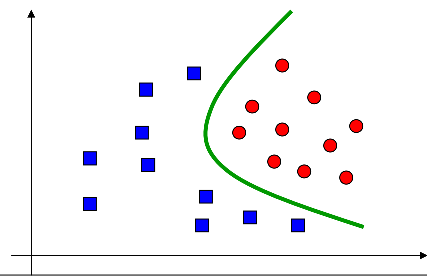
$$\begin{aligned} K(\mathbf{x}', \mathbf{x}) &= \boldsymbol{\varphi}(\mathbf{x}')^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1x_2x'_1\, x'_2 + 2x_1x'_1 + 2x_2x'_2 + 1 \\ &= (x_1x'_1 + x_2x'_2 + 1)^2 \\ &= (1 + (\mathbf{x}^T\mathbf{x}'))^2 \end{aligned}$$

- The computation of the linear separation in the higher dimensional space is performed implicitly in the original input space

Kernel function example



Linear separator
in the feature space



Non-linear separator
in the input space

Kernel functions

- **Linear kernel**

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = \left[1 + \mathbf{x}^T \mathbf{x}'\right]^k$$

- **Radial basis kernel**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2} \|\mathbf{x} - \mathbf{x}'\|^2 \right]$$

Kernels

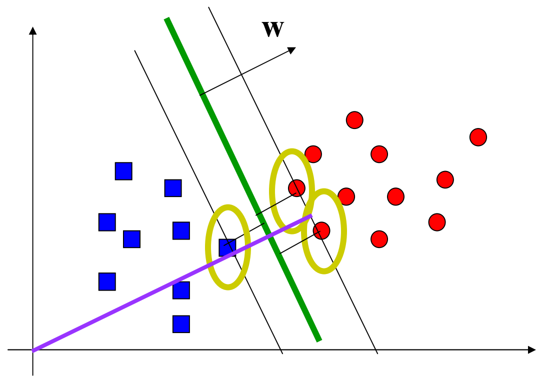
- The dot product $\mathbf{x}^T\mathbf{x}$ is a **distance measure**
- **Kernels** can be seen as distance measures
  - Or conversely express degree of similarity
- Design criteria - we want kernels to be
  - valid – Satisfy Mercer condition of positive semidefiniteness
  - good – embody the "true similarity" between objects
  - appropriate – generalize well
  - efficient – the computation of k(x,x') is feasible
    - NP-hard problems abound with graphs

Kernels

- Research have proposed kernels for comparison of variety of objects:
  - Strings
  - Trees
  - Graphs
- **Cool thing:**
  - SVM algorithm can be now applied to classify a variety of objects

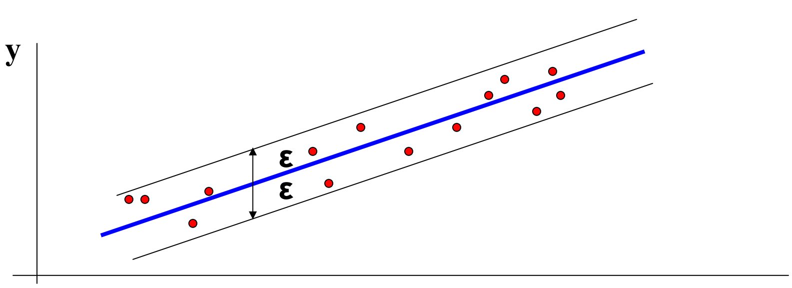
Support vector machine SVM

- SVM maximize the margin around the separating hyperplane.
- The decision function is fully specified by a subset of the training data, the support vectors.



Support vector machine for regression

- **Regression** = find a function that fits the data.
- A data point may be wrong due to the noise
- **Idea:** Error from points which are close **should count as a valid noise**
- Line should be influenced by the real data not the noise.



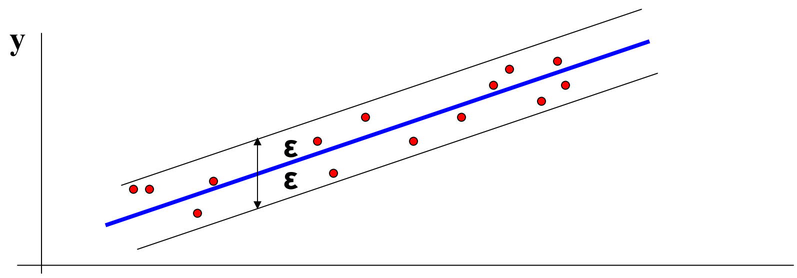
Linear model

- **Training data:**

  $$\{(x_1, y_1), \ldots, (x_l, y_l)\}, \quad x \in R^n, y \in R$$

- Our goal is to find a function f(x) that has at most ε deviation from the actually obtained target for all the training data.

$$f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$



Linear model

Linear function:

$$f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

We want a function that is:

- **flat:** means that one seeks small **w**
- all data points are within its ε neighborhood

The problem can be formulated as a **convex optimization problem:**

$$\text{minimize} \quad \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad \begin{cases} \mathrm{y_i} - \langle w_i, x_i \rangle - b \le \varepsilon \\ \langle w_i, x_i \rangle + b - \mathrm{y_i} \le \varepsilon \end{cases}$$

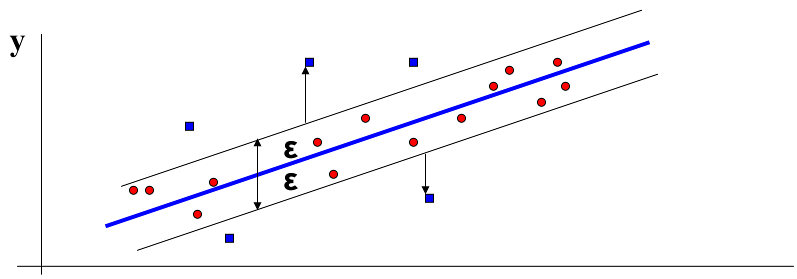
All data points are assumed to be in the ε neighborhood

Linear model

- **Real data: not all data points always fall into the ε** neighborhood

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

- **Idea:** penalize points that fall outside the ε neighborhood



Linear model

Linear function:

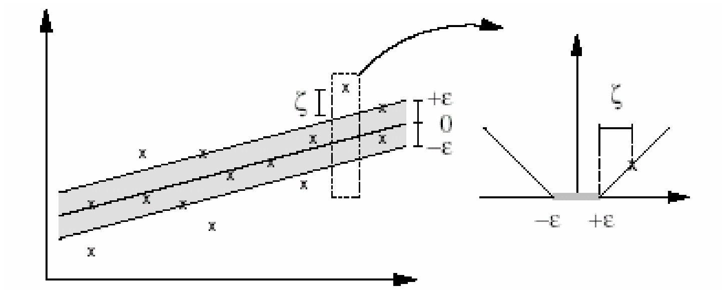
$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

Idea: penalize points that fall outside the ε neighborhood

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

$$\text{subject to} \quad \begin{cases} \mathrm{y}_i - \langle w_i, x_i \rangle - b \le \varepsilon + \xi_i \\ \langle w_i, x_i \rangle + b - \mathrm{y}_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

Linear model



$$|\xi|_\varepsilon = \begin{cases} 0 & \text{for } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases}$$

ε-intensive loss function

Optimization

Lagrangian that solves the optimization problem

$$L = \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$- \sum_{i=1}^{l} a_i(\varepsilon - \xi_i - y_i + \langle w, x_i\rangle + b) - \sum_{i=1}^{l} a_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b)$$
$$- \sum_{i=1}^{l}(\eta_i \xi_i + \eta_i^* \xi_i^*)$$

Subject to $a_i, a_i^*, \eta_i, \eta_i^* \ge 0$

Primal variables w, b, ξ_i, ξ_i^*

Optimization

Derivatives with respect to primal variables

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} (a_i^* - a_i) = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} (a_i^* - a_i)\mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial L}{\partial \xi_i^{(*)}} = C - a_i^{(*)} - \eta_i^{(*)} = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - a_i - \eta_i = 0$$

Optimization

$$L = \frac{1}{2}\langle w, w\rangle + \sum_{i=1}^{l} C\xi_i + \sum_{i=1}^{l} C\xi_i^*$$

$$- \sum_{i=1}^{l} a_i\varepsilon - \sum_{i=1}^{l} a_i\xi_i - \sum_{i=1}^{l} a_i y_i - \sum_{i=1}^{l} a_i\langle \omega, x_i\rangle + \sum_{i=1}^{l} a_i b$$

$$- \sum_{i=1}^{l} a_i^*\varepsilon - \sum_{i=1}^{l} a_i^*\xi_i^* - \sum_{i=1}^{l} a_i^* y_i + \sum_{i=1}^{l} a_i^*\langle \omega, x_i\rangle + \sum_{i=1}^{l} a_i^* b$$

$$- \sum_{i=1}^{l} \eta_i\xi_i - \sum_{i=1}^{l} \eta_i^*\xi_i^*$$

Optimization

$$L = \frac{1}{2}\langle w, w\rangle + \sum_{i=1}^{l} \xi_i \underbrace{(C - \eta_i - a_i)}_{=0 (C - \eta_i^{(*)} - a_i^{(*)} = 0)} +$$

$$\sum_{i=1}^{l} \xi_i^* \underbrace{(C - \eta_i^* - a_i^*)}_{=0 (C - \eta_i^{(*)} - a_i^{(*)} = 0)} - \sum_{i=1}^{l} (a_i + a_i^*)\varepsilon - \sum_{i=1}^{l} (a_i + a_i^*) y_i$$

$$- \sum_{i=1}^{l} \underbrace{(a_i - a_i^*)\langle \omega, x_i\rangle}_{=\langle w,w\rangle (\omega = \sum_{i=1}^{l} (a_i + a_i^*) x_i)} + \sum_{i=1}^{l} \underbrace{(a_i^* - a_i) b}_{=0 (\sum_{i=1}^{l} (a_i^* - a_i) = 0)}$$

Optimization

$$L = -\frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{l} (a_i + a_i^*)\varepsilon - \sum_{i=1}^{l} (a_i + a_i^*) y_i$$

Maximize the dual

$$L(a, a^*) = -\frac{1}{2} \sum_{i=1}^{l} (a_i - a_i^*)(a_j - a_j^*)\langle x_i, x_j\rangle$$

$$- \sum_{i=1}^{l} (a_i + a_i^*)\varepsilon - \sum_{i=1}^{l} (a_i + a_i^*) y_i$$

subject to $: \begin{cases} \sum_{i=1}^{l} (a_i - a_i^*) = 0 \\ a_i, a_i^* \in [0, C] \end{cases}$

Solution

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} (a_i^* - a_i)\mathbf{x}_i = \mathbf{0}$$

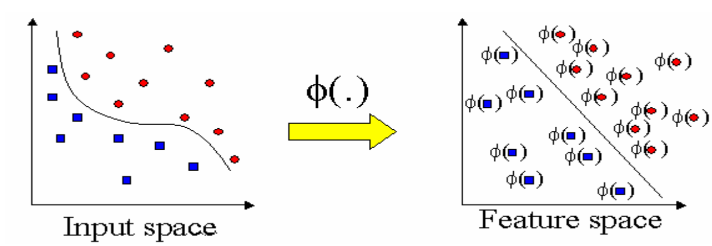
$$\mathbf{w} = \sum_{i=1}^{l} (a_i - a_i^*)\mathbf{x}_i$$

We can get:

$$f(\mathbf{x}) = \sum_{i=1}^{l} (a_i - a_i^*)\langle \mathbf{x}_\mathbf{i}, \mathbf{x} \rangle + b$$

at the optimal solution the Lagrange multipliers are non-zero only **for points outside the ε band.**

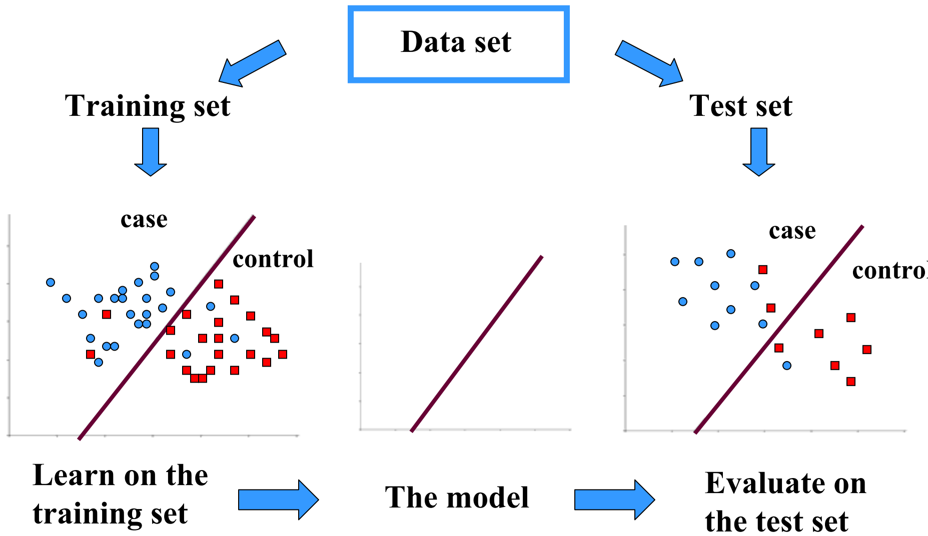
Nonlinear extension



Kernel trick

- Replace the inner product with a kernel
- A well chosen kernel leads to efficient computation

Evaluation framework



Evaluation metrics

Confusion matrix: Records the percentages of examples in the testing set that fall into each group

		Actual: Case	Actual: Control
Prediction	Case	**TP** 0.3	**FP** 0.1
	Control	**FN** 0.2	**TN** 0.4

Misclassification error:

$$E = FP + FN$$

Sensitivity:

$$SN = \frac{TP}{TP + FN}$$

Specificity:

$$SP = \frac{TN}{TN + FP}$$

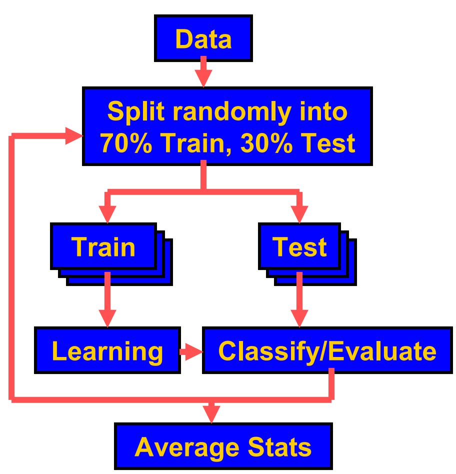
Evaluation

- **Problem:** if the sample size is relatively small one split may be lucky or unlucky hence biasing the statistics
- **Solution:** use multiple train/test splits and average their results

- **Random resampling validation techniques:**
  - **random sub-sampling**
  - **k-fold cross-validation**
  - **bootstrap-based validation**

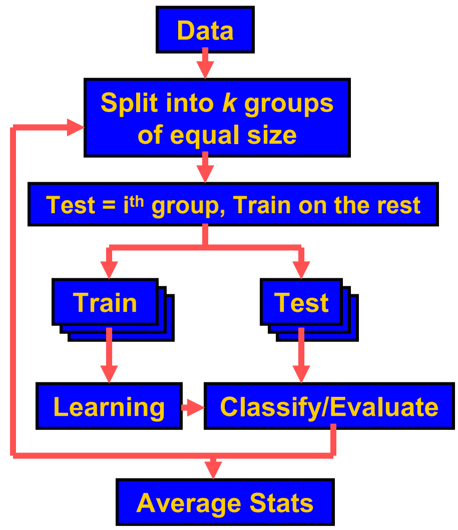
Random sub-sampling

- Split the data into **train and test set** with some split ratio (typically 70:30)
- Repeat this k times for different random splits
- Average the results of statistics



K-fold cross-validation

- Split the data into k **equal size groups**
- Use each group once as a test set, and the remaining groups as the training set
- Repeat this k times for k groups
- Average the results of statistics



Bootstrap-based validation

- **Bootstrap technique** –used primarily to estimate the sampling distribution of an estimator
- **Generate randomly with replacement** a training dataset of size n that equals the original data size
- Some examples are repeated in the training set, some are missing
- Build a test set from examples not used in the training set.