Support Vector Machines

Mehul Motani
Electrical & Computer Engineering
National University of Singapore
Email: motani@nus.edu.sg

AI and Machine Learning

- "AI is the new electricity. Just as 100 years ago electricity transformed industry after industry, AI will now do the same." – Andrew Ng
- Machine learning is about learning from data.
- Supervised learning – Learning from data with labels which serve a supervisory purpose
- Unsupervised learning – Learning from data without labels allows tasks such a clustering.
- Reinforcement learning – Learning from data without labels but there is feedback from the environment.

Support Vector Machines (SVM)

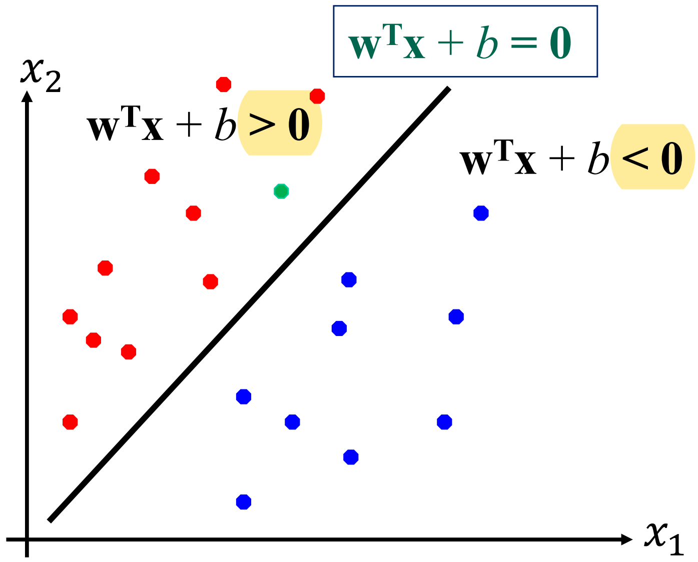
- SVM is a supervised learning algorithm
  - Useful for both classification and regression problems
- Linear SVM – Maximum-Margin Classifier
  - Formalize notion of the <u>best</u> linear separator
- Optimization Problem with Lagrangian Multipliers
  - Technique to solve a constrained optimization problem
- Nonlinear SVM – Extending Linear SVM with Kernels
  - Project data into higher-dimensional space to make it linearly separable.
  - create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes.
  - Complexity: Depends only on the number of training examples, not on dimensionality of the kernel space!

SVM – A Brief History

- Pre-1980: Almost all learning methods learned linear decision surfaces.
  - Linear learning methods have nice theoretical properties
- 1980's: Decision trees and Neural Nets allowed efficient learning of non-linear decision surfaces
  - Little theoretical basis and all suffer from local minima
- 1990's: Efficient learning algorithms for non-linear functions based on computational learning theory developed
- Support Vector Machines
  - The original SVM algorithm was invented by Vapnik and Chervonenkis in 1963.
  - Nonlinear SVMs using the kernel trick were first introduced in a conference paper by Boser, Guyon and Vapnik in 1992.
  - The SVM with soft margin was proposed by Cortes and Vapnik in 1993 and published in 1995.

Supervised Learning: Linear Separators

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



- Two features: x_1 and x_2
- Two classes: **red** and **blue**
- Linear separator given by line:

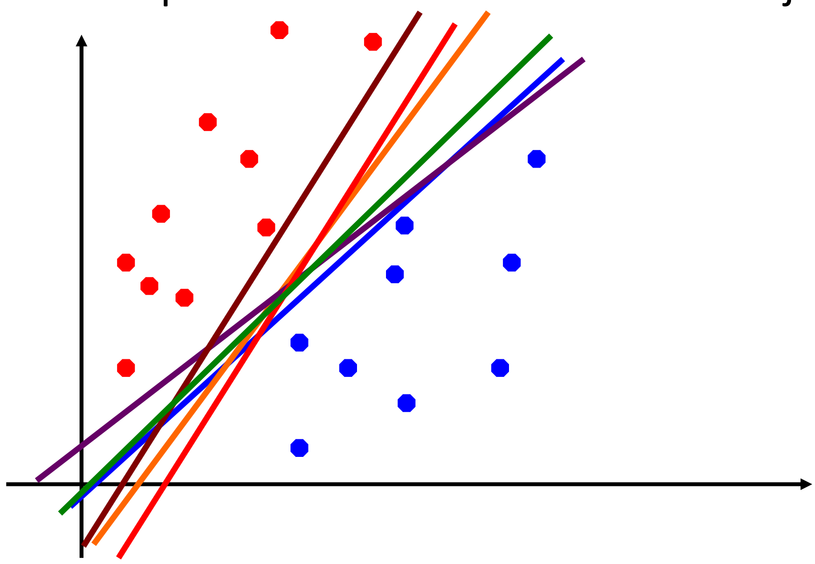
$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0 \qquad (1)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

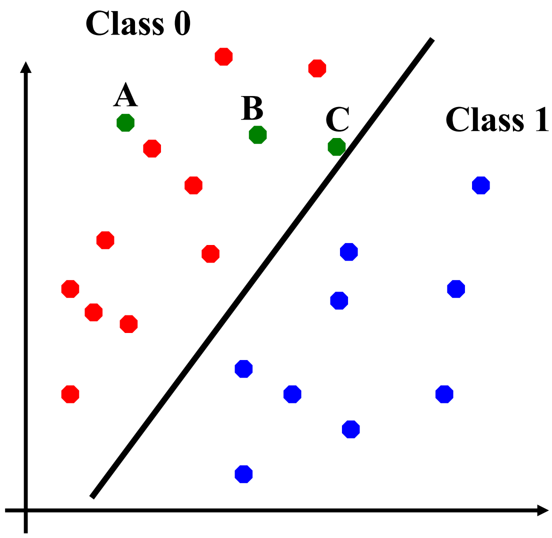
- Classification
  - $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b < 0 \rightarrow$ **blue (-1)**
  - $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b > 0 \rightarrow$ **red (+1)**
- Classifier function
  - $f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$
- New data: **Green dot** will be classified as **red class (+1)**

Linear Separators

- There are many possible linear separators!
- Which of the linear separators is optimal?
- The linear SVM solution defines an objective and finds the linear separator which maximizes that objective.



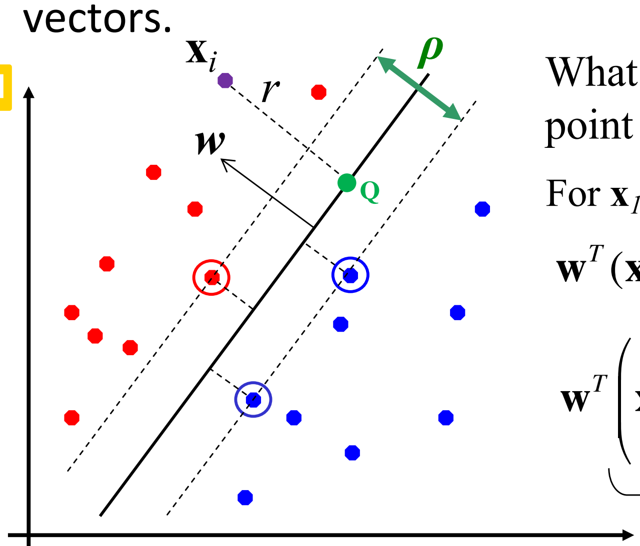
Linear Separators and Margin



- Consider three new data points: A, B, C (green dots), all of which are classified as class 0.
- How confident are you that point A is class 0?
- What about point C?
- What about point B?
- Intuitively, we are more confident about point A than point C.
- Intuition: if a point is far from the separating hyperplane (i.e., large margin), then we may be more confident in our prediction.

Classification Margin

- Data points closest to the hyperplane are called the ***support vectors*** (circled data points)
- ***Margin*** ρ of separator is the distance between support vectors
- Note that the separator is completely defined by its support vectors.



What is the distance, r, from data point $\mathbf{x}_i$ to the separator?

For $\mathbf{x}_1$ and $\mathbf{x}_2$ on the separating hyperplane:

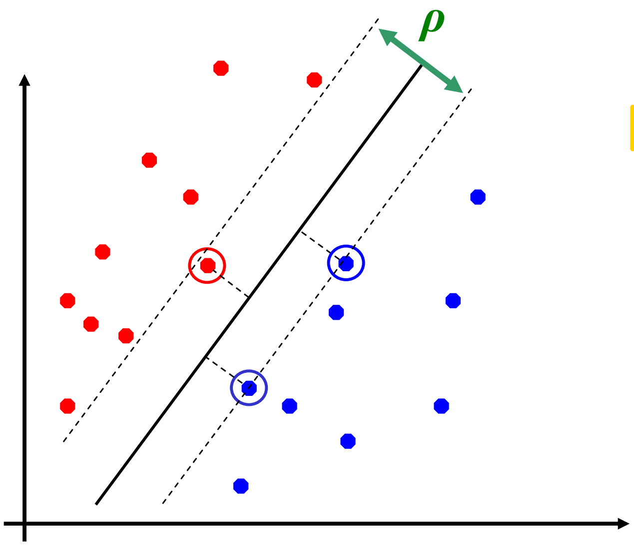
$$\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0 \Rightarrow \mathbf{w} \perp hyperplane \qquad (1)$$

$$\mathbf{w}^T \underbrace{\left(\mathbf{x}_i - r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right)}_{\text{Point Q}} + b = 0 \Rightarrow r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|} \qquad (2)$$

Norm of **w**

Maximum Margin Classification

- Maximizing the margin is provably good and intuitive
  - Larger margin leads to lower generalization error (Vapnik).
- Implies that only support vectors matter; other training examples can be ignored → SVM is stable and robust to outliers



Linear SVM Mathematically

- Let training set be $S=\{(\mathbf{x}_i, y_i)\}_{i=1,2,\ldots,n}$ with $\mathbf{x}_i \in \mathbf{R}^d$ and $y_i \in \{-1, 1\}$.
- Suppose we have a separating hyperplane with margin ρ, weight vector **w** and scalar b.
- Then for each training example $(\mathbf{x}_i, y_i)$:

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

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling **w** and *b* by *ρ/2* in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \qquad \textbf{(2)}$$

- Then the margin can be expressed through (rescaled) **w** and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|} \qquad \textbf{(3)}$$

Linear SVMs Mathematically (cont.)

- Then we can formulate the quadratic optimization problem:

Find $\mathbf{w}$ and b such that $\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized	**(a)**	**(1)**
and for all $(\mathbf{x}_i, y_i) \in S$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$	**(b)**	

- We can reformulate the problem in (1) as follows:

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

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

and for all $(\mathbf{x}_i, y_i) \in S$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

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

- 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.
- Solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal problem:

Dual: 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)**
(2) $\alpha_i \geq 0$ for all α_i

See https://en.wikipedia.org/wiki/Quadratic_programming

The Optimization Problem Solution

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

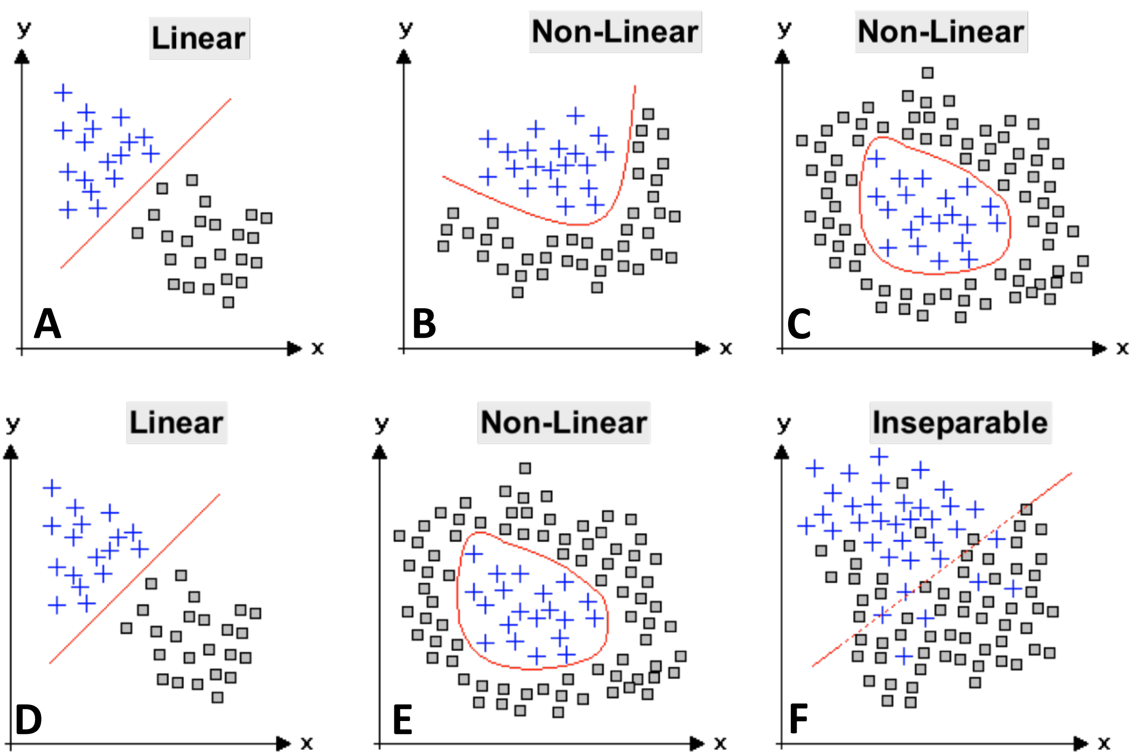
$$\boxed{\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma\alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0} \qquad \textbf{(1)}$$

- 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 **w** explicitly):

$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b} \qquad \textbf{(2)}$$

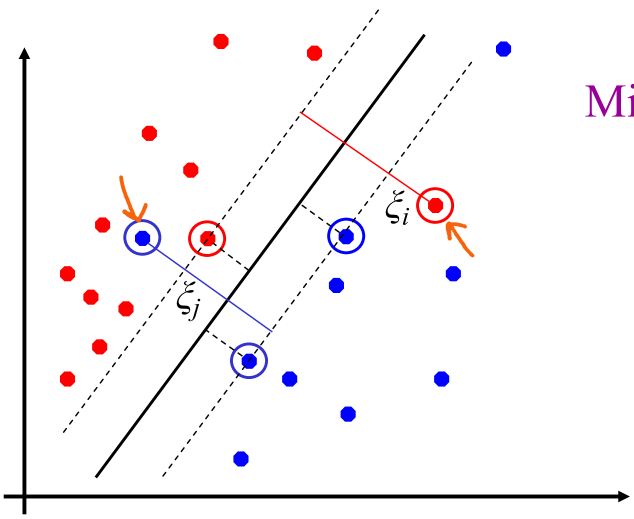
- The quantity $\mathbf{x}^{\mathrm{T}}\mathbf{y}$ is called the <u>inner product</u> or dot product between the vector **x** and the vector **y**.
- Notice that the solution relies on the inner product between the test point **x** and the support vectors $\mathbf{x}_i$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$ between all training points.

Linear and nonlinear data models



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*.



What should our quadratic optimization criterion be?

Minimize:

$$\underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{w}}_{\text{Maximize Margin}} + \underbrace{C\sum_{k=1}^{R}\xi_k}_{\text{Misclassification Penalty}}$$

Note: that ξ is the Greek letter Xi and is pronounced as ‘zai’ or ‘ksi’.

Hard margin vs Soft margin

- The hard-margin SVM formulation:

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

(1)

- Modified soft-margin SVM formulation with slack variables:

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

(2)

- 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.
  - Larger C → the more the penalty for misclassifications. This leads to smaller and smaller margins but less misclassifications. This is essentially overfitting.
  - Small C → the lower the penalty for misclassifications. This leads to larger margins but more misclassifications.

Soft Margin Classification – Solution

- Dual problem is identical to separable case:

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

(1)

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

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

(2)

Note: We don't need to compute **w** explicitly for classification:

Note: 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…

Theoretical Justification for Maximum Margins

- VC dimension is a measure of the complexity of a classifier. The more complex the classifier, the more prone it is to overfitting.
- Vapnik proved the following:

The class of optimal linear separators has VC dimension h bounded from above as

$$h \leq \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\right\} + 1 \quad \textbf{(1)}$$

where ρ is the margin, D is the diameter of the smallest sphere that can enclose all of the training examples, and m_0 is the dimensionality.

- Intuitively, this implies that regardless of dimensionality m_0 we can minimize the VC dimension by maximizing the margin ρ.
- Thus, complexity of the classifier is kept small regardless of dimensionality.

Summary of Linear SVMs

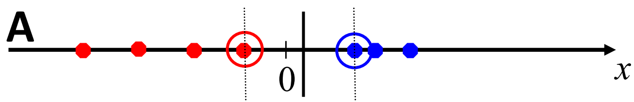
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, the 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 **(1)**

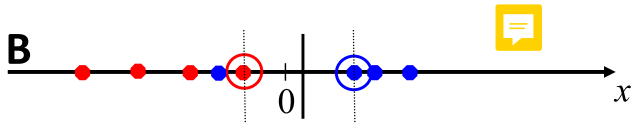
$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x}_i^T\mathbf{x}} + b$ **(2)**

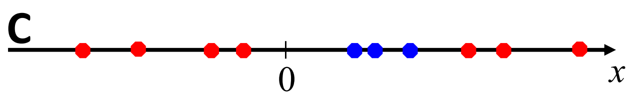
Non-linear SVMs

Consider this noisy dataset:



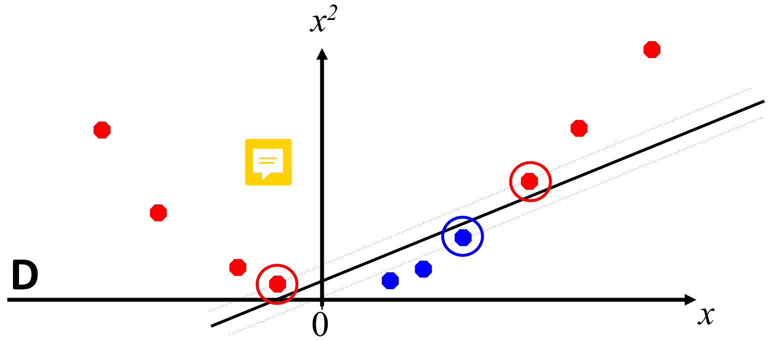
How about this dataset?





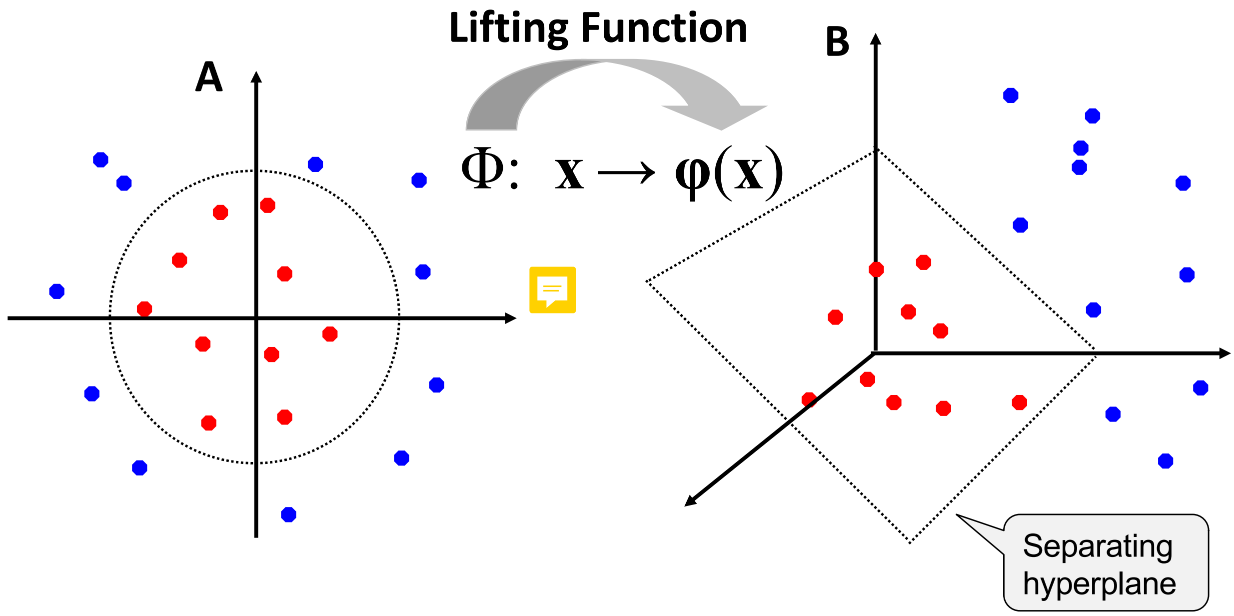
But what are we going to do if the dataset is just too hard?

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



Non-linear SVMs: Feature spaces

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



The "Kernel Trick"

- The linear SVM classifier relies on the inner product between vectors, for example: $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
- 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 higher dimensional feature space.
- Example: 2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]^{\mathbf{T}}$
  - Let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2$ **(1)**
  - Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$ for some $\boldsymbol{\varphi}(\mathbf{x})$

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$
$$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]\,[1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]^{\mathbf{T}} \quad \textbf{(2)}$$
$$\rightarrow K(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j), \text{ where } \boldsymbol{\varphi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]^{\mathbf{T}} \quad \textbf{(3)}$$

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

$$K=\begin{array}{|c|c|c|c|c|}\hline K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \dots & K(\mathbf{x}_1,\mathbf{x}_n) \\ \hline K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_n) \\ \hline & & & & \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline K(\mathbf{x}_n,\mathbf{x}_1) & K(\mathbf{x}_n,\mathbf{x}_2) & K(\mathbf{x}_n,\mathbf{x}_3) & \dots & K(\mathbf{x}_n,\mathbf{x}_n) \\ \hline \end{array} \qquad \textbf{(1)}$$

Check out the discussion at: https://www.quora.com/What-is-the-kernel-trick

Examples of Kernel Functions

1. 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

2. 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, where d is the original feature space dimension.

3. 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.

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

(1)

- The solution is:

$$f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b \qquad \textbf{(2)}$$

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

Nonlinear SVM - Summary

- In summary, linear SVM locates a separating hyperplane in the feature space and classifies points in that space
- Nonlinear SVM lifts the problem to a higher dimensional space and performs linear SVM in the higher dimensional space.
- This corresponds to a nonlinear separator in the original feature space.
- The algorithm does not need to represent the space explicitly, it does this by simply defining a kernel function, which plays the role of the inner product in the high dimensional feature space.

Properties of SVM

- Sparseness of solution when dealing with large data sets as only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces as the complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Mathematically nice – a simple convex optimization problem which is guaranteed to converge to a single global solution
- Supported by theory and intuition
- SVM empirically works very well
  - Text (and hypertext) categorization, image classification,
  - Protein classification, Disease classification
  - Hand-written character recognition

Weakness of SVM

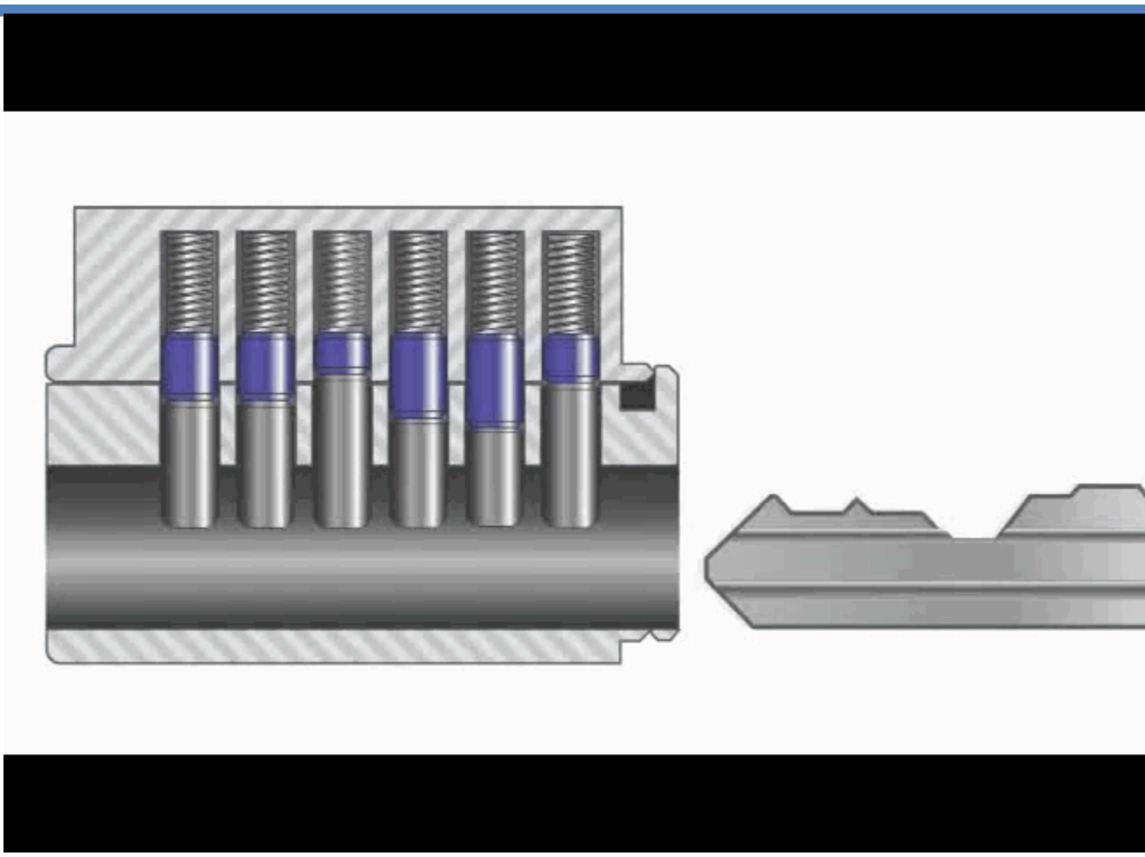
- SVM is sensitive to noise

  - A relatively small number of mislabeled examples can dramatically decrease the performance
- Standard SVM only considers two classes
- Question: How to do multi-class classification with SVM?
- Answer: Build multiple SVMs
  1. With m classes, learn m SVM's
     - SVM 1 learns "Output = 1" vs "Output != 1"
     - SVM 2 learns "Output = 2" vs "Output != 2"
     - :
     - SVM m learns "Output = m" vs "Output != m"
  2. To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

SVM Summary

- 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.
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.
- Some references on VC-dimension and Support Vector Machines:
  - C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.
  - The VC/SRM/SVM Bible: Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience, 1998
  - https://en.wikipedia.org/wiki/Support_vector_machine

What do data engineers and thieves have in common?



Thank you!

- Please send me your feedback and any questions you may have.
- The best way to contact me is via email:

  mehul.motani@gmail.com
- Thanks for listening!