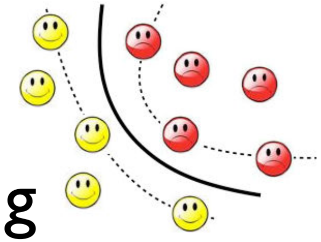


Machine Learning and Data Mining

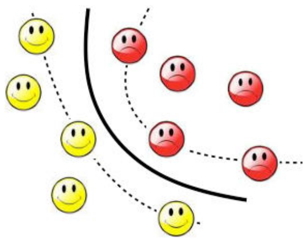
(COMP 5318)

Support Vector Machines

Nguyen Hoang Tran

Announcements

- This lecture is based on:
  - Murphy's book: Chapters 8, 14
  - Ullman's book: Chapter 12

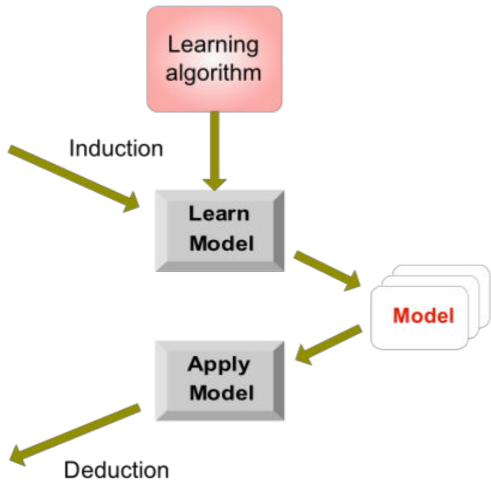


Quick Review

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



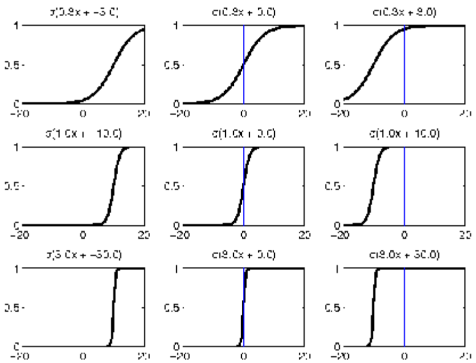
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Logistic Regression

- Discriminative model for binary classification

$$\begin{aligned} p(y|\mathbf{x},\mathbf{w}) &= \mathrm{Ber}(y|\sigma(\eta)) = \sigma(\eta)^y(1-\sigma(\eta))^{1-y} \\ \eta &= \mathbf{w}^T\mathbf{x} \\ \sigma(\eta) &\overset{\text{def}}{=} \frac{1}{1+\exp(-\eta)} = \frac{e^\eta}{e^\eta+1} \end{aligned}$$



Sigmoid or Logistic function

Logistic Regression

- Assumes a parametric form for directly estimating P(Y | X). For binary concepts, this is:

$$P(Y=0|X) = \frac{1}{1+\exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

$$P(Y=1|X) = 1 - P(Y=0|X)$$

$$= \frac{\exp(w_0 + \sum_{i=1}^n w_i X_i)}{1+\exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

- Equivalent to a one-layer backpropagation neural net.
- Logistic regression is the source of the sigmoid function used in backpropagation.
- Objective function for training is somewhat different.

Logistic Regression as a Log-Linear Model

- Logistic regression is basically a linear model, which is demonstrated by taking logs.

$$\text{Assign label } Y=0 \text{ iff } 1 < \frac{P(Y=0\,|\,X)}{P(Y=1\,|\,X)}$$

$$1 > \exp(w_0 + \sum_{i=1}^n w_i X_i)$$

$$0 > w_0 + \sum_{i=1}^n w_i X_i$$

- Also called a **maximum entropy model** (**MaxEnt**) because it can be shown that standard training for logistic regression gives the distribution with maximum entropy that is consistent with the training data.

Loss for density estimation

- Suppose output is $\hat{p}(y|\mathbf{x})$, truth is $p(y|\mathbf{x})$
- Use KL (Kullback-Leibler) divergence

$$L(p(y|\mathbf{x}), \hat{p}(y|\mathbf{x})) = KL(p(y|\mathbf{x}), \hat{p}(y|\mathbf{x})) = \sum_y p(y|\mathbf{x}) \log \frac{p(y|\mathbf{x})}{\hat{p}(y|\mathbf{x})}$$

- Risk is expected negative log likelihood

$$R(\hat{p}) = -E_{\mathbf{x}} \sum_y p(y|\mathbf{x}) \log \hat{p}(y|\mathbf{x}) = -E_{\mathbf{x},y} \log \hat{p}(y|\mathbf{x})$$

Logistic Regression Training

- Weights are set during training to maximise the **conditional data likelihood** :

$$W \leftarrow \underset{W}{\operatorname{argmax}} \prod_{d \in D} P(Y^d \mid X^d, W)$$

where D is the set of training examples and Y^d and X^d denote, respectively, the values of Y and X for example d.

- Equivalently viewed as maximising the **conditional log likelihood** (CLL)

$$W \leftarrow \underset{W}{\operatorname{argmax}} \sum_{d \in D} \ln P(Y^d \mid X^d, W)$$

Preventing Overfitting in Logistic Regression

- To prevent overfitting, one can use **regularisation** (a.k.a. smoothing) by penalising large weights by changing the training objective:

$$W \leftarrow \underset{W}{\operatorname{argmax}} \sum_{d \in D} \ln P(Y^d \mid X^d, W) - \frac{\lambda}{2}\|W\|^2$$

Where λ is a constant that determines the amount of smoothing

- This can be shown to be equivalent to assuming a Gaussian prior for W with zero mean and a variance related to $1/\lambda$.

Relation Between NB and Logistic Regression

- Naïve Bayes with Gaussian distributions for features (GNB), can be shown to give the same functional form for the conditional distribution $P(Y \mid X)$.
  - But converse is not true, so Logistic Regression makes a weaker assumption.
- Logistic regression is a **discriminative** rather than generative model, since it models the conditional distribution $P(Y \mid X)$ and directly attempts to fit the training data for predicting Y from X. Does not specify a full joint distribution.

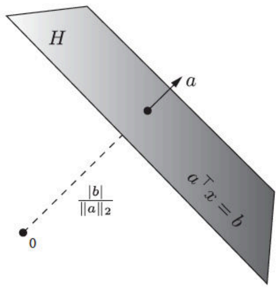
Support Vector Machines

Hyperplanes and Halfspaces

- A hyperplane in $\mathbb{R}^n$ is a set of the form

$$H = \left\{ x \in \mathbb{R}^n \, : \, a^\top x = b \right\},$$

where $a \in \mathbb{R}^n$, $a \neq 0$, and $b \in \mathbb{R}$ are given.

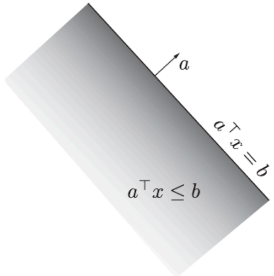


Hyperplanes and Halfspaces

- An hyperplane H separates the whole space in two regions:

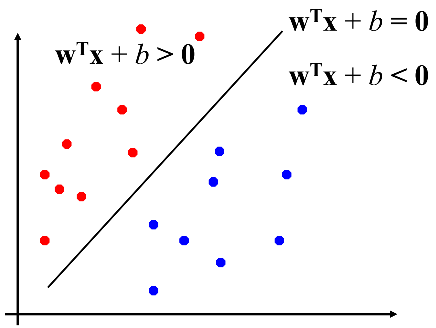
$$H_- = \left\{ x \, : \, a^\top x \leq b \right\}, \quad H_{++} = \left\{ x \, : \, a^\top x > b \right\}.$$

- These regions are called halfspaces (H_- is a closed halfspace, H_{++} is an open halfspace).
- the halfspace H_- is the region delimited by the hyperplane $H = \{a^\top x = b\}$ and lying in the direction opposite to vector a. Similarly, the halfspace H_{++} is the region lying above (i.e., in the direction of a) the hyperplane.



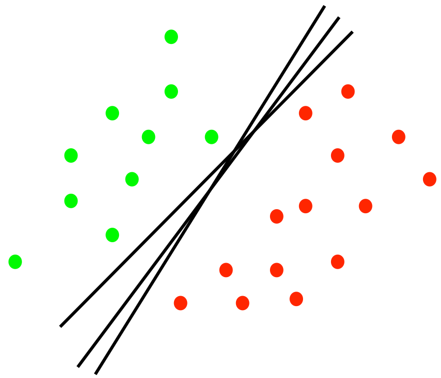
Linear Classifiers

- Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^\mathrm{T}\mathbf{x} + b)$$

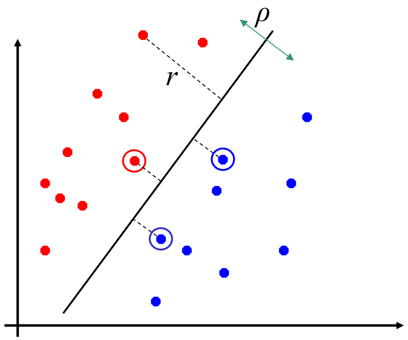
Linear classifiers



Which line is better?

Classification Margin

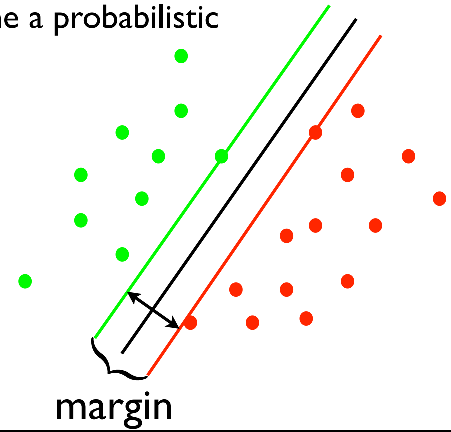
- Distance from example $\mathbf{x}_i$ to the separator is $r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



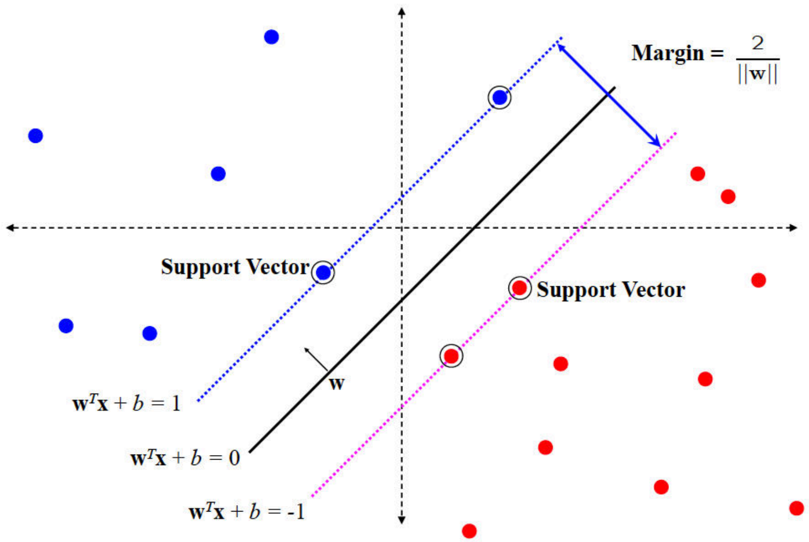
Maximum margin

Things to note

- The boundary is defined by only a few points (support vectors)
- Difficult to define a probabilistic explanation



SVM



Linear SVM Mathematically

- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{array}{ll} \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq \rho/2$$

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is $r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$

- Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, i=1..n : $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, i=1..n : $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, i=1..n : $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal (original) problem:

Find $\alpha_1 \dots \alpha_n$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- Given a solution $\alpha_1 \dots \alpha_n$ to the dual problem, solution to the primal is:

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

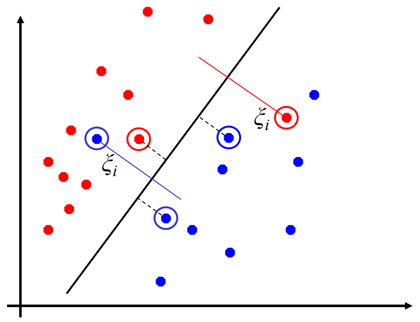
- Each non-zero α_i indicates that corresponding $\mathbf{x}_i$ is a support vector.
- Then the classifying function is (note that we don't need $\mathbf{w}$ explicitly):

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^T\mathbf{x}_j$ between all training points.

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

- The old formulation:

> Find **w** and b such that
> $\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
> and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Modified formulation incorporates slack variables:

> Find **w** and b such that
> $\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized
> and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

- Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification – Solution

- Dual problem is identical to separable case (would *not* be identical if the 2-norm penalty for slack variables $C\Sigma\xi_i^2$ was used in primal objective, we would need additional Lagrange multipliers for slack variables):

> Find $\alpha_1 \ldots \alpha_N$ such that
> $\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
> (1) $\Sigma\alpha_i y_i = 0$
> (2) $0 \leq \alpha_i \leq C$ for all α_i

- Again, $\mathbf{x}_i$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

> $\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$
> $b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x}_k$ for any k s.t. $\alpha_k > 0$

Again, we don't need to compute **w** explicitly for classification:

> $f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$

Linear SVMs: Overview

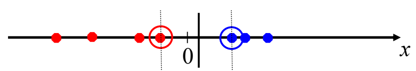
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

> Find $\alpha_1 \ldots \alpha_N$ such that
> $\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x}_i^T\mathbf{x}_j}$ is maximized and
> (1) $\Sigma\alpha_i y_i = 0$
> (2) $0 \leq \alpha_i \leq C$ for all α_i

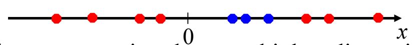
> $f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x}_i^T\mathbf{x}} + b$

Non-linear SVMs

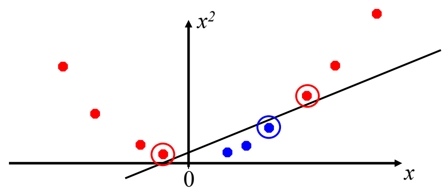
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

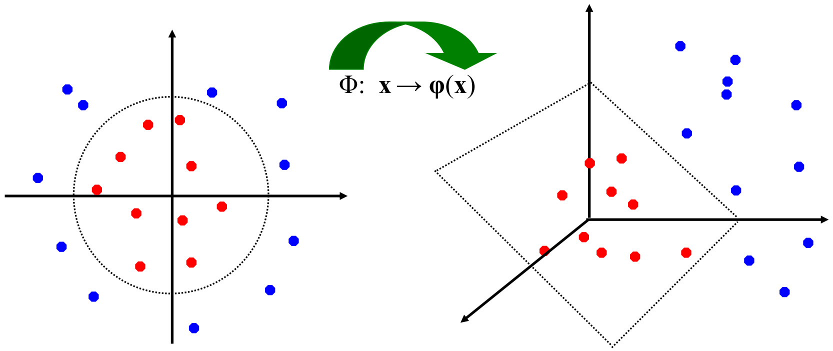


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$$

- A *kernel function* is a function that is equivalent to an inner product in some feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^{\mathbf{T}} [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}] =$

  $= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$, where $\boldsymbol{\varphi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

- Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

What Functions are Kernels?

- For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathbf{T}}\varphi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem:

  Every semi-positive definite symmetric function is a kernel

- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
…	…	…	…	…
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	…	$K(\mathbf{x}_n,\mathbf{x}_n)$

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself
- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions
- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.
- Higher-dimensional space still has *intrinsic* dimensionality d (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Non-linear SVMs Mathematically

- Dual problem formulation:

Find $\alpha_1 \dots \alpha_n$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- The solution is:

$f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$

- Optimization techniques for finding α_i's remain the same!

SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i's at a time, e.g. SMO [Platt '99] and [Joachims '99]
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is

Overfitting

- Huge feature space with kernels, what about overfitting???
  - Maximising margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

Summary

- SVM advantages
  - Efficient QP learning
  - Sparse
- SVM disadvantages
  - Not probabilistic
  - Do not directly handle multi-class problems
  - QP might struggle in very large datasets