Support Vector Machines

Pascal Fua
IC-CVLab
Non-Linearly Separable Data

Adaboost can handle this using linear classifiers.

\[ \rightarrow \text{Map the data to a higher dimension.} \]
Mapping to a Higher Dimension: Three Examples

1D classification. 2D classification. Polynomial approximation.
How can we handle this 1D/2-class data?

We can map it to 2D:

\[ x \rightarrow \phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix} \]

\[\rightarrow \text{We can now use a linear classifier.}\]
2D Classification Example

How about this 2D/2-class data?
We can map the 2D data to 3D:

\[
x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix}
\]

\[ \rightarrow \text{We can now use a linear classifier.} \]
Lifting from 2D to 3D

SVM with a polynomial Kernel visualization

Created by: Udi Aharoni
Polynomial Approximation

\[(x_n, t_n)\]

For \(1 \leq n \leq N:\)
\[t_n = f(x_n) + \epsilon\]

- The \((x_i, t_i)\) are given.
- \(f\) is unknown.

- Find \(w = [w_0, w_1, \ldots, w_M]\) such that:

  \[\forall x, f(x) \approx \sum_{i=0}^{M} w_i x^i\]

- Least squares solution: \(w^* = \arg\min_w \sum_n (t_n - \sum_{i=0}^{M} w_i x_n^i)^2\)

- For \(M=1\), reduces to linear regression.
For a given M, we plot in green:

\[ f_M(x) = \sum_{i=0}^{M} w_i x^i \]
Polynomial Feature Expansion

\[ x \rightarrow \phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \]

The polynomial can be rewritten as:

\[ \sum_{i=0}^{M} w_i x^i = w \cdot \phi(x) = w^T \phi(x) \text{ with } w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix} \]

The least squares solution becomes:

\[ w^* = \underset{w}{\text{argmin}} \sum_n (t_n - w^T \phi(x_n))^2 \]
Least-Squares Formulation

\[ w^* = \arg \min_w \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2 \]

\[ = \arg \min_w \|\Phi w - t\|^2 \]

with

\[
\Phi = \begin{bmatrix}
\phi(x_1)^T \\
\phi(x_2)^T \\
\vdots \\
\phi(x_N)^T \\
\end{bmatrix} = \begin{bmatrix}
1 & x_1 & x_1^2 & \ldots & x_1^M \\
1 & x_2 & x_2^2 & \ldots & x_2^M \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_N & x_N^2 & \ldots & x_N^M \\
\end{bmatrix}, \quad w = \begin{bmatrix}
w_0 \\
w_1 \\
\vdots \\
w_M \\
\end{bmatrix}, \quad \text{and} \quad t = \begin{bmatrix}
t_0 \\
t_1 \\
\vdots \\
t_N \\
\end{bmatrix}.
\]

Intuitively: \[ \Rightarrow \Phi w^* \approx t \]

Formally: \[ \Rightarrow (\Phi^T \Phi) w^* = \Phi^T t \]
Optional: Proof Sketch

We want to minimize:

\[ R = \frac{1}{2} \| \Phi w - t \|^2 \]

\[ = \frac{1}{2} (\Phi w - t)^T (\Phi w - t) \]

The gradient of \( R \) w.r.t \( w \) is:

\[ \nabla R = \Phi^T (\Phi w - t) \]

At the minimum:

\[ 0 = \nabla R = \Phi^T (\Phi w - t) \]

\[ \Rightarrow \Phi^T \Phi w = \Phi^T t \]
Adding Noise
Regularization

\[
\mathbf{w}^* = \arg \min_{\mathbf{w}} \| \Phi \mathbf{w} - \mathbf{t} \|^2 + \frac{\lambda}{2} \| \mathbf{w} \|^2
\]

⇒ Solve: \((\Phi^T \Phi + \lambda \mathbf{I}) \mathbf{w} = \Phi^T \mathbf{t}\)

• This is known as weight decay because in iterative algorithms it encourages the weight values to decay to zero, unless supported by the data.
• It discourages large weights and therefore quick variations.
Use cross-validation data to select the value of $\lambda$. 
Linear and Non-Linear Regression

For both kind of regressions, the trick is to find the best compromise between simplicity and goodness of fit.
Application: Rainfall in Switzerland

The circles represent actual measurements

--> Extends to Higher Dimensions.
Application: Stock Price Prediction

\[ x_t = [x_{t-T+1}, \ldots, x_{t-1}, x_t] \]

\[ y(x_t; w) = x_{t+\Delta t} \]

\[ \rightarrow \text{Regression problem} \]
But Be Careful!

Never trust a statistic you have not faked yourself!

https://xkcd.com/2048/
But Be VERY Careful!

Dow Jones Industrial Average
INDEXDJX: .DJI

19'173.98  -913.21 (4.55%) ↓
20 Mar, 18:31 GMT-4 · Disclaimer

1 day  5 days  1 month  6 months  YTD  1 year  5 years  Max

25'962.51  21 Mar 2019

Open  High  Low
20'253.15  20'531.26  19'094.27

March 2019 to March 2020
Recap:
Mapping to a Higher Dimension

• We have seen three examples in which mapping to a higher dimension makes the problem linear.
• This idea also applies to classification.
Classification in Feature Space

- Map from $\mathbb{R}^d$ to $\mathbb{R}^D$
- Learn a linear classifier in $\mathbb{R}^D$

$$y(x) = \sigma(w^T \phi(x) + w_0)$$

$$\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$$
Polynomial Feature Expansion

$1$-Dimensional Input

$$x \rightarrow \phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

$d$-Dimensional Input

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \\ \vdots \\ x_1^M \\ \vdots \\ x_d \\ x_d^2 \\ \vdots \\ x_d^M \\ \vdots \\ x_1 x_2 \\ \vdots \\ x_1 x_d \\ \vdots \\ x_{d-1} x_d \\ x_1^2 x_2 \\ \vdots \end{bmatrix}$$

- The dimension of $\phi(x)$ grows quickly with the degree $M$ of the polynomial.
- $\phi(x)$ can be used in any algorithm that we have seen so far.
Reminder: Linear SVM

\[ w^* = \arg\min_{(w, \{\xi_n\})} \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n, \]

subject to \( \forall n, \quad t_n \cdot (\tilde{w} \cdot x_n) \geq 1 - \xi_n \) and \( \xi_n \geq 0. \)

- C is constant that controls how costly constraint violations are.
- The problem is still convex.
Polynomial SVM

\[
w^* = \arg \min_{(w, \{\xi_n\})} \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n,
\]

subject to \( \forall n, \ t_n \cdot (\tilde{w} \cdot \phi(x_n)) \geq 1 - \xi_n \) and \( \xi_n \geq 0 \).

- C is constant that controls how costly constraint violations are.
- The problem is still convex.
Interpretation

The linear decision boundary in a high dimensional space becomes a curvy one in the original low dimensional space.
Rosenbrock:

\[
\begin{align*}
 r(x, y) &= 100 \cdot (y - x^2)^2 + (1 - x)^2 \\
 f(x, y) &= \begin{cases} 
 -1 & \text{if } r(x, y) < T \\
 1 & \text{otherwise}
\end{cases}
\end{align*}
\]
5% noise

10% noise

$\left[ x, y, x^2, \ldots, xy^7, y^8 \right]$. 

5% noise

10% noise
Polynomial SVM

- A higher-degree polynomial expansion yields a more flexible boundary.
- It also increases the dimensionality of the problem.
- The computational complexity of training SVMs grows like the cube of the dimension.

→ Inherent limitation of polynomial SVMs.
Another Way to Map to a Higher Dimension

People Detection in Images

https://github.com/richaagrawa/
Training Data

- Positive data – 1208 positive window examples

- Negative data – 1218 negative window examples (initially)
Histogram of Oriented Gradients

- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024
Training and Testing

Training:
• Represent each example window by a HOG classifier.

\[
\begin{align*}
\text{Training:} & \\
& \bullet \text{Represent each example window by a HOG classifier.} \\
\text{Testing:} & \\
& \bullet \text{Train a linear classifier.}
\end{align*}
\]

\[
y(x; w, w_0) = \sigma(w^T x + w_0)
\]
Sliding Window
Non Maxima Suppression
Cover’s Theorem

A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

Geometrical and Statistical properties of systems of linear inequalities with applications, 1965

\[ N : \text{Dimension of space} \]
\[ p : \text{Number of samples} \]
\[ \frac{C(p, N)}{2^p} : \text{Percentage of separable partitions} \]
Optional: Recursive Computation

<table>
<thead>
<tr>
<th>p \ n</th>
<th>N=1</th>
<th>N=2</th>
<th>N=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p=1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>p=2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>p=3</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>p=4</td>
<td>5</td>
<td>11</td>
<td>15</td>
</tr>
</tbody>
</table>

\( \forall n \), \( C(1, n) = 2 \)

\( \forall p \), \( C(p, 1) = p + 1 \)

\( C(p, N) = C(p - 1, N) + C(p - 1, N - 1) \)

High Dimension is Good

When $N$ is large, almost all partitions are separable if the number $p$ of samples is less than $2N$. 

$$C(p, N)/2^p$$

$N = 300$
$N = 30$
$N = 3$
Problem Solved?

- Facebook or Google deal with BILLIONS of images.
- $p$ and therefore $N$ should be of that magnitude.
- Dealing with matrices of dimension $N \times N$ is impractical.
Neither Solved nor Hopeless

Bad news:

• The ratio of the number of points to the dimension must be less than 2.

• The dimension must be huge for large databases.

• As the dimension increases, the boundaries become increasingly irregular and sensitive to noise.

Good news:

• The world is structured and the points we want to classify are NOT randomly distributed.

• We can compute feature vectors that are “close” for objects that belong to the same class.
• The MNIST images are 28x28 arrays.
• They are **not** uniformly distributed in $\mathbb{R}^{784}$.
• In fact they exist on a low dimensional manifold.
• The same can be said about face images.
• And of many other things.
—> Non linear classification is a practical proposition.
Increasing the Dimension Further

Can we increase the dimension massively:
• in a principled way,
• while keeping the computational burden down?

—> Non-linear support vector machines that use the so-called kernel trick.
Reminder: Polynomial SVM

\[ w^* = \min_{(w, \{\xi_n\})} \frac{1}{2} ||w||^2 + C \sum_{n=1}^{N} \xi_n, \]

subject to \( \forall n, \quad t_n \cdot (\tilde{w} \cdot \phi(x)_n) \geq 1 - \xi_n \) and \( \xi_n \geq 0 \).

• C is constant that controls how costly constraint violations are.
Polynomial SVM w/o Slack Variables

• For simplicity

\[ \mathbf{w}^* = \min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2, \]

subject to \( \forall n, \quad t_n \cdot (\tilde{\mathbf{w}} \cdot \phi(\mathbf{x})_n) \geq 1. \)

• We will re-introduce the slack variables later.

\( \rightarrow \) Constrained optimization.
Constrained Optimization

Minimize \( f(x, y) \) subject to \( g(x, y) \leq c \).

At the constrained minimum
\[ \exists \lambda \in \mathbb{R}, \nabla f = \lambda \nabla g \]

\( \lambda \) is known as a Lagrange multiplier.

• Blue dotted lines are “level” lines.
• In this example, \( d_1 < d_2 < d_3 \).
• The blue arrows represent \( \nabla f \).
• The red arrows represent \( \nabla g \).
Lagrangian Formulation

Lagrangian:

\[ L(w, \Lambda) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} \lambda_n(t_n \tilde{w} \cdot \phi(x_n) - 1) \]

\[ \Lambda = [\lambda_1, ..., \lambda_n] \]

Theorem:

A solution of the constrained minimization problem must be such that \( L \) is minimized with respect to the components of \( w \) and maximized with respect to the Lagrange multipliers, which must remain greater or equal to zero.

Will be discussed again in the next lecture.

Revised Bishop, Chapter 7.1
Optional: Minimizing the Lagrangian

\[ L(w, \Lambda) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \lambda_n (t_n \tilde{w} \cdot \phi(x_n) - 1) \]

Setting the derivatives of \( L(w, \Lambda) \) to zero with respects to the elements of \( w \) and \( b \) yieds

\[ w = \sum_n \lambda_n t_n \phi(x_n) \]

\[ 0 = \sum_{n=1}^{N} \lambda_n t_n \]
Optional: Dual Problem

Therefore, we minimize

\[
\tilde{L}(\Lambda) = L(w, \Lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m k(x_n, x_m)
\]

subject to

\[
\lambda_n \geq 0 \quad \forall n ,
\]

\[
\sum_{n=1}^{N} \lambda_n t_n = 0
\]

and with

\[
k(x, x') = \phi(x)^T \phi(x') .
\]

\(\rightarrow\) Quadratic programming problem with \(N\) variables.

\(\rightarrow\) Complexity in \(O(N^3)\) instead of \(O(D^3)\).
Support Vectors

- Only for a subset of the data points is $\lambda_n$ is non zero.
- The are denoted by green circles.
- The corresponding $x_n$ are the support vectors and satisfy $t_n y(x_n) = 1$.
- They are the only ones that need to be considered as test time.

$\rightarrow$ That is what makes SVMs practical!
At Inference Time

- Only for a subset of the data points is $\lambda_n$ non-zero.
- The feature vector $f(x)$ does not appear explicitly anymore.
- The kernel function $k(.,.)$ can be understood as a similarity measure.

\[ y(x) = \sum_{n=1}^{N} \lambda_n t_n k(x, x_n) + b \]

\[ = \sum_{n \in S} \lambda_n t_n k(x, x_n) + b \]
The Kernel Trick

\[ y(x) = \sum_{n \in \mathcal{S}} \lambda_n t_n k(x, x_n) + b \]

\[ k(x, x') = \phi(x)^T \phi(x') \]

- \( \phi \) is implicit: In practice, we never compute it.
- We only need to compute \( k \).
- This is known as the kernel trick and is used in many different algorithms besides SVMs.
Role of the Kernel

Polynomial kernels: From small to high dimension.
Gaussian kernels: From small to infinite dimension.
Influence of the Kernel

\[ y(x) = \sum_{n=1}^{N} \lambda_n t_n k(x, x_n) + b, \]

\[ k(x, x') = 1 + (x^T x')^d \quad \text{(Polynomial terms up to degree } d). \]

\[ k(x, x') = \exp\left( -\frac{\|x - x'\|^2}{\sigma^2} \right) \quad \text{(Gaussian, feature space of infinite dimension).} \]
Back to Cover’s Theorem

\[ C(p, N) / 2^p \]

\[ p/N \]

\[ N = 300 \]
\[ N = 30 \]
\[ N = 3 \]

- Good news: Working with a Gaussian kernel virtually makes the dimension as large as the number of samples.

- Bad news: It is still not ideal for very large values of the number of points \( N \) due to the \( O(N^3) \) computational complexity.

References

Overlapping Class Distributions

- Some training examples must be allowed to be misclassified.
- Cannot satisfy all the hard constraints.
- For linear SVMs, we used slack variables to achieve this.
- For kernel SVMs, we can do so by bounding the Lagrange multipliers.

Some blues among the reds and some reds among the blues.
Optional: Dual Problem with Slack Variables

We now minimize

$$\tilde{L}(\Lambda) = L(w, \Lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m k(x_n, x_m)$$

subject to

$$\forall n, \quad 0 \leq \lambda_n \leq C,$$

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

and with

$$k(x, x') = \phi(x)^T \phi(x').$$

$$\lambda_n$$ cannot become infinite and therefore some of the constraints can be violated.

Bishop, Chapter 7.2
Finally a Simple Usable Formula

The green circles denote the support vectors.

\[ y(x) = \sum_{n \in S} \lambda_n t_n k(x, x_n) + b \]

where \( S \) is the set of support vectors.

- \( \lambda_n < C \): \( x_n \) lies on the margin.
- \( \lambda_n = C \): \( x_n \) lies inside the margin.
Non-Separable Distributions

The slack variables allow some training points to be misclassified.

- A large \( \sigma \) tends to smooth the decision boundary.
- A large \( C \) tends to minimize the number of misclassified training points.

\[ \Rightarrow \] Validation data is required to choose them properly.
Recognizing Hand-Written Digits

Test sample

0
2
4
9

Nearest neighbors

00000000
22288887
44444444
99999999
97777777
k-Nearest Neighbors vs SVM on MNIST

- Better accuracy.
- But the kernel and its parameters must be well chosen.

Knn: 96.8%
Rbf-SVM: 98.6%
SVMs in Short

• The data can be separable in a high-dimensional feature space without being separable in the input space.

• Classifiers can be learned in the feature space without having to actually perform the mapping.

• However the $O(D^3)$ or $O(N^3)$ complexity at training time makes it hard to exploit large training sets.
Reminder: SLIC Superpixels

- Superpixel segmentations with centers on a 64x64, 256x256, and 1024x1024 grid.
- Can be used to describe the image in terms of a set of small regions.
Optional: Electron Microscopy

Let us SVMs to model structures of interest!
Optional: Mitochondria Segmentation
Optional: Assigning Probabilities

- Compute image features for each superpixel.
- Train an SVM classifier to assign a probability to be within a mitochondria.
- Can be used to produce segmentations using graph-based techniques.
Optional: 3D Mitochondria

Lucchi et al. TMI’11
Here we use three classes instead of two:

- **Inside**
- **Membrane**
- **Everything else**

—>Because the inside is fully enclosed by the membranes, we can still find a global optimum.
Optional: Speeding up the Analysis

$3.21 \mu m \times 3.21 \mu m \times 1.08 \mu m$: 53 mitochondria

- By hand: 6 hours.
- Semi-automatically: 1.5 hours

$\rightarrow$ Substantial time saving for the neuroscientists.