SVM Kernels
Outline

1. Linear Separability and Feature Augmentation
2. Sample Complexity
3. Computational Complexity
4. Kernels and Nonlinear SVMs
5. Mercer’s Conditions
6. Gaussian Kernels and Support Vectors
Data Representations

• Linear separability is a property of the data *in a given representation*.
• A set that is not linearly separable. Boundary $x_2 = x_1^2$. 
Feature Transformations

- \( \mathbf{x} = (x_1, x_2) \rightarrow \mathbf{z} = (z_1, z_2) = (x_1^2, x_2) \)

- Now it is! Boundary \( z_2 = z_1 \)
Feature Augmentation

• Feature transformation:
  \( \mathbf{x} = (x_1, x_2) \rightarrow \mathbf{z} = (z_1, z_2) = (x_1^2, x_2) \)

• Problem: We don’t know the boundary!

• We cannot guess the correct transformation

• Feature *augmentation*:
  \( \mathbf{x} = (x_1, x_2) \rightarrow \mathbf{z} = (z_1, z_2, z_3) = (x_1, x_2, x_1^2) \)

• Why is this better?

• Add *many* features in the hope that some combination will help
Not Really Just a Hope!

- Add all monomials of $x_1, x_2$ up to some degree $k$
- Example: $k = 3 \Rightarrow d' = \binom{d+k}{d} = \binom{2+3}{2} = 10$ monomials
  $z = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3)$
- From Taylor’s theorem, we know that with $k$ high enough we can approximate any hypersurface by a linear combination of the features in $z$
- Issue 1: Sample complexity: More dimensions, more training data (remember the curse)
- Issue 2: Computational complexity: More features, more work
- With SVMs, we can address both issues
A Detour into Sample Complexity

• The more training samples we have, the better we generalize
• With a larger $N$, the set $T$ represents the model $p(x, y)$ better
• How to formalize this notion?
• Introduce a number $\epsilon$ that measures how far from optimal a classifier is
• The smaller $\epsilon$ we want to be, the bigger $N$ needs to be
• Easier to think about: the bigger $1/\epsilon$ ("exactitude"), the bigger $N$
• The rate of growth of $N(1/\epsilon)$ is the *sample complexity*, more or less
• Removing “more or less” requires care
Various Risks Involved

• We train a classifier on set $T$, by picking the best $h \in \mathcal{H}$:
  $$\hat{h} = \text{ERM}_T(\mathcal{H}) \in \arg \min_{h \in \mathcal{H}} L_T(h)$$

• *Empirical* risk actually achieved by $\hat{h}$:
  $$L_T(\hat{h}) = L_T(\mathcal{H}) = \min_{h \in \mathcal{H}} L_T(h)$$

• When we deploy $\hat{h}$ we want its *statistical* risk to be small
  $$L_p(\hat{h}) = \mathbb{E}_p[\ell(y, \hat{h}(x))]$$
  We can get some idea of $L_p(\hat{h})$ by testing $\hat{h}$

• Typically, $L_p(\hat{h}) > L_T(\hat{h})$

• More importantly: How small can $L_p(\hat{h})$ conceivably be?

• $L_p(\hat{h})$ is typically bigger than $L_p(\mathcal{H}) = \min_{h \in \mathcal{H}} L_p(h)$
Risk Summary

- Empirical training risk $L_T(\hat{h})$ is just a means to an end
- That’s what we minimize for training. Ignore that
- Statistical risk achieved by $\hat{h}$: $L_p(\hat{h})$
- Smallest statistical risk over all $h \in \mathcal{H}$: $L_p(\mathcal{H}) = \min_{h \in \mathcal{H}} L_p(h)$
- Obviously $L_p(\hat{h}) \geq L_p(\mathcal{H})$ (by definition of the latter)
- Typically, $L_p(\hat{h}) > L_p(\mathcal{H})$. Why?
- Because $T$ is a poor proxy for $p(x, y)$
- Also, often $L_p(\mathcal{H}) > 0$. Why?
- Because $\mathcal{H}$ may not contain a perfect $h$
- Example: Linear classifier for a non linearly-separable problem
Sample Complexity

- Typically, $L_p(\hat{h}) > L_p(H) \geq 0$
- Best we can do is $L_p(\hat{h}) = L_p(H) + \epsilon$ with small $\epsilon > 0$
- High performance (large $1/\epsilon$) requires lots of data (large $N$)
- Sample complexity measures how fast $N$ needs to grow as $1/\epsilon$ grows
- It is the rate of growth of $N(1/\epsilon)$
- Problem: $T$ is random, so even a huge $N$ might give poor performance once in a while if we have bad luck (“statistical fluke”)
- We cannot guarantee that a large $N$ yields a small $\epsilon$
- We can guarantee that this happens with high probability
Sample Complexity, Cont’d

• We can only give a probabilistic guarantee:
  • Given probability $0 < \delta < 1$ (think of this as “small”), we can guarantee that if $N$ is large enough then the probability that $L_p(\hat{h}) \geq L_p(\mathcal{H}) + \epsilon$ is less than $\delta$:

$$\mathbb{P}[L_p(\hat{h}) \geq L_p(\mathcal{H}) + \epsilon] \leq \delta$$

• The sample complexity for hypothesis space $\mathcal{H}$ is the function $N_{\mathcal{H}}(\epsilon, \delta)$ that gives the smallest $N$ for which this bound holds, regardless of model $p(x, y)$

• Tall order: Typically, we can only give asymptotic bounds for $N_{\mathcal{H}}(\epsilon, \delta)$
Sample Complexity for Linear Classifiers and SVMs

- For a binary linear classifier, the sample complexity is
  \[ \Omega \left( \frac{d + \log(1/\delta)}{\epsilon} \right) \]
  - Grows linearly with \( d \), the dimensionality of \( X \), and \( 1/\epsilon \)
  - Not too bad, this is why linear classifiers are so successful
  - SVMs with bounded data space \( X \) do even better
  - “Bounded:” Contained in a hypersphere of finite radius
  - For SVMs with bounded \( X \), the sample complexity is independent of \( d \). No curse!
  - We can augment features to our heart’s content
What About Computational Complexity?

• Remember our plan: Go from $\mathbf{x} = (x_1, x_2)$ to $\mathbf{z} = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$ in order to make the data separable
• Can we do this without paying the computational cost?
• Yes, with SVMs
SVMs and the Representer Theorem

- Recall the formulation of SVM training: Minimize

\[ f(w, \xi) = \frac{1}{2} \|w\|^2 + \gamma \sum_{n=1}^{N} \xi_n. \]

with constraints

\[ y_n(w^T x_n + b) - 1 + \xi_n \geq 0 \]
\[ \xi_n \geq 0. \]

- Representer theorem: \( w = \sum_{n \in A(w,b)} \alpha_n y_n x_n \)

\[ \|w\|^2 = w^T w = \sum_{m \in A(w,b)} \sum_{n \in A(w,b)} \alpha_m \alpha_n y_m y_n x_m^T x_n \]
Using the Representer Theorem

- Representer theorem: \( w = \sum_{n \in A(w, b)} \alpha_n y_n x_n \)
- In the constraint \( y_n (w^T x_n + b) - 1 + \xi_n \geq 0 \) we have
  \[
  w^T x_n = \sum_{m \in A(w, b)} \alpha_m y_m x_m^T x_n
  \]
- Summary: \( x \) appears in an inner product, never alone:
  \[
  \min_{w, b, \xi} \frac{1}{2} \sum_{m \in A(u)} \sum_{n \in A(u)} \alpha_m \alpha_n y_m y_n x_m^T x_n + C \sum_{n=1}^{N} \xi_n
  \]
  subject to the constraints
  \[
  y_n \left( \sum_{m \in A(u)} \alpha_m y_m x_m^T x_n + b \right) - 1 + \xi_n \geq 0
  \]
  \[
  \xi_n \geq 0
  \]
The Kernel

• Augment \( \mathbf{x} \in \mathbb{R}^d \) to \( \varphi(\mathbf{x}) \in \mathbb{R}^{d'} \), with \( d' \gg d \) (typically)

\[
\begin{align*}
\min_{\mathbf{w}, b, \xi} & \quad \frac{1}{2} \sum_{m \in \mathcal{A}(u)} \sum_{n \in \mathcal{A}(u)} \alpha_m \alpha_n y_m y_n \varphi(\mathbf{x}_m)^T \varphi(\mathbf{x}_n) + C \sum_{n=1}^N \xi_n \\
\text{subject to the constraints} & \quad y_n \left( \sum_{m \in \mathcal{A}(u)} \alpha_m y_m \varphi(\mathbf{x}_m)^T \varphi(\mathbf{x}_n) + b \right) - 1 + \xi_n \geq 0 \\
& \quad \xi_n \geq 0 .
\end{align*}
\]

• The value \( K(\mathbf{x}_m, \mathbf{x}_n) \overset{\text{def}}{=} \varphi(\mathbf{x}_m)^T \varphi(\mathbf{x}_n) \) is a \textit{number}
• The optimization algorithm needs to know only \( K(\mathbf{x}_m, \mathbf{x}_n) \), not \( \varphi(\mathbf{x}_n) \). \textit{K} is called a \textit{kernel}
Decision Rule

• Same holds for the decision rule:

\[ \hat{y} = h(x) = \text{sign}(w^T x + b) \]

becomes

\[ \hat{y} = h(x) = \text{sign} \left( \sum_{m \in A(w,b)} \alpha_m y_m x_m^T x + b \right) \]

because of the representer theorem

\[ w = \sum_{n \in A(w,b)} \alpha_n y_n x_n \]

and therefore, after feature augmentation,

\[ \hat{y} = h(x) = \text{sign} \left( \sum_{m \in A(w,b)} \alpha_m y_m \varphi(x_m)^T \varphi(x) + b \right) \]
Kernel Idea 1

- Start with some $\varphi(x)$ and use the kernel to save computation

- Example: $\varphi(x)$ =
  $$(1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$$

- Don’t know how to simplify. Try this: $\varphi(x)$ =
  $$(1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_1^2, \sqrt{6}x_1x_2, \sqrt{3}x_2^2, x_1^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, x_2^3)$$

- Can show (see notes) that
  $$K(x, z) = \varphi(x)^T \varphi(z) = (x^Tz + 1)^3$$

- Something similar works for any $d$ and $k$

- 4 products and 2 sums instead of 10 products and 9 sums

- Meager savings, but grows exponentially with $d$ and $k$, as we know
Much Better Kernel Idea 2

• Just come up with $K(x, z)$ without knowing the corresponding $\varphi(x)$
• Not just any $K$. Must behave like an inner product
• For instance, $x^T z = z^T x$ and $(x^T z)^2 \leq \|x\|^2 \|z\|^2$ (symmetry and Cauchy-Schwartz), so we need at least $K(x, z) = K(z, x)$ and $K^2(x, z) \leq K(x, x) K(z, z)$
• These conditions are necessary, but they are not sufficient
• Fortunately, there is a theory for this
Mercer’s Conditions

Mercer Conditions

- $K(\mathbf{x}, \mathbf{z}) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a kernel function if there exists $\varphi$ for which $K(\mathbf{x}, \mathbf{z}) = \varphi(\mathbf{x})^T \varphi(\mathbf{z})$

- Finite case: Given $\mathbf{x}_n \in \mathbb{R}^d$ for $n = 1, \ldots, N$ (as in $T$), a symmetric function $K(\mathbf{x}, \mathbf{z})$ is a kernel function on that set iff the $N \times N$ matrix $A = [K(\mathbf{x}_i, \mathbf{x}_j)]$ is positive semi-definite

- Problem: We would like to know if $K(\mathbf{x}, \mathbf{z})$ is a kernel for any $T$, or even for $\mathbf{x}$ we have not yet seen

- Infinite case: $K(\mathbf{x}, \mathbf{z})$ is a kernel function iff for every $f : \mathbb{R}^d \rightarrow \mathbb{R}$ s.t. $\int_{\mathbb{R}^d} f(\mathbf{x}) \, d\mathbf{x}$ is finite,
  \[ \int_{\mathbb{R}^d \times \mathbb{R}^d} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) \, d\mathbf{x} \, d\mathbf{z} \geq 0 \]

- Immediate extension of positive-definiteness to the continuous case
The “Kernel Trick”

- There is a theory for checking the Mercer conditions algorithmically (eigenfunctions instead of eigenvectors)
- There is a calculus for how to build new kernel functions
- A whole cottage industry tailors kernels to problems
- This is rather tricky. However, the Gaussian kernel is very popular

\[ K(x, z) = e^{-\frac{||x-z||^2}{\sigma^2}} \]

- A measure of similarity between \( x \) and \( z \)
- Gaussian kernels are also called Radial Basis Functions
Recall: Decision rule for SVM is \( h(x) = \text{sign}(w^T \varphi(x) + b) \) (in transformed space, where the SVM is linear).

The separating hyper-plane is \( w^T \varphi(x) + b = 0 \).

From representer theorem, \( w = \sum_n \alpha_n y_n \varphi(x_n) \) where the sum is over support vectors only.

Therefore the separating hyperplane is \( \sum_n \alpha_n y_n \varphi(x_n)^T \varphi(x) + b = 0 \).

That is, \( \sum_n \alpha_n y_n K(x_n, x) + b = 0 \).

\( x_n \) and \( x \) are in the original space.

This equation describes the decision boundary induced in the original space.
The “Kernel Trick:” Summary, Part 1

- In a linear SVM, feature vectors $\mathbf{x}$ always show up in inner products: $\mathbf{x}_m^T \mathbf{x}_n$ or $\mathbf{x}_n^T \mathbf{x}$
- If features are augmented, $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, also $\varphi(\mathbf{x})$ always shows up in inner products: $\varphi(\mathbf{x}_m)^T \varphi(\mathbf{x}_n)$ or $\varphi(\mathbf{x}_n)^T \varphi(\mathbf{x})$
- Define a kernel $K(\mathbf{x}, \mathbf{x}')$ such that there exists an (often unknown) mapping $\varphi()$ for which

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

- We always work with $K(\mathbf{x}, \mathbf{x}')$ without ever involving $\varphi(\mathbf{x})$ or $\varphi(\mathbf{x}')$ (which are large, possibly infinite)
- We avoid the computational cost of feature augmentation
The “Kernel Trick:” Summary, Part 2

Given \( K(x, x') \) to there exists a mapping \( \varphi() \) for which

\[
K(x, x') = \varphi(x)^T \varphi(x')
\]

iff \( K \) satisfies the Mercer condition:

For every \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) s.t. \( \int_{\mathbb{R}^d} f(x) \, dx \) is finite,

\[
\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} K(x, z) f(x) f(z) \, dx \, dz \geq 0
\]

This condition can be verified through eigenfunction computations

Important examples: The Radial Basis Function (RBF)

\[
K(x, x') = e^{-\frac{\|x-x'\|^2}{\sigma^2}}
\]

is a kernel

What does the decision boundary look like now?
Gaussian Kernels and Support Vectors

• The decision boundary in the original space is
  \[ \sum_n \alpha_n y_n K(x_n, x) + b = 0 \]
  where the sum is over support vectors

• For RBF SVMs,
  \[ \sum_n \alpha_n y_n e^{-\frac{||x-x_n||^2}{\sigma^2}} = -b \]

• Simple geometric interpretation
Classification

http://mldemos.b4silio.com
Regression

http://mldemos.b4sil.io.com