Maximum Margin Classifiers

- We have already defined the Maximum Margin criterion
  \[
  \max_w \min_{x^i} y_i(w \cdot x^i + w_0)
  \]
  Subject to \(\|w\|^2 = 1\)

- and have shown that it is equivalent to the optimization problem:
  \[
  \min_v \|v\|^2 \\
  \text{Subject to } y_i(v \cdot x^i + v_0) \geq 1
  \]

Maximum Margin Classifiers

\[
\min_v \|v\|^2 \\
\text{Subject to } y_i(v \cdot x^i + v_0) \geq 1
\]

This is a Quadratic Optimization Problem: optimizing a quadratic function of \(v\) subject to linear constraints on \(v\).

Algorithms (and software packages) for such problems exist.

Also known as Quadratic Programming: QP

### Primal/Dual SVM

- By forming the Lagrangian and following standard procedures in optimization we can translate the "primal" problem into a "dual" problem that provides the same solutions.

\[
\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (x^i \cdot x^j) \\
\text{Subject to } \sum_{i=1}^{N} \alpha_i y_i = 0 \\
\alpha_i \geq 0
\]

### Dual SVM: some properties

- This is also a QP
- The first constraint: equal weight to positive and negative examples

\[
\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (x^i \cdot x^j) \\
\text{Subject to } \sum_{i=1}^{N} \alpha_i y_i = 0 \\
\alpha_i \geq 0
\]
Dual SVM

\[
\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (x^i \cdot x^j)
\]
Subject to \(\sum_{i=1}^{N} \alpha_i y_i = 0\)
\(\alpha_i \geq 0\)

• The corresponding primal solution is:
\[w = \sum_{k} \alpha_k y_k x^k\]
• Same as dual perceptron!
• \(\alpha_k = 0\) unless \(x^k\) is "on the margin"

Dual SVM

\[
\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (x^i \cdot x^j)
\]
Subject to \(\sum_{i=1}^{N} \alpha_i y_i = 0\)
\(\alpha_i \geq 0\)

• The corresponding primal solution is:
\[w = \sum_{k} \alpha_k y_k x^k\]
• \(\alpha_k = 0\) unless \(x^k\) is "on the margin"
\(\alpha_k \neq 0 \rightarrow x^k\) is a "support vector"

Dual SVM

\[
\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (x^i \cdot x^j)
\]
Subject to \(\sum_{i=1}^{N} \alpha_i y_i = 0\)
\(\alpha_i \geq 0\)

• Using examples only through inner products \(\rightarrow\) can be used with kernels

Summary: "Hard Margin" SVM

The primal formulation is given by
\[
\min \|v\|^2
\]
Subject to \(y_i (v \cdot x^i + v_0) \geq 1\)

The dual formulation is given by
\[
\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(x^i, x^j)
\]
Subject to \(\sum_{i=1}^{N} \alpha_i y_i = 0\)
\(\alpha_i \geq 0\)

Max Margin Classifier

• Consider again the original problem
\[
\min \|v\|^2
\]
Subject to \(y_i (v \cdot x^i + v_0) \geq 1\)

• There is a problem when the data is noisy or just not linearly separable
  • Why?
  • How can we get around it?
Soft Margin SVM

- Consider again the original problem
  \[
  \min_v ||v||^2
  \]
  Subject to \(y_i(v \cdot x_i + v_0) \geq 1\)

- Allowing slack for "hard to separate" points
  \[
  \min_v ||v||^2 + C \sum_i \xi_i
  \]
  Subject to \(y_i(v \cdot x_i + v_0) \geq 1 - \xi_i\)
  \(\xi_i \geq 0\)

Primal & Kernel Soft Margin SVM

\[
\min_{v, \xi} ||v||^2 + C \sum_i \xi_i
\]
Subject to \(y_i(v \cdot x_i + v_0) \geq 1 - \xi_i\)
\(\xi_i \geq 0\)

The dual formulation is given by

\[
\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j K(x_i, x_j)
\]
Subject to \(\sum_i \alpha_i y_i = 0\)
\(0 \leq \alpha_i \leq C\)

Support vector machines

- Max margin linear separators
- Soft margin can tolerate "noisy data"
- And is the standard approach in practice
- Both versions are kernel methods
- Solved with QP optimization packages
- And/or with specialized SVM solvers
- Must tune C and Kernel parameters

SVM in Practice

- Very successful.
- Robust and mature systems, e.g., libsvm
- Important to normalize features
- Important to pick kernel for problem
- Important to pick good parameter setting for C and any kernel parameters