Nice properties: convex, theoretically motivated, nonlinear with kernels.

[Bose, Guyon, Vapnik '92; Corres & Vapnik '95]
\[ f(x, a) = \text{classifier}(x, a) \]

\[ P(x', y) = \text{disjointness} \quad \text{if} \quad y \in C(x') \]

\[ \forall \epsilon > 0, \, \exists \delta > 0 \quad \text{such that} \quad \text{observation} \]

\[ \text{Hypothesis: } \text{Suppose } \]
\[ \text{loss} \]
\[ \text{Empirical Risk: } \sum_{i=1}^{n} \mathbb{I}(y_i \neq f(x_i, \theta)) / n \]
\[ \text{What if we know } P(x_1, y_1) \approx \text{(e.g., uniform)} \]
\[ \mathbb{E} \left( \sum_{i=1}^{n} \mathbb{I}(y_i \neq f(x_i, \theta)) / n \right) \]
\[ \mathbb{E} (x) = \int \int_{x \in X} \left( \frac{1}{2} \left| y - f(x, \theta) \right| \right) P(x \mid y) \, dx 
\]
\[ \mathbb{E} (x) = \sqrt{\int_{x \in X} \left( \frac{1}{2} \left| y - f(x, \theta) \right| \right)^2 P(x \mid y) \, dx} \]
\[ \text{Expected Risk:} \]
\[ \text{VC Confidence} = \sqrt{\text{log}(2e/\epsilon) + 1} - \text{log}(1/\delta) \]

Testing \( \geq \) Training + Confidence

\[ R(x) \geq R^\text{emp}(x) + \sqrt{\text{VC Confidence}} \]

(Vapnik, 1995)

Risk Bound

(1 - \( \eta \))

Now, suppose loss = 1 / categorical
If we know \( A \) — the complete bound term

winds by the set \( \mathcal{F}(A) \).

That can be represented in all possible

is the maximum number of points

see: VC dimension for a set \( \mathcal{F}(A) \).

\( \mathcal{A} = \text{ VC dimension} \)

VC dimension
In general, for $f(x)$ = hyperplanes in $\mathbb{R}^m$, $\text{VCdim} = m+1$.

Example: $f(x)$ = line in $\mathbb{R}^2$
won't get low training error.

If you take a very simple set of models, you have low complexity, but

training error. But you might "overfit".

If you take a high capacity set of functions (explain a lot) you get low

regularization.

The complexity function is often called a measure of capacity. Actually, a lot of bounds of this form have been proved (different

\[ \text{complexity of classifier} \leq \text{true risk} + \text{test risk} \]
Images taken from a talk by B. Scholkopf.

\[ \text{Compressive of } f(x) \]
How to choose the complexity?
\[ X^* = \text{support vectors} \]

\[ \text{minimizing } \|m\| \implies \text{equivalent to obtaining a large margin classifier} \]

\[ \text{minimizing } \|m\| \text{ and have low capacity} \implies \]

\[ \text{minimizing } \|m\| \]

\[ \text{where } R \text{ is the radius of the smallest sphere around the origin containing } X^* \]

\[ R^2 \geq \eta \]

\[ \forall \eta \text{ has a VC dimension satisfying } \exists X \text{ such that } \forall \eta \geq \|m\| \text{ such that } \]

\[ \text{The set of decision functions } f_{\text{sign}} = \text{sign}(x \cdot m) \text{ defined on } X^* \]

\[ \text{Consider hyperplanes } (x \cdot m) = 0 \text{ where } m \text{ is normalized wrt a set of Vapnik \& Chervonenkis also showed the following:} \]

\[ \text{Coverability as Hyperplanes} \]
\[ f(x) = mx + b \Rightarrow -1 \leq 0 \]

A: Solve for subject to \( \frac{||w||}{1} \leq \frac{||y||}{2} \)

Problem: We can minimize \( ||y||^2 \)

So, to maximize the variance.

\[ \frac{||w||}{1} - b \leq 1 \quad \text{H} \]

Plane: \( b = \frac{-b}{w} \)

For: \( \frac{||w||}{1} - b \leq 1 \)

\( 1 - b < b \Rightarrow 1 < 2b \)

\( 1 < 2b \Rightarrow b > \frac{1}{2} \)

Linearly separable case

\( \text{SUM} \)
\[
L = \begin{cases}
\frac{1}{2} x^T \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} x - \frac{1}{2} x^T \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 1 & \text{if } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \Delta
\\
\infty & \text{otherwise}
\end{cases}
\]

\text{minimize } L

\text{subject to}

\begin{align*}
& x_1 \geq 0, \quad x_2 \geq 0, \\
& x_1 + x_2 = 1
\end{align*}

\text{Convex Quadratic Programming Problem}

\[\begin{aligned}
& \text{minimize } \frac{1}{2} x^T P x + \begin{bmatrix} q_1 & q_2 \end{bmatrix} x \\
& \text{subject to } A x = b
\end{aligned}\]
to cases:

\[ C = \text{Geese} \]

\[ \frac{1}{2} \|w\|^2 + C (\frac{1}{3} \|x\|^2 + 1) - 1 \leq 0 \]

subject to

\[ \frac{1}{3} \|x\|^2 + C (\frac{1}{3} \|w\|^2) \]

Program:

we can minimize \( \|w\|^2 \)

So, to maximize the margin:

\[ A \|w\|^2 + \|x\|^2 \]

H2 plane

4

\[ \frac{1}{3} + 1 - q + m \leq 0 \]

\[ m \geq \frac{1}{3} - 1 \leq q + m \leq \frac{1}{3} - 1 \]

\[ A \|w\|^2 \]

\[ \text{Linearly Non-Separable Case} \]
\[
L \leq \frac{1}{2} x_{e} - \frac{1}{2} \leq 1 \quad \text{subject to } 0 \leq x_{i} \leq c, \quad \sum_{i=1}^{n} x_{i} = m - \frac{1}{2} x_{h}.
\]

\[
L = \frac{1}{\|w\|^2} \quad \text{minimize } L
\]

Formulation (unconstrained)
Linear classifier

Other solution: Map data into a richer feature space, including non-linear features, then use a

Heuristics needed to train; etc.

Problems: Local minima; many parameters;

Classifiers (neurons): a neural network

One solution: Creating a net of simple linear

+ This formulation only deals with vectorial data

Noisy data

Non-linearly separable data

Linear classifiers cannot deal with
\[ q + (x)\Phi \cdot m = (x)f \]

\( \forall i \) learn the map from \( (x)\Phi \) to \( (x)f \)

\( (x)\Phi \leftarrow x \)

Formally, preprocess the data with:

Constrain a hyperplane in the space so all other equations are the same.

Map data into a richer feature space including nonlinear features, then

Linear classifiers aren't complex enough sometimes. SVM solution:
\[(\xi x, x^1 x (\xi)) \land (x^1 x) =: (\xi z, z x, x) \leftarrow (z x, x)\]

\[\xi \mathcal{H} \leftarrow \zeta \mathcal{H} : \Phi\]
And a generalization theory problem (curse of
very large vectors)

There is a computational problem (working with
functions)

Solves the problem of expressing complex
working in high dimensional feature spaces

(\text{dimensionality})

\text{Generalized: } 
R(x,y) = (x \cdot y)^p, \quad p \in [0,1), \quad \forall x, y \in \mathbb{R}^d

\text{For } p = 2r \Rightarrow \psi = \frac{1}{2}, \quad T = 183, 181, 376

\text{Result: } x = 256 \notin p = 4, \quad \forall \psi \in [0, \frac{1}{2}] \text{ where } 2 = (p+1)
The kernel function is:

\[
q(x) \Phi \cdot (x)^{\top} \Phi = (x)^{\top} \Phi \Phi^{\top} = (x)^{\top} P
\]

The decision rule is now:

\[
\text{optimize } \omega
\]

for some variables \( \omega \). Instead of optimizing \( \omega \) directly we can thus:

\[
(\Phi) = \sum_{i=1}^{m} \omega_i
\]

SVMs as a special case:

The Representer Theorem (Kimeldorf & Wahba, 1971) shows that for:

The representability in memory, and hard for the GP to solve.

Problem: the dimensionality of \( \Phi(x) \) can be very large, making \( \omega \) hard to

\[\text{linear map} \Phi : R \rightarrow \text{R^2}\]
Before looking at SVMs and kernel methods in general, it is useful to analyze the Perceptron algorithm.

The Perceptron Algorithm (Rosenblatt, 57)

- Is a hyperplane in input space
- Simpler case: classification, decision function
- Great for software engineering (and for analysis)

- Any K-B algorithm can be fitted with any kernel
- Any kernel-based learning algorithm composed of two modules:
  - A problem-specific kernel function
  - A general purpose learning machine

KERNEL - CLASSIFIERS
Need to compute $Y(x_i, x)$ for all $SVS x_i$.

\[
\begin{align*}
(x_i^T x)_j Y_{j,k} & \sum_{i=1}^{0<Y_{j,k}} + 0_{j,k} \sum_{i=1}^{Y_{j,k}} \sin = f_i \\
\end{align*}
\]

The classifier:

Need to compute the kernel matrix for the training data.

\[
\begin{align*}
\left\{ (x_i^T x)_j Y_{j,k} \sum_{i=1}^{0<Y_{j,k}} + 0_{j,k} \sum_{i=1}^{Y_{j,k}} \sin \right\} \max \\
\end{align*}
\]

The optimization problem:

Replace dot products in the SVM formulation with kernel values.

The kernel trick.
We can write the expansion explicitly, by concatenating powers up to \( d \) and multiplying by appropriate weights.

\[
K(x, z) = \tanh(c + x^T z)^d.
\]

This leads to the original, linear SVM.

\[
K(x, z) = x^T z.
\]

The linear kernel:

Popular kernels:
To compute, whereas the explicit representation may not fit in memory.

Interestingly, if $p$ is large the kernel is still only requires $n$ multiplications.

\[
\tilde{z} \cdot \tilde{z} \cdot x^\dagger x \gamma \tilde{z} + \tilde{z} \cdot x^\dagger x \gamma \tilde{z}^\dagger x + \gamma \tilde{z} \cdot x^\dagger x = \gamma (\tilde{z} \cdot x^\dagger x + x \gamma \tilde{z}^\dagger x) = \\
\gamma ((\tilde{z} \cdot x^\dagger x) \cdot (\tilde{z} \cdot x^\dagger x)) = \gamma (x \cdot x) = (x^\dagger x)^\gamma
\]

is the same as:

\[
\tilde{z} \cdot \tilde{z} \cdot x^\dagger x \gamma \tilde{z} + \tilde{z} \cdot x^\dagger x \gamma \tilde{z} \cdot x^\dagger x + \gamma \tilde{z} \cdot x^\dagger x = \\
(\tilde{z} \cdot x^\dagger x \gamma \tilde{z} \cdot x^\dagger x) \cdot (\tilde{z} \cdot x^\dagger x \gamma \tilde{z} \cdot x^\dagger x) = (\tilde{z} \cdot x^\dagger x) \Phi \cdot (\tilde{z} \cdot x^\dagger x) \Phi
\]

\[
(\tilde{z} \cdot x^\dagger x \gamma \tilde{z} \cdot x^\dagger x) = (\tilde{z} \cdot x^\dagger x) \gamma \Phi \Phi \gamma \tilde{z} \cdot x^\dagger x
\]

The kernel gives the same result as the explicit polynomial kernel:

\[p(x \cdot x) = (x \cdot x)^\gamma\]
Compare to the effect of model order in regression or logistic regression.

(∞ > C) using $C$.

- $8^{th}$ order polynomial
- $4^{th}$ order polynomial
- $2^{nd}$ order polynomial
- Linear
Using this one can get state-of-the-art results.

\[ q + (\frac{1}{\sigma}||x - \hat{x}||^2) \exp \left( \frac{1}{\sigma} \right) \sum_{w} = (x,f) \]

Popular kernel functions. It adds a "bump" around each data point.

The RBF kernel $K$ is one of the most popular Kernel functions.
\[
\begin{align*}
\exp\left(-\frac{1}{2}||x||^2\right) e^{xp(x)}(\frac{1}{2} ||y||^2) e^{xp(y)} &= \exp\left(-\frac{1}{2}||x+y||^2\right) e^{xp(x+y)} e^{xp(-\frac{1}{2}||x-y||^2)} \\
&= \exp\left(-\frac{1}{2}||x+y||^2\right) e^{xp(x+y)} = K(x,y) \exp\left(-\frac{1}{2}||x||^2\right) e^{xp(x)}
\end{align*}
\]

Now, if \( x \neq y \) then \( K(x,y) \neq 0 \) \( \forall x,y \in \mathbb{R}^n \). Let \( \mathcal{E}(x) = (\mathcal{E}_1, \ldots, \mathcal{E}_n) \) be an embedded \( \mathcal{E}(x) \). Indeed, \( \mathcal{E}(x) \) could be

\[
\begin{align*}
\mathcal{E}(x) &= \sum_{i=1}^{n} e_i (x_i - \bar{x}_i) \\
&= \sum_{i=1}^{n} (x_i - \bar{x}_i) (\mathbf{e}_i - \bar{\mathbf{e}})
\end{align*}
\]

So, if \( x \neq y \), then \( x < x' < y < y' < x \) implies

\[
\mathcal{E}(x') = (\mathcal{E}_1', \ldots, \mathcal{E}_n') = (\mathcal{E}_1 + \frac{1}{2}p, \ldots, \mathcal{E}_n + \frac{1}{2}p - 1)
\]

and so \( x \) is mapped by \( \mathcal{E}(x) \)
\[ x \in \mathbb{R} \quad \text{and} \quad y \in \mathbb{R} \]

So, if we map vectors \( x \) and \( y \) to \( \mathbb{R}^2 \),

\[
\mathbf{u}_n = \sum_{n=0}^{\infty} \frac{(2\pi i)^n}{n!} \mathbf{u} \exp\left(-\frac{1}{2} \|\mathbf{u}\|^2\right)
\]

Dueing Fourier Series
Polynomial SVM has around 1% test error.
Linear SVM has around 8.5% test error.
60'000 training examples, 10'000 test examples, 28x28.
For example MNIST hand-writing recognition.
Choosing a good mapping $\phi(\cdot)$ (encoding prior knowledge + getting right complexity of function class) for your problem improves results.

<table>
<thead>
<tr>
<th>%</th>
<th>Translation invariant SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.56%</td>
<td>Boosted Lenet</td>
</tr>
<tr>
<td>0.7%</td>
<td>Lenet</td>
</tr>
<tr>
<td>1.1%</td>
<td>Tangent distance</td>
</tr>
<tr>
<td>1.1%</td>
<td>RBF-SVM</td>
</tr>
<tr>
<td>1.4%</td>
<td>3-nearest-neighbor</td>
</tr>
<tr>
<td>2.4%</td>
<td>Linear</td>
</tr>
<tr>
<td>8.4%</td>
<td>Classifier</td>
</tr>
</tbody>
</table>

Test Error

Compute $\Delta$ Test Results
All examples become SVS if likely overfitting.

\[ \begin{cases} x \neq x \iff 0 \\ x = x \iff 1 \end{cases} \leftarrow (\varphi; x, x) y \]

What is the role of parameter \( \varphi \)? Consider \( \varphi = 0 \).

The feature space is infinite-dimensional.

The RBF kernel is a measure of similarity between two examples.

\[ \left( \sqrt{\|z - x\|^2 + \frac{\varphi^2}{2}} \right) \exp \left( -\frac{\|z - x\|^2}{\varphi^2} \right) = (\varphi; z, x) y \]

Radial basis function kernel
Why do some SVSs here appear to be far from the boundary?

Instead of simply evaluating the RBF kernel, we don't need to explicitly compute dot products in the feature space. Data are linearly separable in the (infinite-dimensional) feature space.

 SVM with RBF (Gaussian) kernels.
The kernel matrix must be positive definite:

\[
\begin{bmatrix}
(Nx, Nx) & (N x, N x) & \cdots & (N x, N x) \\
\cdots & \cdots & \cdots & \cdots \\
(N x, 1 x) & (N x, 1 x) & \cdots & (N x, 1 x)
\end{bmatrix}
\]

- non-negative: for any \( x_i \), \( nx \) is a non-negative real number.
- symmetric: \( (x, z)^T K = (z, x)^T K \)
- continuous:

Mercer's theorem due to Mercer (1909): \( K \) must be a valid kernel, i.e., such that there exists a feature space \( \phi \) in which \( K(x, z) = (\phi(x), \phi(z))^T \).

What kind of function \( K \) is a valid kernel?
can make complex kernels from simple ones:  

- etc etc etc...  

- $a_k + bk'$ is a kernel, for $a, b > 0$  

- $ck'$ is a kernel, if $c > 0$  

- $k + k'$ is a kernel  

- The set of kernels is closed under some operations. If $k$, $k'$ are kernels, then:  

-Making Kernels
In $K(\cdot, \cdot)$ is a kernel so is

\[ K(x, y) = a K_1(x, y) + b K_2(x, y) \]

\[ K(x, y) = a K_1(y, x) + b K_2(y, x) \]

\[ K(x, y) + b K_2(x, y) \]
The ideal kernel

\[
\begin{array}{cccccc}
\downarrow & \ldots & \downarrow & \downarrow & \downarrow \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\downarrow & \ldots & \downarrow & \downarrow & \downarrow \\
\downarrow & \ldots & \downarrow & \downarrow & \downarrow \\
\end{array}
\]

\[K = \]