Lecture 8: Kernels

Introduction to Learning and Analysis of Big Data
Non-linear Classifiers

- SVM tries to find a good linear classifier on the training set.
- What if such a classifier does not exist?
- There might be a good non-linear classifier.
- Can we make this problem linearly separable?
- Change representations: Instead of $x \in \mathbb{R}$, use $\psi(x) = (x, x^2) \in \mathbb{R}^2$.

There is a linear separator in this new representation!
Feature maps

- \( \psi(x) = (x, x^2) \) maps examples from \( \mathbb{R} \) to \( \mathbb{R}^2 \).
- A linear predictor in new space is a **non-linear** predictor in original space.
- Many different **feature mappings** are possible.
- The feature mapping procedure:
  1. Given \( \mathcal{X} \) and some learning task, choose some feature mapping \( \psi : \mathcal{X} \rightarrow \mathcal{F} \).
     - \( \mathcal{F} \) is the **feature space**.
     - \( \mathcal{F} \) is usually \( \mathbb{R}^n \) for some \( n \). (later: \( n \) can also be infinite!)
  2. When given the training set \( S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \), create an image of the set in the feature space:
     \[ \hat{S} = \{ (\psi(x_1), y_1), \ldots, (\psi(x_m), y_m) \} \]
  3. Find a good linear predictor \( h_w : \mathcal{F} \rightarrow \mathcal{Y} \) on \( \hat{S} \).
     - \( w \) is a vector in the feature space \( \mathbb{R}^n \).
  4. For new examples \( x \in \mathcal{X} \), predict the label
     \[ h_w(\psi(x)) = \text{sign}(\langle w, \psi(x) \rangle). \]
The feature mapping induces a new distribution.

\[ \mathcal{D} \] is the original distribution over \( X \times Y \).

\[ \mathcal{D}^\psi := \text{the new distribution on } \mathcal{F} \times Y, \text{ induced by } \psi \text{ and } \mathcal{D}. \]

For any \( h \), \( \text{err}(h, \mathcal{D}^\psi) = \text{err}(h \circ \psi, \mathcal{D}). \)

So, a good classifier on \( \mathcal{D}^\psi \) induces a good classifier on \( \mathcal{D} \).

Which feature map should we use?

Similar to selecting a hypothesis class:

- Low approximation error: There is a good predictor for \( \mathcal{D} \) in the mapped feature space.
- Low estimation error: A learning algorithm on \( \mathcal{D}^\psi \) will find a good predictor from the training set.

E.g., if there is a linear classifier with a \textbf{large margin} in the feature space.
General-purpose feature maps

- The feature mapping is selected **before** observing the training sample!
  - Otherwise we allow any function $\rightarrow$ overfitting 😞.
- How to choose a good feature mapping?
  - Domain knowledge
  - A general-purpose feature mapping
- **Polynomial feature maps**: Allow polynomials as separators.
- For $x \in \mathbb{R}$ (one-dimensional), define $\psi : \mathbb{R} \rightarrow \mathbb{R}^{k+1}$
  \[
  \psi(x) = (1, x, x^2, \ldots, x^k).
  \]
- For $x \in \text{reals}$, can now learn any separators that are polynomials of degree $\leq k$:
  \[
  \text{sign} \left( \sum_{i=0}^{k} w_i x^i \right) = \text{sign}(\langle w, \psi(x) \rangle).
  \]
Polynomial feature maps in high dimensions

- For $x \in \mathbb{R}^d$: Allow multivariate polynomials.
  - Multivariate polynomial example:
    $$3 \cdot x(4)^6 \cdot x(16)^4 - 1.17 \cdot x(3)^{19} \cdot x(27)^{11} \cdot x(15) + 4.$$
  - $k$-degree multivariate polynomial: Each monomial’s powers sum to at most $k$.
  - Define all integer sequences of length $d$ which sum to at most $k$:
    $$I^k_d := \left\{(t_1, t_2, \ldots, t_d) \mid \sum_{j=1}^{d} t_j \leq k\right\}$$
  - The set of all multivariate polynomials of degree $k$:
    $$\left\{ \sum_{z \in I^k_d} w(z) \prod_{i=1}^{d} x(i)^{z(i)} \mid w \in \mathbb{R}^{\left|I^k_d\right|} \right\}.$$
  - $w_z$ are the coefficients of the polynomial.
Polynomial feature maps in high dimensions

- Define a feature mapping that allows all multivariate polynomials up to degree \( k \).
- Every coordinate in \( \psi(x) \) corresponds to one \( z \in I_d^k \).

\[
I_d^k := \{(t_1, t_2, \ldots, t_d) \mid \sum_{j=1}^{d} t_j \leq k\}
\]

Examples:

- \( d = 2, k = 2: \psi(x) = (1, x(1), x(2), x(1)x(2), x(1)^2, x(2)^2). \)
- \( d = 6, k = 7: \)
  - Coordinate \((3, 0, 0, 2, 2, 0)\) in \( \psi(x) \) corresponds to \( x(1)^3 \cdot x(4)^2 \cdot x(5)^2. \)
  - Coordinate \((0, 0, 5, 0, 0, 0)\) corresponds to \( x(3)^5. \)
  - If \( w \) has 7 in coordinate \((0, 0, 5, 0, 1, 0)\) and the rest is zero, the polynomial represented by \( w \) is \( 7 \cdot x(3)^5 \cdot x(5). \)
Learning with feature maps

- Feature maps can be in a very high dimension.
  - Polynomial map in degree $k$: If $\mathcal{X} = \mathbb{R}^d$, number of features in $\mathcal{F} = \mathbb{R}^n$, needs to be $n \approx k^d$.
  - E.g. text documents with the bag-of-words representation: $d = 100,000$ (English words), $k = 10$, $n$ = enormous.

- Two problems:
  - High dimension $\implies$ need a lot of training examples for ERM.
    - Solution: use a large-margin algorithm such as SVM.
  - High dimension $\implies$ heavy computation and memory consumption.
    - Even representing examples and $w$ in memory can be impossible!
  - Solution for computational issues: kernels.
The Kernel Trick

- If $\psi$ maps to a very high dimension, it can be computationally expensive to use the representations $\psi(x_i)$ directly.

- Define $K(x, x') := \langle \psi(x), \psi(x') \rangle$.

- The function $K$ is also called a kernel.

- The Kernel Trick: Many learning algorithms for linear separators can be implemented using only the function $K(x, x')$, and never $\psi(x)$.

- Advantage: $\psi(x)$ can be huge-dimensional or even infinite-dimensional, while $K(x, x')$ might be easy to compute.
The Kernel Trick

- How can the algorithm find and output $w$ when the dimension is huge or infinite?
- We need an alternative representation for $w$.
- Claim: The hard/soft SVM objectives on $\psi$ can be rewritten as:

$$\text{Minimize } f(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_m) \rangle) + R(\|w\|),$$

where $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is a function, and $R : \mathbb{R}_+ \rightarrow \mathbb{R}$ is a monotonic non-decreasing function.

- SVM objectives:
  - Soft SVM:
    $$\text{Minimize } \lambda \|w\|^2 + \ell^h(w, \hat{S}) \equiv$$
    $$\text{Minimize } \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, \psi(x_i) \rangle\}.$$  
    Define $R(a) := \lambda a^2$ and $f(a_1, \ldots, a_m) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i a_i\}$.

  - Hard SVM: Minimize $\|w\|^2$ s.t. $\forall i, y_i \langle w, \psi(x_i) \rangle \geq 1$.
    Define $R(a) := a^2$ and
    $$f(a_1, \ldots, a_m) := \begin{cases} 0 & \forall i, y_i a_i \geq 1, \\ \infty & \text{otherwise.} \end{cases}$$

- There are other SVM-like algorithms with the same form.
The representer theorem

- Many popular objectives on $\psi$ can be rewritten as:

\[
\text{Minimize } f(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_m) \rangle) + R(\|w\|),
\]

where $f: \mathbb{R}^m \rightarrow \mathbb{R}$ is a function, and 
$R: \mathbb{R}_+ \rightarrow \mathbb{R}$ is a monotonic non-decreasing function.

Theorem (The representer theorem)

If $\psi$ maps the data to $\mathbb{R}^n$ for some $n$, and the objective has the form above, then the optimal solution to the objective can be written as

\[
w = \sum_{i=1}^{m} \alpha(i)\psi(x_i), \quad \text{where } \alpha = (\alpha(1), \ldots, \alpha(m)) \in \mathbb{R}^m.
\]

- This allows kernel-SVM algorithms to represent $w$ using $\alpha(1), \ldots, \alpha(m)$ instead of $w(1), \ldots, w(n)$.
- If $n \gg m$, this is a huge improvement!
- $x_i$ with $\alpha(i) \neq 0$ are the support vectors of $w$. 
The representer theorem

Objective: Minimize \( f(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_m) \rangle) + R(\|w\|) \)
where \( f : \mathbb{R}^m \to \mathbb{R} \) is a function, and
\( R : \mathbb{R}_+ \to \mathbb{R} \) is a monotonic non-decreasing function.

Need to show that there is an optimal solution with the form: \( w = \sum_{i=1}^{m} \alpha(i)\psi(x_i) \).

Note: \( w \in \mathbb{R}^n \), the feature space.

Proof.

Let \( w^* \) be the optimal solution to the objective.

\( w^* \in \mathbb{R}^n \) and so are \( \psi(x_i) \), so we can write: \( w^* = \sum_{i=1}^{m} \alpha(i)\psi(x_i) + u \), where \( \forall i \leq m, \langle u, \psi(x_i) \rangle = 0 \).

Let \( w := w^* - u \). Note \( w = \sum_{i=1}^{m} \alpha(i)\psi(x_i) \).

Then \( \|w^*\|^2 = \|w\|^2 + \|u\|^2 + 2\langle w, u \rangle \).

\( \langle w, u \rangle = \langle \sum_{i=1}^{m} \alpha(i)\psi(x_i), u \rangle = 0 \). So \( \|w^*\|^2 = \|w\|^2 + \|u\|^2 \).

Therefore \( \|w\| \leq \|w^*\| \), so \( R(\|w\|) \leq R(\|w^*\|) \).

For all \( i \), \( \langle w, \psi(x_i) \rangle = \langle w^* - u, \psi(x_i) \rangle = \langle w^*, \psi(x_i) \rangle \).

Therefore, \( f(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_m) \rangle) = f(\langle w^*, \psi(x_1) \rangle, \ldots, \langle w^*, \psi(x_m) \rangle) \).

So \( w \) is also an optimal solution, and it has the wanted representation.
Rewriting the objective

- **Objective:** Minimize \( f(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_m) \rangle) + R(\|w\|) \)

- **Solution** has the form \( w = \sum_{i=1}^{m} \alpha(i)\psi(x_i) \) for some vector \( \alpha \in \mathbb{R}^m \).

- To avoid representing \( w \) directly, rewrite everything using \( \alpha \) instead:
  \[
  \langle w, \psi(x_i) \rangle = \left\langle \sum_{j=1}^{m} \alpha(j)\psi(x_j), \psi(x_i) \right\rangle = \sum_{j=1}^{m} \alpha(j) \langle \psi(x_j), \psi(x_i) \rangle
  \]
  \[
  \|w\|^2 = \left\langle \sum_{j=1}^{m} \alpha(j)\psi(x_j), \sum_{j=1}^{m} \alpha(j)\psi(x_j) \right\rangle = \sum_{i,j=1}^{m} \alpha(i) \alpha(j) \langle \psi(x_i), \psi(x_j) \rangle
  \]

- Recall \( K(x, x') := \langle \psi(x), \psi(x') \rangle \).

- So, we can rewrite the objective as:
  \[
  \text{Minimize } f \left( \sum_{j=1}^{m} \alpha(j)K(x_j, x_1), \ldots, \sum_{j=1}^{m} \alpha(j)K(x_j, x_m) \right) + R \left( \sum_{i,j=1}^{m} \alpha(i) \alpha(j)K(x_i, x_j) \right).
  \]
Solving the objective using the kernel function

- **Objective:** Minimize $f(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_m) \rangle) + R(\| w \|)$

- Rewriting the objective using the kernel function:

  $$
  \text{Minimize } f \left( \sum_{j=1}^{m} \alpha(j) K(x_j, x_1), \ldots, \sum_{j=1}^{m} \alpha(j) K(x_j, x_m) \right) + R \left( \sqrt{\sum_{i,j=1}^{m} \alpha(i) \alpha(j) K(x_i, x_j)} \right).
  $$

- To solve this objective, the algorithm needs **only the values of** $K(x_i, x_j)$ **for all** $i, j \leq m$.

- The algorithm never needs to see the examples $x_i$ or to calculate $\psi$.

- The kernel values form an $m \times m$ matrix called the **Gram matrix**.

- The gram matrix is $G$, where $G_{i,j} = K(x_i, x_j)$.
Example: Kernel soft-SVM

- The soft-SVM objective for the feature mapping $\psi$:

$$\text{Minimize } \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i$$

s.t. $\forall i, y_i \langle w, \psi(x_i) \rangle \geq 1 - \xi_i$, and $\xi_i \geq 0$.

- How is it implemented using a kernel?
- Recall: $\langle w, \psi(x_i) \rangle = \sum_{j=1}^{m} \alpha(j) K(x_j, x_i)$, $\|w\|^2 = \sum_{i,j=1}^{m} \alpha(i) \alpha(j) K(x_i, x_j)$.

Kernel Soft-SVM

**input**  The training sample Gram matrix $G \in \mathbb{R}^{m \times m}$, the training labels $y_1, \ldots, y_m$, parameter $\lambda > 0$.

**output** $\alpha \in \mathbb{R}^m$

1: Find $w$ that solves the following problem:

$$\text{Minimize } \lambda \alpha^T G \alpha + \frac{1}{m} \sum_{i=1}^{m} \xi_i$$

s.t. $\forall i, y_i \langle \alpha, G[i] \rangle \geq 1 - \xi_i$, and $\xi_i \geq 0$.

($G[i]$ is row $i$ of $G$)

2: Return $\alpha$. 
The output of a kernel algorithm

- The output of a kernel algorithm is $\alpha \in \mathbb{R}^m$ instead of $w$.
- How do we use this to predict new labels?
- Solution is $w = \sum_{i=1}^{m} \alpha(i)\psi(x_i)$.
- For any example $x$,

$$h_w(\psi(x)) = \text{sign}(\langle w, \psi(x) \rangle) = \text{sign} \left( \sum_{j=1}^{m} \alpha(j)K(x_j, x) \right).$$

- Calculate $K(x_1, x), \ldots, K(x_m, x)$, and use $\alpha$ to get the label $h_w(\psi(x))$. 

•
Calculating the kernel function

- Kernel algorithms never need to represent $x$ or $w$.
- But to run them we need to provide $K(x_i, x_j)$.
- Also, to predict using $\alpha$ we need $K(x_i, x)$.
- A simple option to calculate $K$: use $K(x, x') = \langle \psi(x), \psi(x') \rangle$.

**Problem:** $\psi$ could be huge dimensional, or even infinite dimensional!

**Solution:** There are several useful functions $K$ (kernels) which can be calculated without representing $\psi$. 
Polynomial kernels

The $k$-degree polynomial kernel is

$$K(x, x') = (1 + \langle x, x' \rangle)^k.$$ 

- **Claim**: This is a kernel function.
- In other words: There exists a $\psi$ such that $K(x, x') = \langle \psi(x), \psi(x') \rangle$.

**Proof (first part)**

- An example vector is $x = (x(1), \ldots, x(d))$. Denote $x(0) = 1$.
- Then $1 + \langle x, x' \rangle = \sum_{j=0}^{d} x(j)x'(j)$.
- The polynomial kernel is:

$$K(x, x') = (1 + \langle x, x' \rangle)^k = \left( \sum_{i=0}^{d} x(j)x'(j) \right)^k$$

- The set of possible $k$-tuples is: $\{0, \ldots, d\}^k$.

$$K(x, x') = \sum_{z \in \{0, \ldots, d\}^k} \prod_{i=1}^{k} x(z(i)) \cdot x'(z(i)) = \sum_{z \in \{0, \ldots, d\}^k} \left( \prod_{i=1}^{k} x(z(i)) \right) \left( \prod_{i=1}^{k} x'(z(i)) \right).$$
Polynomial kernels

The \( k \)-degree polynomial kernel is

\[
K(x, x') = (1 + \langle x, x' \rangle)^k.
\]

Claim: This is a kernel function.

In other words: There exists a \( \psi \) such that \( K(x, x') = \langle \psi(x), \psi(x') \rangle \).

Proof (Second part).

The polynomial kernel is:

\[
K(x, x') = \sum_{z \in \{0, \ldots, d\}^k} \left( \prod_{i=1}^k x(z(i)) \right) \left( \prod_{i=1}^k x'(z(i)) \right).
\]

Define \( \psi : \mathbb{R}^d \to \mathbb{R}^{(d+1)^k} \):

- Map the \( k \)-tuples \( z \in \{0, \ldots, d\}^k \) to coordinates in \( \mathbb{R}^{(d+1)^k} \).
- Coordinate \( z \) of \( \psi(x) \) is \( \prod_{i=1}^k x(z(i)) \).

Then

\[
K(x, x') = \langle \psi(x), \psi(x') \rangle.
\]
Polynomial kernels

The $k$-degree polynomial kernel is

$$K(x, x') = (1 + \langle x, x' \rangle)^k.$$  

- We showed that this is a kernel function.
- The coordinates in the feature map are $\prod_{i=1}^{k} x(z(i))$ for all $z \in \{0, \ldots, d\}^k$.
- Every monomial of degree up to $k$ has a coordinate in the feature map.
- For degrees smaller than $k$, use $z$ with several 0's.
- This representation causes many duplicate coordinates.
- E.g. $(1, 2, 1)$ and $(2, 1, 1)$ give the same value $x(1)^2 \cdot x(2)$.
- The identical coordinates can be “collapsed” to a single coordinate (it will have a coefficient which is the number of duplicates).
Polynomial Kernels

- The polynomial kernel $K(x, x') = (1 + \langle x, x' \rangle)^k$:

- Any **multivariate polynomial** in $x$ of degree up to $k$ is a **linear function** in the feature map that matches $K$.

- So, kernel SVM with the polynomial kernel looks for a predictor on $\mathcal{X}$ which is defined by a polynomial.

- Without the **kernel trick**, need to calculate $\psi$, which would cost $O(d^k)$. 😞

- Using the kernel trick, only calculate $K(x, x')$, which costs $O(d)$. 😊
The Gaussian kernel with a parameter $\sigma > 0$ is $K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma}}$.

- **Claim**: This is a kernel function.
- **Proof** for $\mathcal{X} = \mathbb{R}$ (one dimensional examples):
  - Define $\psi(x)$ such that the $n$'th coordinate of $\psi(x)$ is: $\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2\sigma}} \cdot (x/\sqrt{\sigma})^n$.
  - **Allow an infinite dimension**: all coordinates in $\{0, 1, 2, \ldots\}$.
  - A “nice” infinite-dimensional inner-product space is called a **Hilbert space**.
  - Then
    $$\langle \psi(x), \psi(x') \rangle = \sum_{n=0}^{\infty} \left( \frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2\sigma}} \cdot (x/\sqrt{\sigma})^n \right) \cdot \left( \frac{1}{\sqrt{n!}} e^{-\frac{x'^2}{2\sigma}} \cdot (x'/\sqrt{\sigma})^n \right)$$
    $$= e^{-\frac{x^2}{2\sigma}} e^{-\frac{x'^2}{2\sigma}} \sum_{n=0}^{\infty} \frac{(xx')^n}{\sigma^n n!}$$
    $$= e^{-\frac{x^2-x'^2}{2\sigma}} \cdot xx'/\sigma = e^{-(x-x')^2/2\sigma} = e^{-\|x-x'\|^2/2\sigma}.$$
  - If $\mathcal{X} = \mathbb{R}^d$: $\psi(x)$ has a coordinate for every sequence $z$ of integers in $\{1, \ldots, d\}$ of any length $n$.
  - The coordinate $z$ (for $z$ of size $n$) is $\frac{1}{\sqrt{n!}} e^{-\frac{\|x\|^2}{2\sigma}} \cdot \prod_{i=1}^{n} x(z(i))/\sigma$. 
Gaussian kernels

The Gaussian kernel with a parameter $\sigma > 0$ is $K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma}}$.

- The coordinate $z$ is $\frac{1}{\sqrt{n!}} e^{-\frac{\|x\|^2}{2\sigma}} \cdot \prod_{i=1}^{n} x(z(i))/\sigma$.
- So, every multivariate polynomial (of any degree!) can be used as a linear predictor with this kernel.
- What does the predictor look like in the original space?

$$h_w(\psi(x)) = \text{sign}(\langle w, \psi(x) \rangle) = \text{sign} \left( \sum_{j=1}^{m} \alpha(j) K(x_j, x) \right) = \text{sign} \left( \sum_{j=1}^{m} \alpha(j) e^{-\frac{\|x-x_j\|^2}{2\sigma}} \right).$$

- Predict the label of $x$ based on the distance of $x$ from each of the $x_j$ with $\alpha(j) \neq 0$.
- If $x$ is close enough to $x_j$, sign of $\alpha(j)$ determines label of $x$.
- $\sigma \approx$ “radius of influence” of each $x_j$.
- Can illustrate the predictor as “circles” around some of the $x_j$’s.

- A trade-off in the size of $\sigma$:
  - Small $\sigma$: Can have a very fine-tuned boundary. Larger estimation error
  - Large $\sigma$: Function is more smooth. Larger approximation error
- $\sigma$ is usually selected using cross validation.
- Gaussian kernel is also called Radial Basis Function (RBF) kernel.
Using kernels to encode a hypothesis class

- If we have prior knowledge on the problem, we might know which hypotheses make sense.
- Use a kernel to convert hypotheses to linear predictors.

Example: Learn to identify executable files that contain a virus

- Viruses are identified by a signature: a substring in the executable file.
- \( \mathcal{X} \) is the set of all possible files of length at most \( n \): strings in \( \Sigma^{\leq n} \), where \( \Sigma \) is the alphabet.
- We would like to find the most predictive substring in the file:
  - For each substring \( \nu \), \( h_\nu(x) \) is positive iff \( \nu \) is a substring of \( x \).
  - The hypothesis class: \( \mathcal{H} = \{ h_\nu \mid \nu \in \Sigma^{\leq k} \} \).
- How to make \( \mathcal{H} \) into a set of linear predictors?
Using kernels to encode a hypothesis class

- $h_v(x)$ is positive iff $v$ is a substring of $x$, $\mathcal{H} = \{ h_v \mid v \in \Sigma^*, |v| \leq k \}$.

- The feature space $\mathcal{F} = \mathbb{R}^d$:
  - A feature $v$ for every possible string $|v| \leq k$.
  - $\psi(x)(v) = \mathbb{I}[v \text{ is a substring of } x]$.
  - An extra feature 0 that is always 1 (for bias): $\psi(x)(0) = 1$.

- $w_v \in \mathbb{R}^d$: A vector with $w_v(v) = 2$, $w_v(0) = -1$, other coordinates are zero.

- $h_v(x) = \text{sign}(\langle w_v, \psi(x) \rangle)$.

- $d$ (number of features) is exponential in $k$: $d = \Theta(|\Sigma|^k)$.

- But we have
  \[
  K(x, x') = 1 + \sum_{v \in \Sigma^*, |v| \leq k} \mathbb{I}[v \text{ is a substring of } x] \cdot \mathbb{I}[v \text{ is a substring of } x']
  = 1 + \text{number of common substrings of } x, x' \text{ of length } \leq k.
  \]

- $K(x, x')$ can be calculated in $O(n^2k^2)$. 
Using kernels to encode a hypothesis class

- We saw that this kernel can be efficiently calculated.
- Computational complexity: Good.
- What about the sample complexity?
  - Sample complexity of SVM is $O(\min(d, 1/\gamma^2_*))$.
  - Dimension is huge: $d = \Theta(|\Sigma|^k)$.
  - Let’s check the margin.
  - Suppose there is a single substring $v$ that separates the input.
    
    \[ w = (-1, 0, \ldots, 0, 2, 0, \ldots, 0) \]
    
    \[
    \gamma_D(w) := "\text{the largest value such that } \mathbb{P}_{(X,Y) \sim D} \left[ \frac{1}{R_D} \frac{|\langle w, \psi(X) \rangle|}{\|w\|} \geq \gamma_D(w) \right] = 1"
    \]
    
    \[
    R_D := "\text{the smallest value such that } \mathbb{P}[\|\psi(X)\| \leq R_D] = 1" \leq \sqrt{1 + nk}.
    \]
    
    \[
    \forall x, |\langle w, x \rangle| = 1, \|w\| = \sqrt{5} \implies \gamma_* \geq \gamma_D(w) \geq \frac{1}{\sqrt{5nk}}.
    \]

- Conclusion: The sample complexity is $O\left(\frac{1}{\gamma_*^2}\right) \leq O(nk)$.
- Note: the kernel-SVM uses a richer class than the original $\mathcal{H}$. 

Kernels: Summary

- Feature maps allow using SVM to learn non-linear classifiers.
- Kernel functions can be used to represent large and infinite dimensional feature maps.
- The kernel trick: run SVM without representing $\psi$ directly.
- If the kernel can be calculated efficiently, this can improve computational complexity.
- The sample complexity will depend mainly on the margin.
- There are general-purpose kernels, such as
  - Polynomial kernel
  - Gaussian kernel
- Special-purpose kernels can encode a specific hypothesis class.
- This can work even if the examples are not vectors at all.