Kernels: Key Points

- Many learning tasks are framed as optimization problems
- Primal and Dual formulations of optimization problems
- Dual version framed in terms of dot products between x's
- Kernel functions $k(x,y)$ allow calculating dot products $<\Phi(x),\Phi(y)>$ without bothering to project $x$ into $\Phi(x)$
- Leads to major efficiencies, and ability to use very high dimensional (virtual) feature spaces
Simple Kernel Based Classifier

- Consider finding the centres of mass of positive and negative examples and classifying a test point by measuring which is closest

\[ h(x) = \text{sgn} \left( \| \phi(x) - \phi_{S-} \|^2 - \| \phi(x) - \phi_{S+} \|^2 \right) \]

- we can express as a function of kernel evaluations

\[ h(x) = \text{sgn} \left( \frac{1}{m_+} \sum_{i=1}^{m_+} k(x, x_i) - \frac{1}{m_-} \sum_{i=m_++1}^{m} k(x, x_i) - b \right) \]

where

\[ b = \frac{1}{2m_+^2} \sum_{i,j=1}^{m_+} k(x_i, x_j) - \frac{1}{2m_-^2} \sum_{i,j=m_++1}^{m} k(x_i, x_j) \]

[slide from John Shawe-Taylor]
Linear classifiers – which line is better?

Pick the one with the largest margin!
Parameterizing the decision boundary

\[ w^T x + b > 0 \quad w^T x + b < 0 \]

Labels \( y_i \in \{-1, +1\} \quad \text{class} \)

SVM: Maximize the margin

Margin = Distance of closest examples from the decision line/hyperplane

\[ \text{margin} = \gamma = a/\|w\| \]
Maximizing the margin

Margin = Distance of closest examples from the decision line/hyperplane

Margin = \( \gamma = a/\|w\| \)

\[
\begin{align*}
\max & \quad \gamma = a/\|w\| \\
& \text{s.t. } (w^T x_j + b) y_j \geq a \quad \forall j
\end{align*}
\]

Note: ‘a’ is arbitrary (can normalize equations by a)

Support Vector Machine (primal form)

Primal form:

\[
\begin{align*}
\min & \quad w^T w \\
& \text{s.t. } (w^T x_j + b) y_j \geq 1 \quad \forall j
\end{align*}
\]

Solve efficiently by quadratic programming (QP)
- Well-studied solution algorithms
We can solve either primal or dual forms

**Primal form:** solve for \( w, b \)

\[
\min_{w, b} \quad w^T w \\
\text{s.t.} \quad y_l(w^T x_l + b) \geq 1 \quad \forall l \in \text{training examples}
\]

Classification test for new \( x \): \( w^T x + b > 0 \)

**Dual form:** solve for \( \alpha_1 \ldots \alpha_M \)

\[
\max_{\alpha_1 \ldots \alpha_M} \quad \sum_{l=1}^{M} \alpha_l - \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} \alpha_j \alpha_k y_j y_k \langle x_j, x_k \rangle \\
\text{s.t.} \quad \alpha_l \geq 0 \quad \forall l \in \text{training examples} \\
\sum_{l=1}^{M} \alpha_l y_l = 0
\]

Classification test for new \( x \): \( \sum_{l \in \text{SVs}} \alpha_l y_l \langle x, x_l \rangle + b > 0 \)

Both are QP problems with a single local optimum!

---

**What is quadratic programming?**

A way to solve optimization problems of the form:

\[
\text{Minimize } f(x) = cx + \frac{1}{2} x^T Q x \\
\text{subject to } Ax \leq b \text{ and } x \geq 0
\]

Where \( Q \) is symmetric matrix.

If \( f(x) \) is strictly convex for all feasible points (e.g., \( Q \) is positive definite)
then there is only one local minimum, which is global min.
Support Vectors

\[ \sum_{l \in SVs} \alpha_l y_l \langle x, x_l \rangle + b > 0 \]

Linear hyperplane defined by "support vectors" *

\[ \sum_{l \in SVs} \alpha_l y_l \langle x, x_l \rangle + b < 0 \]

Moving other points a little doesn’t effect the decision boundary

\[ w^T x + b > 0 \]

only need to store the support vectors to predict labels of new points

How many support vectors in linearly separable case, given d dimensions?

\[ \leq d + 1 \]

* KKT conditions on optimization problem assure other \( \alpha = 0 \)

Kernel SVM

And because the dual form depends only on inner products, we can apply the kernel trick to work in a (virtual) projected space \( \Phi : X \rightarrow F \)

Primal form: solve for \( w, b \) in the projected higher dim. space

\[
\begin{align*}
\min_{w,b} & \quad w^T w \\
\text{s.t.} & \quad y_l(w^T \Phi(x_l) + b) \geq 1 \quad \forall l \in \text{training examples}
\end{align*}
\]

Classification test for new \( x \) \( w^T \Phi(x) + b > 0 \)

Dual form: solve for \( \alpha_1 \ldots \alpha_M \) in the original low dim. space

\[
\begin{align*}
\max_{\alpha_1 \ldots \alpha_M} & \quad \sum_{l=1}^{M} \alpha_l - \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} \alpha_j \alpha_k y_j y_k \kappa(x_j, x_k) \\
\text{s.t.} & \quad \alpha_l \geq 0 \quad \forall l \in \text{training examples} \\
& \quad \sum_{l=1}^{M} \alpha_l y_l = 0
\end{align*}
\]

Classification test for new \( x \) \( \sum_{l \in SVs} \alpha_l y_l \kappa(x, x_l) + b > 0 \)
SVM Decision Surface using Gaussian Kernel

\[
\hat{f}(x) = w^T \Phi(x) + b
\]

Circled points are the support vectors: training examples with non-zero \( \alpha_l \)

Points plotted in original 2-D space.

Contour lines show constant \( \hat{f}(x) \)

\[
\hat{f}(x) = b + \sum_{i=1}^{M} \alpha_i y_i \kappa(x, x_i) = b + \sum_{i=1}^{M} \alpha_i y_i \exp\left(-\|x - x_i\|^2/2\sigma^2\right)
\]

What if data is not linearly separable?

Use features of features of features of features.. .

\[x_1^2, x_2^2, x_1x_2, \ldots, \exp(x_1)\]

But run risk of overfitting!
What if data is still not linearly separable?

Allow “error” in classification

\[
\begin{align*}
\min_{w,b} & \quad w^T w + C \sum_j \xi_j \\
\text{s.t.} & \quad (w^T x_j + b) y_j \geq 1 - \xi_j, \quad \forall j \\
& \quad \xi_j \geq 0, \quad \forall j
\end{align*}
\]

\(\xi_j\) - “slack” variables
- \(= (>1 \text{ if } x_j \text{ misclassified})\)
- pay linear penalty if mistake

C - tradeoff parameter (chosen by cross-validation)

Support Vector Machine with soft margins

Allow “error” in classification

Maximize margin and minimize # mistakes on training data

C - tradeoff parameter

Not QP 😞

0/1 loss (doesn’t distinguish between near miss and bad mistake)
Primal and Dual Forms for Soft Margin SVM

Primal form: solve for $w, b$ in the projected higher dim. space

$$\min_{w,b} \frac{1}{2}w^Tw + C \sum_{l=1}^{M} \xi_l$$

s.t. $y_l(w^T\Phi(x_l) + b) \geq 1 - \xi_l \quad \forall l \in \text{training examples}$

$$\xi_l \geq 0 \quad \forall l \in \text{training examples}$$

Dual form: solve for $\alpha_1...\alpha_M$ in the original low dim. space

$$\max_{\alpha_1...\alpha_M} \sum_{l=1}^{M} \alpha_l - \frac{1}{2} \sum_{j=1}^{M} \sum_{k=1}^{M} \alpha_j \alpha_k y_j y_k \kappa(x_j, x_k)$$

s.t. $0 \leq \alpha_l \leq C \quad \forall l \in \text{training examples}$

$$\sum_{l=1}^{M} \alpha_l y_l = 0$$

both are QP problems with a single local optimum

SVM Soft Margin Decision Surface using Gaussian Kernel

Circed points are the support vectors: training examples with non-zero $\alpha_l$

Points plotted in original 2-D space.

Contour lines show constant $\hat{f}(x)$

$$\hat{f}(x) = b + \sum_{l=1}^{M} \alpha_l y_l \kappa(x, x_l) = b + \sum_{l=1}^{M} \alpha_l y_l \exp(-\|x - x_l\|^2/2\sigma^2)$$
What about multiple classes?

One against all

Could try to learn 3 separate classifiers:
Class k vs. rest

$(w_k, b_k)_{k=1,2,3}$

$y = \arg \max_k w_k^T x + b_k$

But $w_k$'s might not be on the same scale.
Note: $(aw)x + (ab)$ is also a solution
Learn single, Multi-class SVM

Simultaneously learn 3 sets of weights

\[ w^{(y_j)} x_j + b^{(y_j)} \geq w^{(y')} x_j + b^{(y')} + 1, \quad \forall y' \neq y_j, \forall j \]

Margin = gap between correct class and nearest other class

Classification rule:

\[ y = \arg \max_k w^{(k)} x + b^{(k)} \]

Learn single Multi-class SVM

Simultaneously learn 3 sets of weights

\[ \text{minimize}_{w,b} \sum_y w(y), w(y) + C \sum_j \sum_{y' \neq y_j} \xi_j^{(y')} \]

such that:

\[ w^{(y_j)} x_j + b^{(y_j)} \geq w^{(y)} x_j + b^{(y)} + 1 - \xi_j^{(y)}, \quad \forall y \neq y_j, \forall j \]

\[ \xi_j^{(y)} \geq 0 \]

Classification rule:

\[ y = \arg \max_k w^{(k)} x + b^{(k)} \]

Joint optimization: \( w \)'s are on the same scale.
SVM Summary

• Objective: maximize margin between decision surface and data
• Primal and dual formulations
  – dual represents classifier decision in terms of support vectors
• Kernel SVM’s
  – learn linear decision surface in high dimension space, working in original low dimension space
• Handling noisy data: soft margin “slack variables”
  – again primal and dual forms
• SVM algorithm: Quadratic Program optimization
  – single global optimum

Maximizing Margin as an Objective Function

• We’ve talked about many learning algorithms, with different objective functions
  • 0-1 loss
  • sum sq error
  • maximum log data likelihood
  • MAP
  • maximum margin

How are these all related?
Slack variables – Hinge loss

Complexity penalization
\[
\xi_j = \text{loss}(f(x_j), y_j)
\]
\[
f(x_j) = \text{sgn}(w \cdot x_j + b)
\]
\[
\xi_j = (1 - (w \cdot x_j + b)y_j)_+
\]

\[
\min_{w,b} w^T w + C \sum_j \xi_j
\]
\[
\text{s.t. } (w^T x_j + b) y_j \geq 1 - \xi_j \quad \forall j
\]
\[
\xi_j \geq 0 \quad \forall j
\]

SVM vs. Logistic Regression

**SVM**: Hinge loss
\[
\text{loss}(f(x_j), y_j) = (1 - (w \cdot x_j + b)y_j)_+
\]

**Logistic Regression**: Log loss (negative log conditional likelihood)
\[
\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, w, b) = \log(1 + e^{-(w \cdot x_j + b)y_j})
\]
SVM: PAC Results?

VC dimension: examples

What is VC dimension of

- $H_2 = \{ (w_0 + w_1x_1 + w_2x_2) > 0 \Rightarrow y=1 \}$
  - $VC(H_2) = 3$
- For $H_n = \text{linear separating hyperplanes in } n \text{ dimensions}$, $VC(H_n) = n+1$

$$m \geq \frac{1}{\epsilon}(4 \log_2(2/\delta) + 8VC(H) \log_2(13/\epsilon))$$
Margin-based PAC Results

Consider a fixed distribution $D$ on pairs $(x, y)$ with $x \in \mathbb{R}^d$ satisfying $||x|| = 1$ and $y \in \{-1, 1\}$. We are interested in finding a weight vector $w$ with $||w|| = 1$ such that the sign of $w \cdot x$ predicts $y$. For $\gamma > 0$ the error rate of $w$ on distribution $D$ relative to safety margin $\gamma$, denoted $\ell_{\gamma}(w, D)$ is defined as follows.

$$\ell_{\gamma}(w, D) = \mathbb{P}_{(x, y) \sim D}[(w \cdot x)y \leq \gamma]$$

Let $S$ be a sample of $m$ pairs drawn IID from the distribution $D$. The sample $S$ can be viewed as an empirical distribution on pairs. We are interested in bounding $\ell_0(w, D)$ in terms of $\ell_{\gamma}(w, S)$ and the margin $\gamma$. Bartlett and Shawe-Taylor use fat shattering arguments [2] to show that with probability at least $1 - \delta$ over the choice of the sample $S$ we have the following simultaneously for all weight vectors $w$ with $||w|| = 1$ and margins $\gamma > 0$.

$$\ell_0(w, D) \leq \ell_{\gamma}(w, S) + 27.18\sqrt{\frac{\log^2 m + 84}{m\gamma^2}} + O\left(\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right)$$

(1)

recall:

$$\text{error}_{\text{true}}(h) < \text{error}_{\text{train}}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

Perceptron Algorithm

Perceptron Algorithm: learn $\hat{y} = h(x) = \text{sign}(\hat{w} \cdot \hat{x})$, where $\hat{x} = < 1, x_1, \ldots, x_n >$, $\hat{w} = < w_0, w_1 \ldots, w_n >$, $y \in \{-1, +1\}$

Input: $\{< \hat{x}_1, y_1 > \ldots < \hat{x}_m, y_m >\}$

Initialize $\hat{w} = 0$;

repeat

• for $i = 1$ to $m$

  - if $y_i \neq \text{sign}(\hat{w} \cdot \hat{x}_i)$

    then $\hat{w} \leftarrow \hat{w} + y_i\hat{x}_i$;

until converged
Mistake Bounds for Perceptron

When data is linearly separable:

**Theorem 1 (Block, Novikoff)** Let \( ((x_1, y_1), \ldots, (x_m, y_m)) \) be a sequence of labeled examples with \( ||x_i|| \leq R \). Suppose that there exists a vector \( u \) such that \( ||u|| = 1 \) and \( y_i(u \cdot x_i) \geq \gamma \) for all examples in the sequence. Then the number of mistakes made by the online perceptron algorithm on this sequence is at most \( (R/\gamma)^2 \).
Mistake Bounds for Perceptron

When data is linearly separable:

**Theorem 1** (Block, Novikoff) Let \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \) be a sequence of labeled examples with \( ||x_i|| \leq R \). Suppose that there exists a vector \( u \) such that \( ||u|| = 1 \) and \( y_i(u \cdot x_i) \geq \gamma \) for all examples in the sequence. Then the number of mistakes made by the online perceptron algorithm on this sequence is at most \( (R/\gamma)^2 \).

When not linearly separable: [Freund & Schapire]

**Theorem 2** Let \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \) be a sequence of labeled examples with \( ||x_i|| \leq R \). Let \( u \) be any vector with \( ||u|| = 1 \) and let \( \gamma > 0 \). Define the deviation of each example as

\[
d_i = \max\{0, \gamma - y_i(u \cdot x_i)\},
\]

and define \( D = \sqrt{\sum_{i=1}^{m} d_i^2} \). Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by

\[
\left( \frac{R + D}{\gamma} \right)^2.
\]

What you should know

Primal and Dual optimization problems
Kernel functions
Support Vector Machines
  - Maximizing margin
  - Kernel SVM’s
  - Noise, slack variables and hinge loss
  - Relationship between SVMs and logistic regression
    - 0/1 loss
    - Hinge loss
    - Log loss
Theory shows overfitting, mistakes depends on margin size