SVMs for regression

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

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 machines

- **The decision boundary:**
  \[
  \hat{w}^T x + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0
  \]

- **The decision:**
  \[
  \hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0 \right]
  \]

- **(!!):**
  - Decision on a new \( x \) requires to compute the inner product between the examples \( (x_i^T x) \)
  - Similarly, the optimization depends on \( (x_i^T x_j) \)
  \[
  J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
  \]

Nonlinear case

- The linear case requires to compute \( (x_i^T x) \)
- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors
  \[
  x \rightarrow \phi(x)
  \]
- It is possible to use SVM formalism on feature vectors
  \[
  \phi(x)^T \phi(x')
  \]
- **Kernel function**
  \[
  K(x, x') = \phi(x)^T \phi(x')
  \]
- **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

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

- Kernel function for the feature space:

\[
K(\mathbf{x}', \mathbf{x}) = \mathbf{\phi}(\mathbf{x}')^T \mathbf{\phi}(\mathbf{x})
= x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_2x_1'x_2' + 2x_1x_1' + 2x_2x_2' + 1
= (x_1x_1' + x_2x_2' + 1)^2
= (1 + (\mathbf{x}^T\mathbf{x}'))^2
\]

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

---

Nonlinear extension

**Kernel trick**

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

---
Kernel functions

- **Linear kernel**
  \[ K(x, x') = x^T x' \]

- **Polynomial kernel**
  \[ K(x, x') = [1 + x^T x']^k \]

- **Radial basis kernel**
  \[ K(x, x') = \exp\left(-\frac{1}{2}||x - x'||^2\right) \]

Kernels

- The dot product \( x^T x \) 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 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),..., (x_n, y_n)\}, x \in \mathbb{R}^n, y \in \mathbb{R}

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

\[f(x) = w^T x + b = \langle w, x \rangle + b\]

Linear function:

\[f(x) = w^T x + b = \langle w, x \rangle + b\]

We want a function that is:

• flat: means that one seeks small \(w\)
• all data points are within its \(\varepsilon\) neighborhood

The problem can be formulated as a convex optimization problem:

minimize \(\frac{1}{2} ||w||^2\)

subject to

\[\begin{align*}
  y_i - \langle w, x_i \rangle - b & \leq \varepsilon \\
  \langle w, x_i \rangle + b - y_i & \leq \varepsilon
\end{align*}\]

All data points are assumed to be in the \(\varepsilon\) neighborhood
Linear model

- **Real data**: not all data points always fall into the $\varepsilon$ neighborhood
  \[ f(x) = w^T x + b = \langle w, x \rangle + b \]
- **Idea**: penalize points that fall outside the $\varepsilon$ neighborhood

![Linear model diagram](image)

**Linear function:**

\[ f(x) = w^T x + b = \langle w, x \rangle + b \]

**Idea:** penalize points that fall outside the $\varepsilon$ neighborhood

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{I} (\xi_i + \xi_i^*) \\
\text{subject to} & \quad y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\
& \quad \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\
& \quad \xi_i, \xi_i^* \geq 0
\end{align*}
\]
Linear model

\[ |\xi| = \begin{cases} 0 & \text{for } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \]

\( \varepsilon \)-intensive loss function

Optimization

Lagrangian that solves the optimization problem

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

Subject to \( a_i, a_i^*, \eta_i, \eta_i^* \geq 0 \)

Primal variables \( w, b, \xi_i, \xi_i^* \)
Optimization

Derivatives with respect to primal variables

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

\[ \frac{\partial L}{\partial w} = w - \sum_{i=1}^{l} (a_i^* - a_i) x_i = 0 \]

\[ \frac{\partial L}{\partial \xi_i^*} = C - a_i^* - \eta_i^* = 0 \]

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

Optimization

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

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

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

\[ - \sum_{i=1}^{l} \eta_i \xi_i - \sum_{i=1}^{l} \eta_i^* \xi_i^* \]
Optimization

\[
L = \frac{1}{2} \langle w, w \rangle + \sum_{i=1}^{l} \xi_i \left( C - \eta_i - a_i \right) + \sum_{i=1}^{l} \xi_i \left( C - \eta_i^* - a_i^* \right) - \sum_{i=1}^{l} (a_i + a_i^*) \epsilon - \sum_{i=1}^{l} (a_i + a_i^*) y_i
\]

\[
\sum_{i=1}^{l} \xi_i \left( C - \eta_i^* - a_i^* \right) - \sum_{i=1}^{l} (a_i + a_i^*) \epsilon - \sum_{i=1}^{l} (a_i + a_i^*) y_i
\]

Optimization

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

Maximize the dual

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

\[
- \sum_{i=1}^{l} (a_i + a_i^*) \epsilon - \sum_{i=1}^{l} (a_i + a_i^*) y_i
\]

subject to:

\[
\sum_{i=1}^{l} (a_i - a_i^*) = 0
\]

\[
a_i, a_i^* \in [0, C]
\]
Solution

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

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

We can get:

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

at the optimal solution the Lagrange multipliers are non-zero only for points outside the \( \varepsilon \) band.