BITS Pilani
Pilani Campus

Machine Learning
DSECL ZG565

Support Vector Machines –II

- Dr. Monali Mavani

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Tom Mitchell, Prof.. Burges, Prof. Andrew Moore and many others who made their course materials freely available online

Topics to be covered

- Soft Margin SVM
- Nonlinear SVM
- Kernel Trick
- SVM Kernels
- Multi-Class Problem
- SVM vs Logistic Regression
- SVM Applications

Linear SVMs: Overview

- **The classifier is a *separating hyperplane.***
- **Most "important" training points are support vectors; they define the decision hyperplane (unlike other algorithms like linear regression, ANN etc. where all training points determine decision hyperplane)**
- **Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.**

Find $\alpha_1 \ldots \alpha_N$ such that

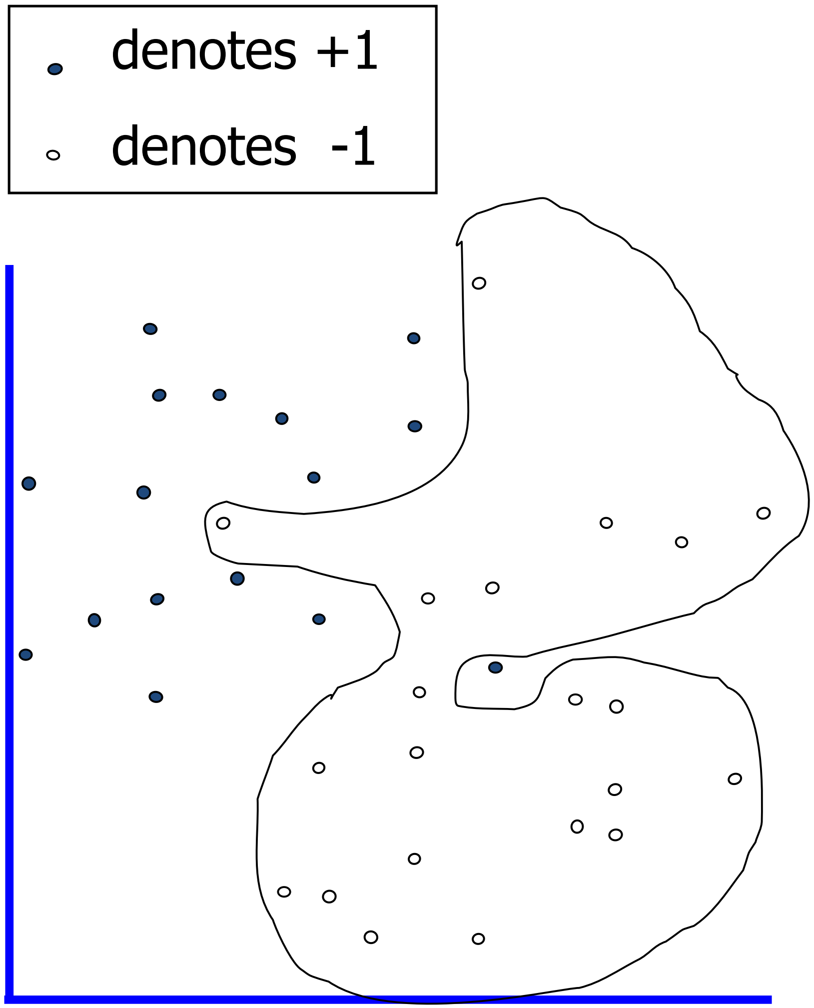
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}\right)$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\boldsymbol{\alpha_i \geq 0}$ **for all** $\boldsymbol{\alpha_i}$

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

Dataset with noise



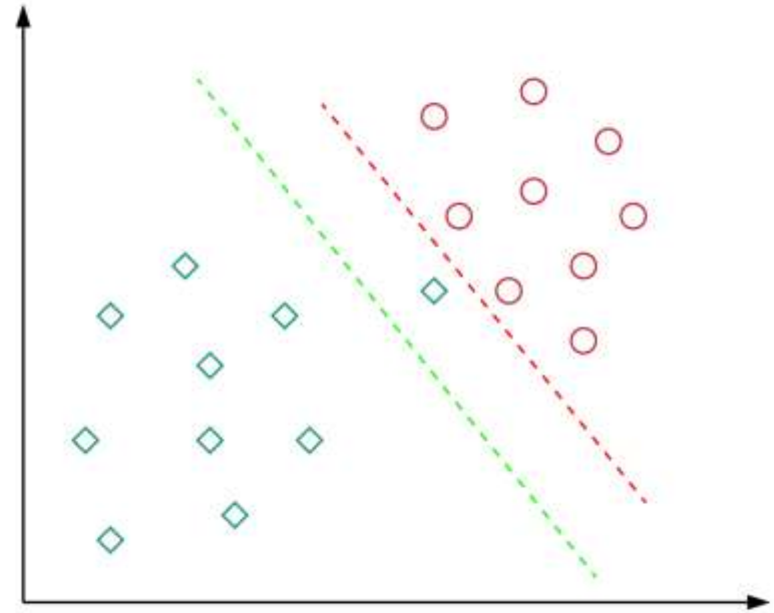
- **Hard Margin:** **So far we require all data points be classified correctly**

  - No training error

- **What if the training set is noisy?**

Motivation : Softmargin

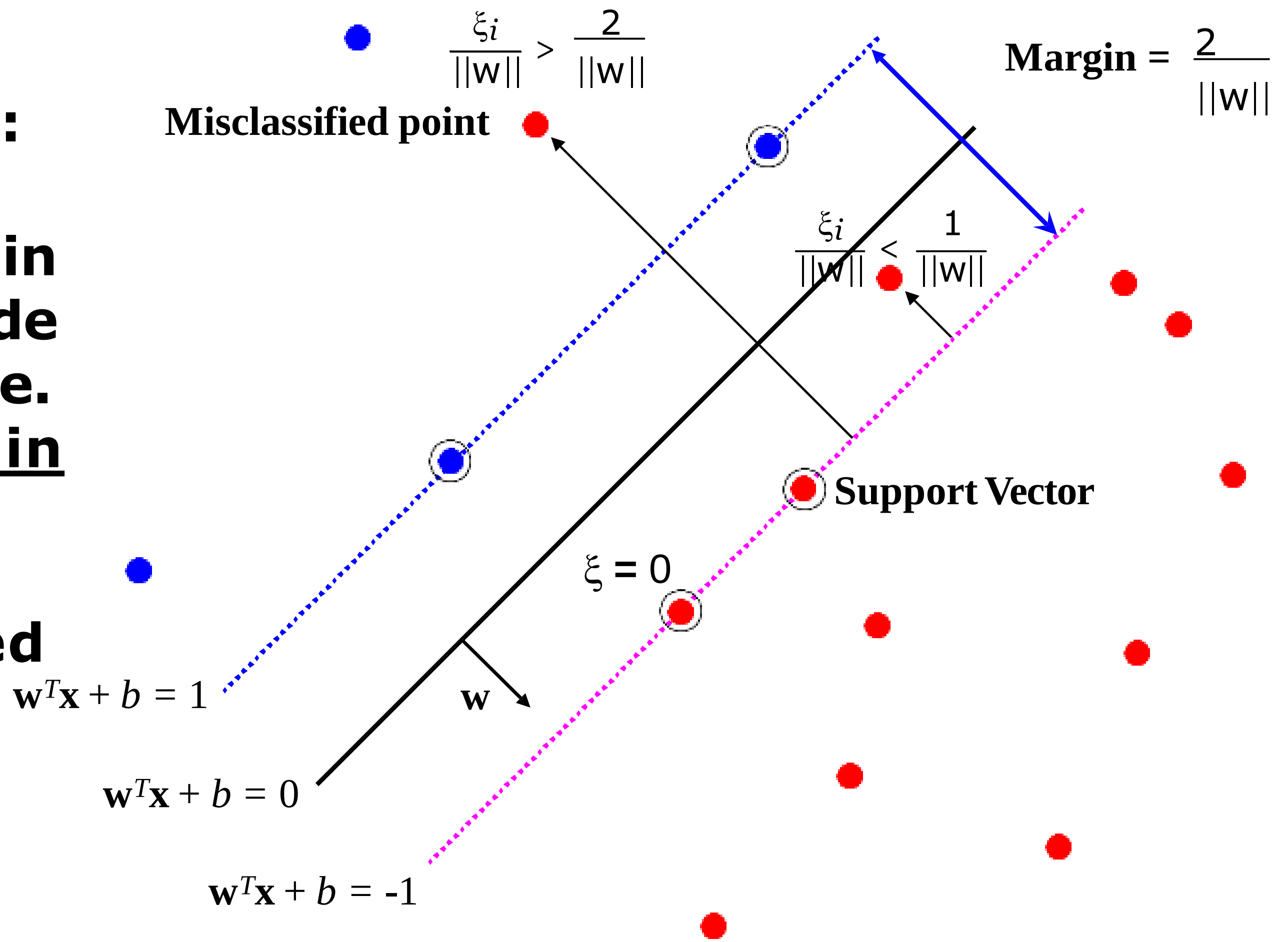
- Almost all real-world applications have data that is linearly inseparable.
- When data *is* linearly separable, choosing perfect decision boundary leads to overfitting



Introduce “slack” variables

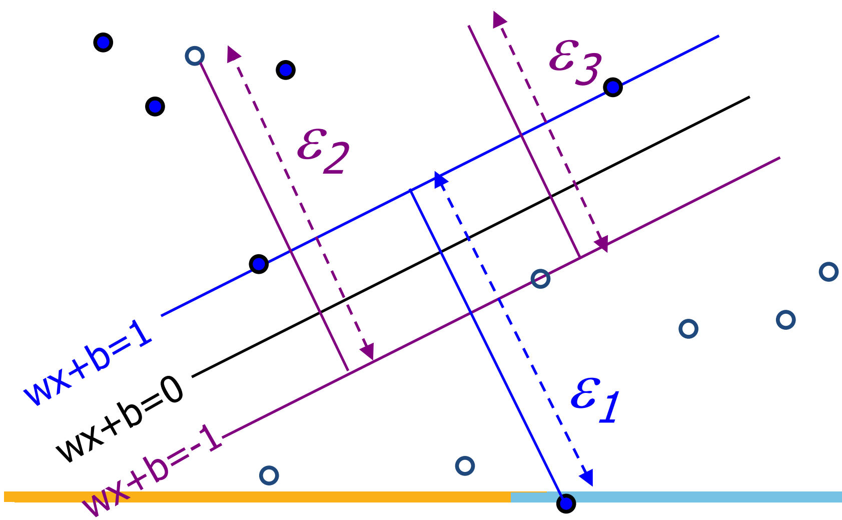
$\xi_i \geq 0$

- **for $0 < \xi \leq 1$: point is between margin and correct side of hyper- plane. This is a <u>margin violation</u>**
- **for $\xi > 1$ point is misclassified**



Soft Margin Classification

- ***Slack variables*** $\boldsymbol{\xi_i}$ **can be added to allow misclassification of difficult or noisy examples.**
- slack variable ξ_i for every data point x_i,
- ξ_i is the distance of x_i from the corresponding class's margin if x_i is on the wrong side of the margin, otherwise zero. **It can be regarded as a measure of confidence**
- Points that are far away from the margin on the wrong side would get more penalty



Soft Margin SVMs

\# data samples

Misclassification cost

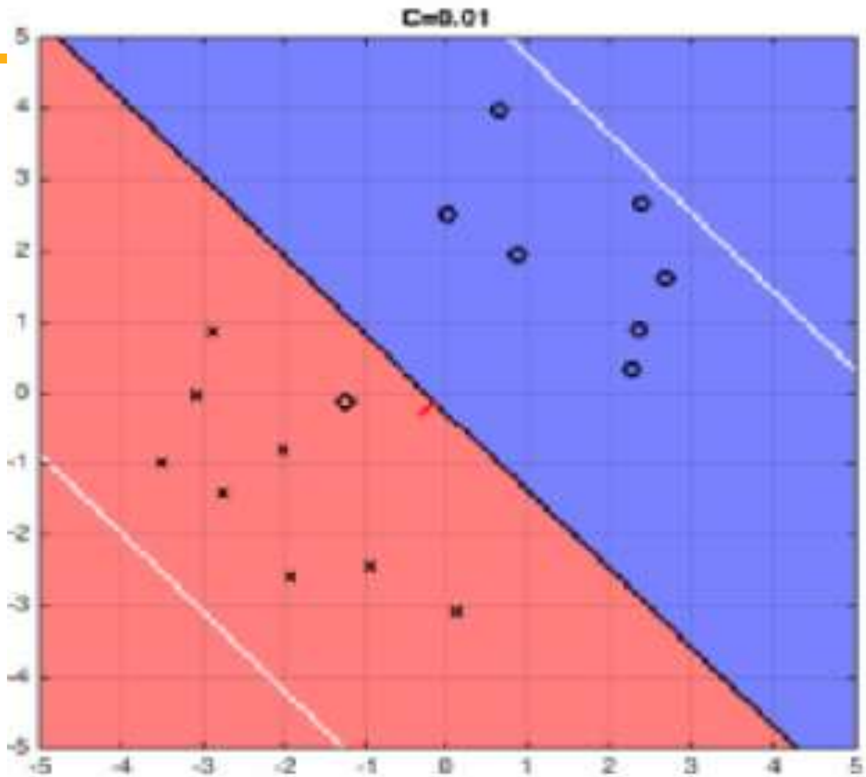
Slack variable

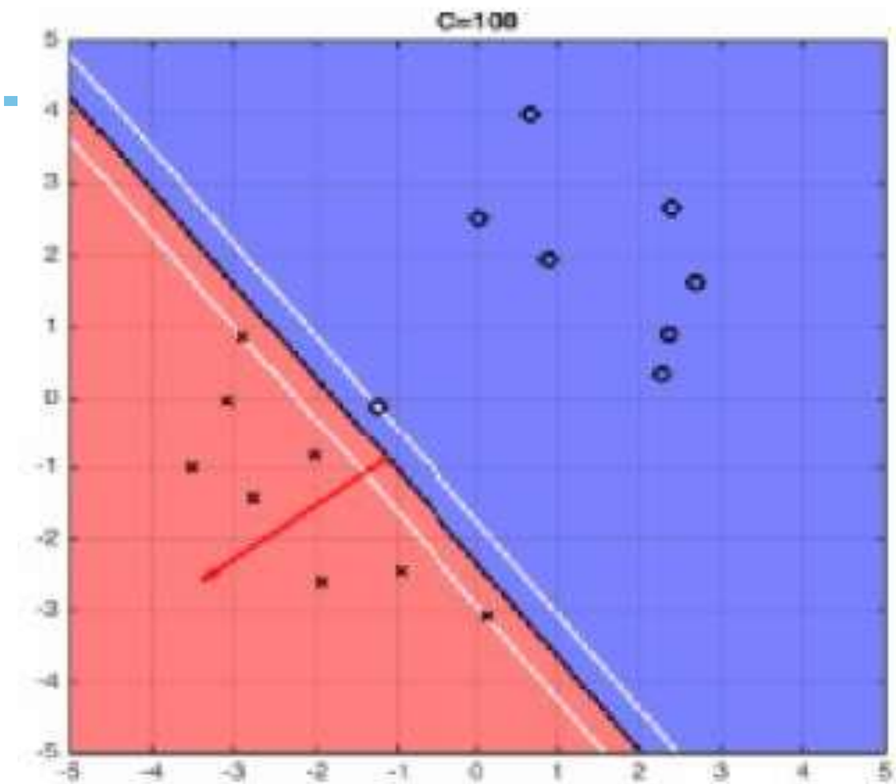
$$\min_{\boldsymbol{w}} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{Maximize margin}} + \underbrace{C\sum_{i=1}^{N}\xi_i}_{\text{Minimize misclassification}}$$

The *w* that minimizes…

$$\boxed{\begin{aligned} \text{subject to} \quad & y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad \forall i = 1, \ldots, N \end{aligned}}$$

Effect of C



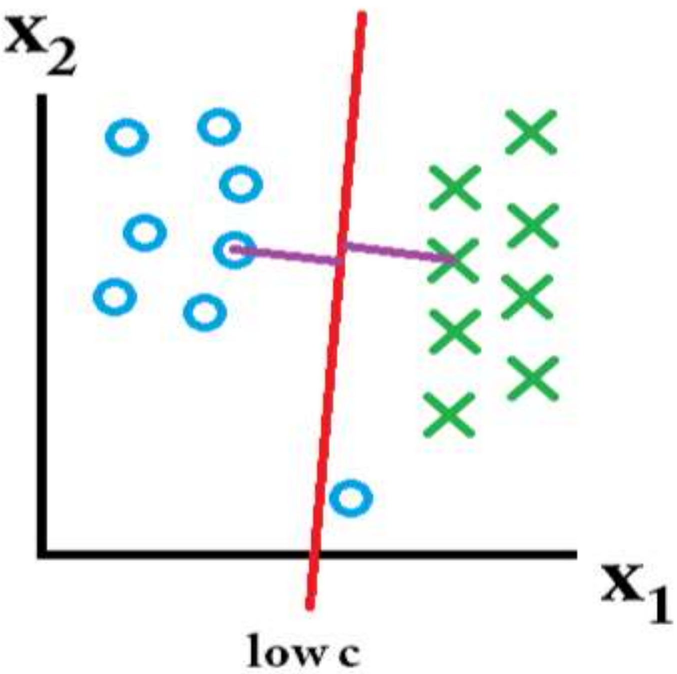


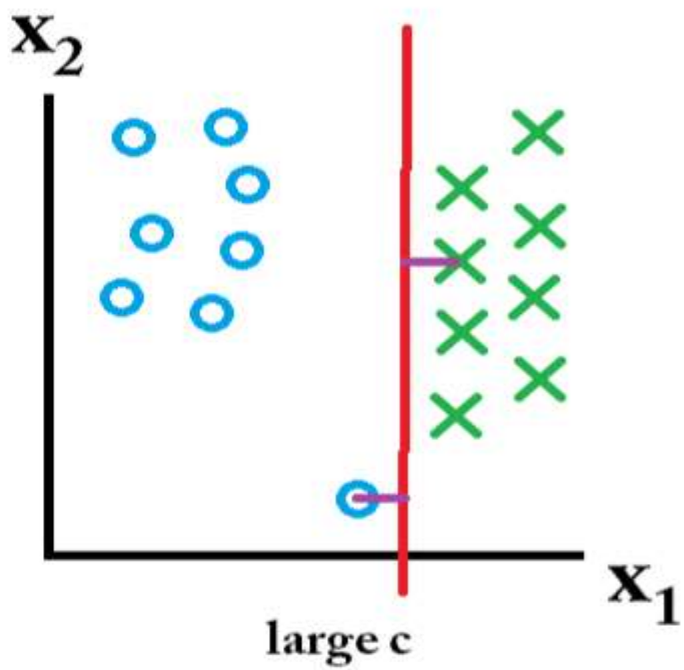
small value of C will cause the optimizer to look for a larger-margin (small penalties) separating hyperplane, even if that hyperplane misclassifies more points.

For large values of C, the optimization will choose a smaller-margin (large penalties) hyperplane if that hyperplane does a better job of getting all the training points classified correctly. C=infinity ->hard margin SVM

Effect of Margin size v/s misclassification cost

– Training set





Misclassification ok, want large margin

Misclassification not ok

classification mistakes are given less importance,

• the focus is more on avoiding misclassification at the expense of keeping the margin small.

SVM Problem

SVM: Optimization

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$

$$\text{Subject to:} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

SVM: Training

Input: (X,y), C
Output: alpha for support vectors, b
Hyperparameter : C

SVM: Classification

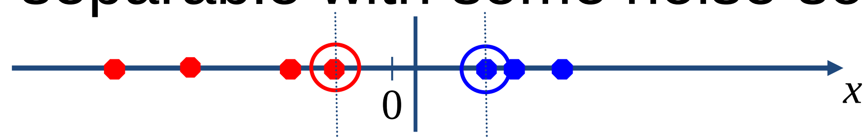
$$sign(\mathbf{w}^T\mathbf{x} + b)$$
$$= sign(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b)$$

- C parameter tells the SVM optimization how much you want to avoid misclassifying each training example and *C* can be viewed as a way to control overfitting

Nonlinear SVM - kernels

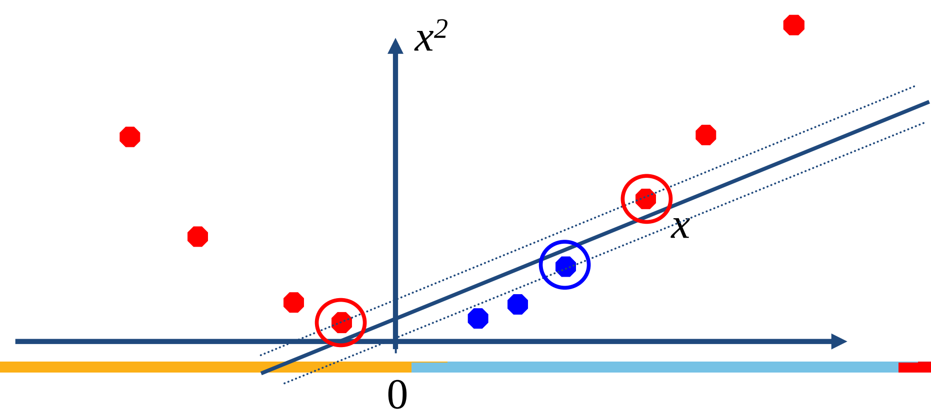
Non-linear SVMs

- What if data is not noisy or having outliers(soft margin can work) but inherently non linear
- Datasets that are linearly separable with some noise soft margin work out great:



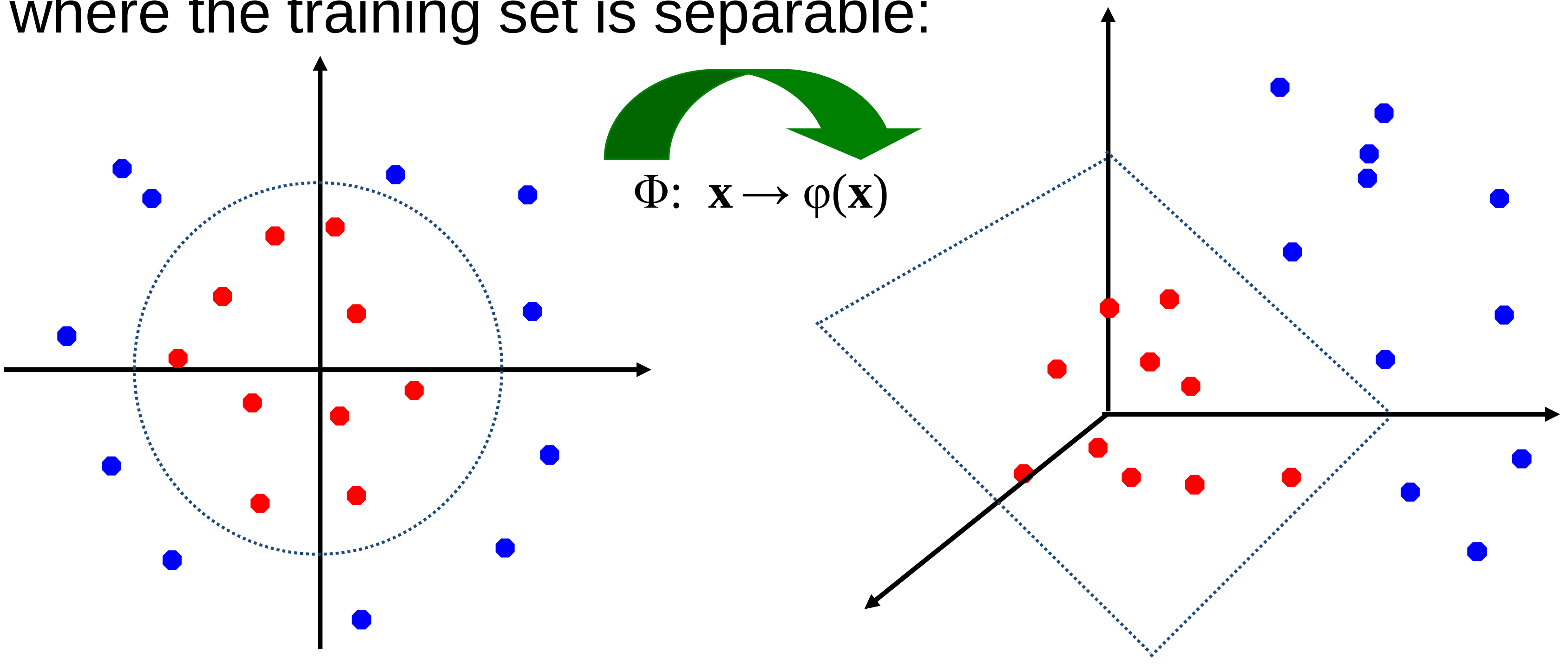
- But what are we going to do if the dataset is just too hard?

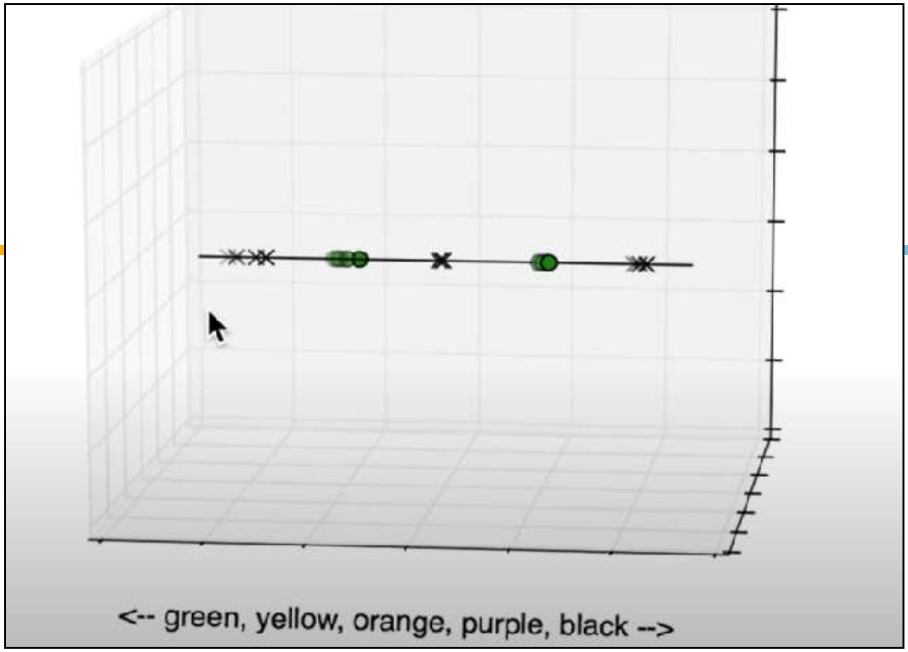
- How about… mapping data to a higher-dimensional space:

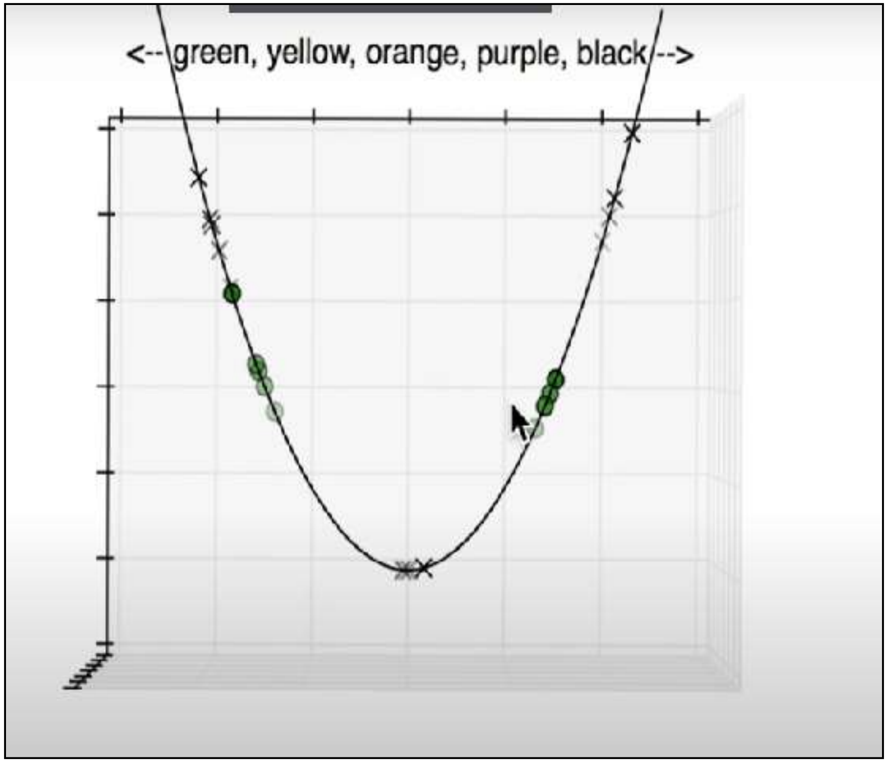


Non-linear SVMs: Feature spaces

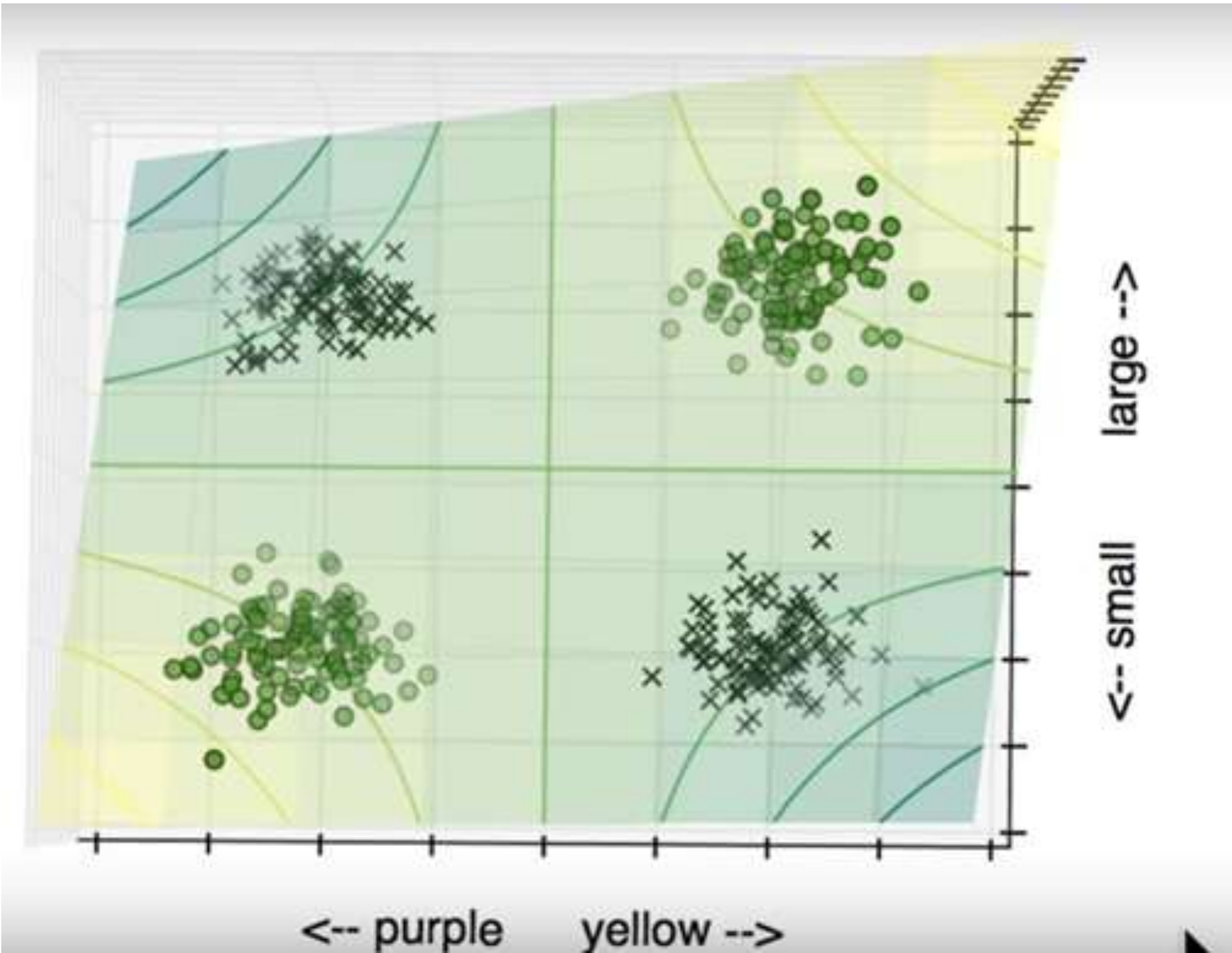
- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



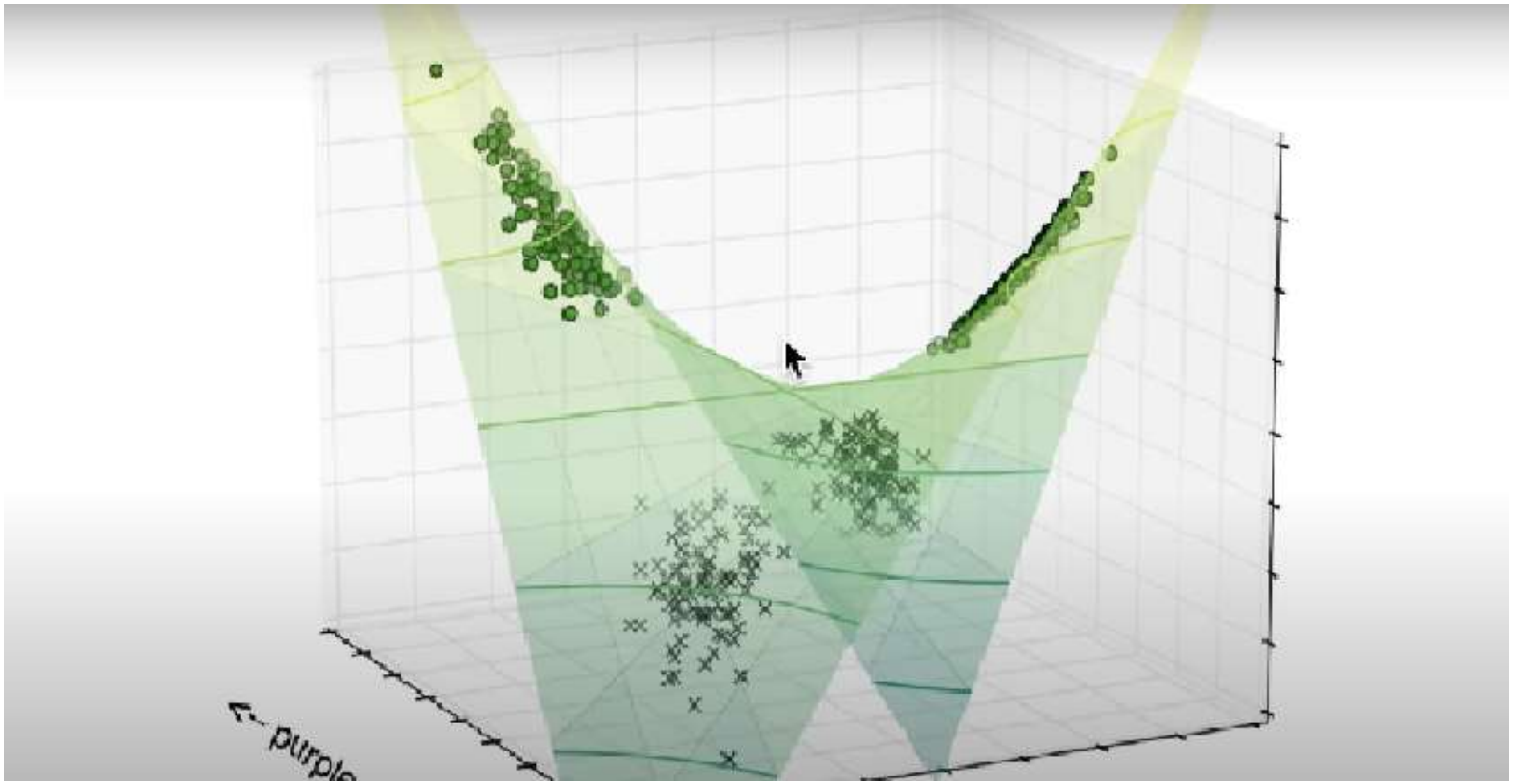

<-- green, yellow, orange, purple, black -->

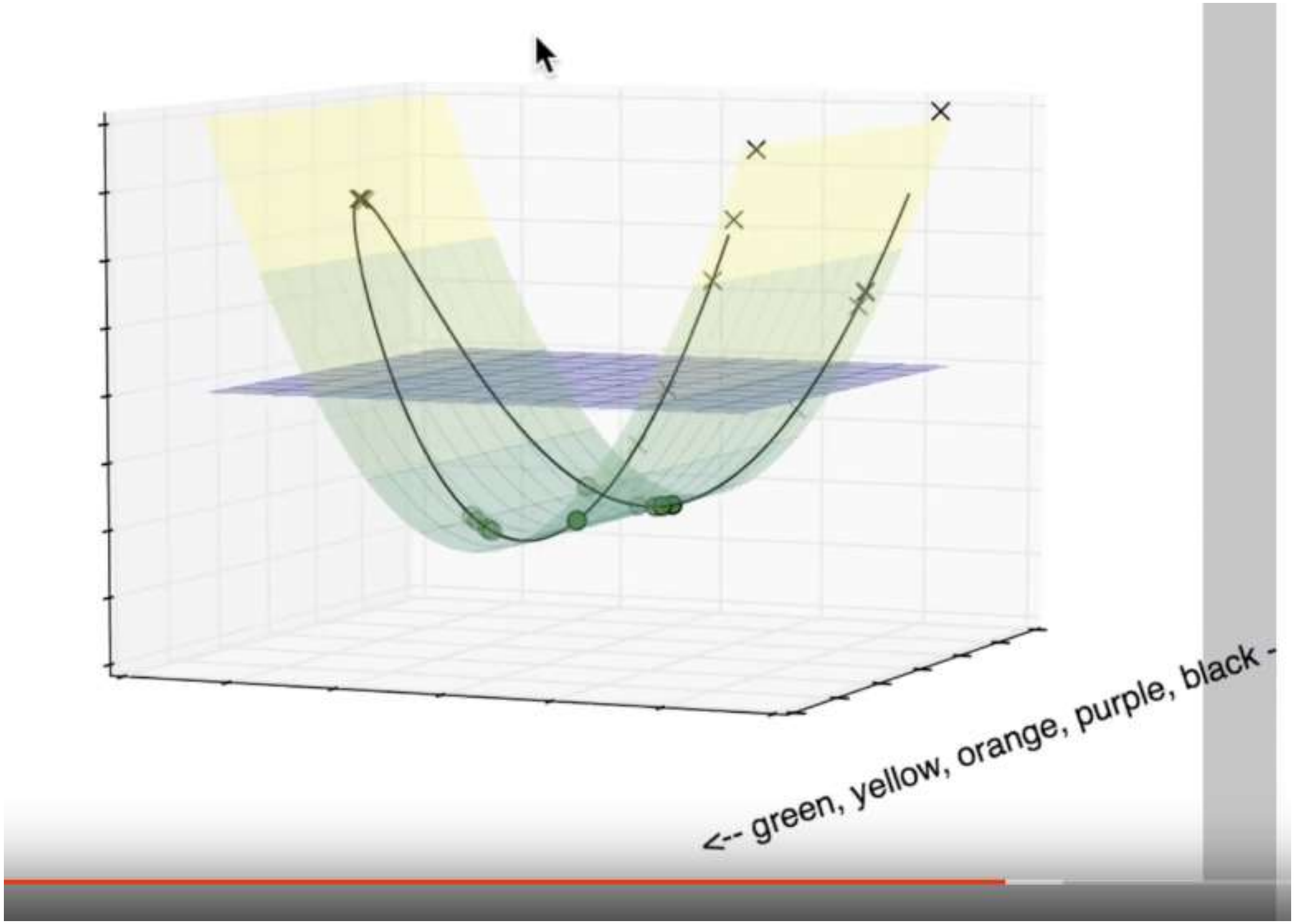

<-- green, yellow, orange, purple, black -->

Non-linear SVMs: Feature spaces



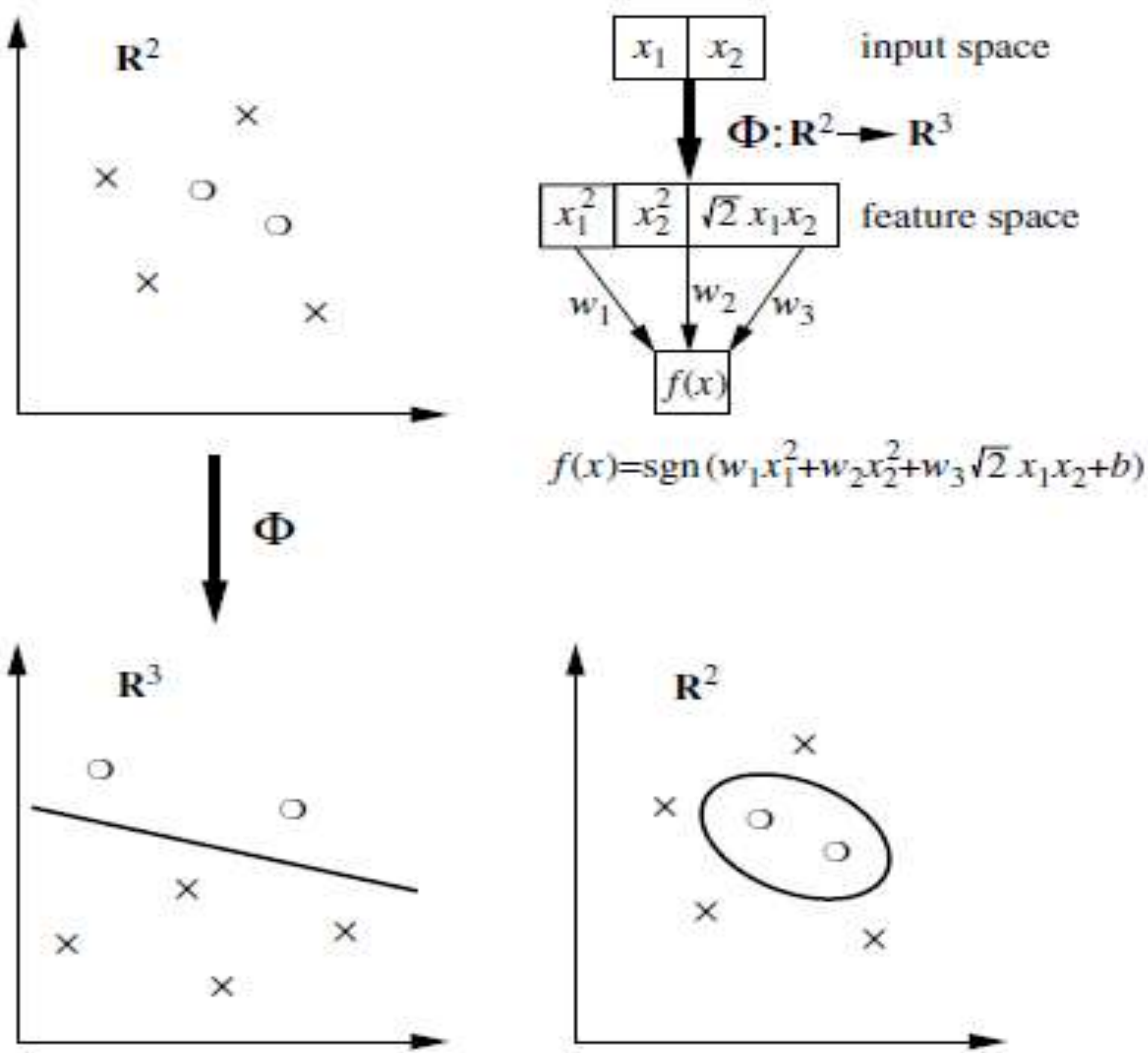
Non-linear SVMs: Feature spaces





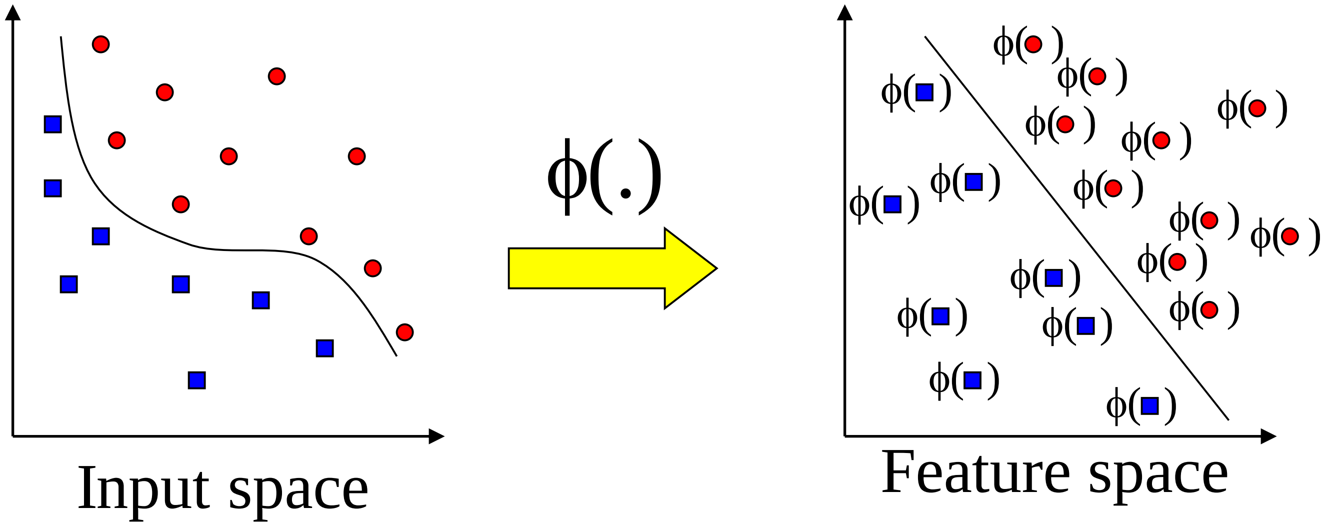
<-- green, yellow, orange, purple, black -

Mapping into a New Feature Space



- Rather than run SVM on x_i, run it on $\Phi(x_i)$
- Find non-linear separator in input space
- What if $\Phi(x_i)$ is really big?
- Use kernels to compute it implicitly!

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

SVM Kernels

- SVM algorithms use a set of mathematical functions that are defined as the kernel.
- Function of kernel is to take data as input and transform it into the required form.
- Different SVM algorithms use different types of kernel functions. Example *linear, nonlinear, polynomial, and sigmoid etc.*

The "Kernel Trick"

- **The linear classifier relies on dot product between vectors**
  - $\mathbf{x_i}^T \cdot \mathbf{x_j}$
- **If every data point is mapped into high-dimensional space via some transformation Φ: x → φ(x), the dot product becomes:**

$$K(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i})^T \, \varphi(\underline{\mathbf{x_j}})$$

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Computing K(x1,x2) should be efficient, much more so than computing Φ(x1) and Φ(x2)**

Kernel: Example
Polynomial Kernel

Let $\mathbf{p} = [p_1, p_2]^T$ and $\mathbf{q} = [q_1, q_2]^T$ be two samples in 2D.

$$\kappa(\mathbf{p}, \mathbf{q}) = (\mathbf{p}^T \mathbf{q})^2 = ([p_1, p_2]^T [q_1, q_2])^2$$
$$= (p_1 q_1)^2 + (p_2 q_2)^2 + 2 p_1 q_1 p_2 q_2$$

This is equivalent to:

$$\phi(\mathbf{p}) = \phi\left(\begin{array}{c} p_1 \\ p_2 \end{array} \right) = \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2} p_1 p_2 \end{bmatrix}$$

Example:

2-dimensional vectors $\mathbf{x}=[x_1 \ \ x_2]$;

let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

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

$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_j$

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

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

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).

Mercer kernels

What functions are valid kernels that correspond to feature vectors $\varphi(x)$?

Mercer kernels K

- K is continuous
- K is symmetric
- K is positive semi-definite, i.e. $x^T K x \geq 0$ for all x

Ensures optimization is concave maximization

What Functions are Kernels?

1) We can *construct kernels from scratch*:

- For any $\varphi : \mathcal{X} \to \mathbb{R}^m$, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathbb{R}^m}$ is a kernel.
- If $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *distance function*, i.e.
  - $d(x, x') \geq 0$ for all $x, x' \in \mathcal{X}$,
  - $d(x, x') = 0$ only for $x = x'$,
  - $d(x, x') = d(x', x)$ for all $x, x' \in \mathcal{X}$,
  - $d(x, x') \leq d(x, x'') + d(x'', x')$ for all $x, x', x'' \in \mathcal{X}$,

  then $k(x, x') := \exp(-d(x, x'))$ is a kernel.

2) We can *construct kernels from other kernels*:

- if k is a kernel and $\alpha > 0$, then αk and $k + \alpha$ are kernels.
- if k_1, k_2 are kernels, then $k_1 + k_2$ and $k_1 \cdot k_2$ are kernels.

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$

- Polynomial of power p: $K(\mathbf{x_i},\mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^p$

- Gaussian (radial-basis function network):

$$K(\mathbf{x_i},\mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x_i},\mathbf{x_j}) = \tanh(\beta_0 \mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

Using a Different Kernel in the Dual Optimization Problem

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

- For example, using the polynomial kernel with d = 4 (including lower-order terms) → $(\langle x_i, x_j \rangle + 1)^4$
- Maximize over α
  - $W(\alpha) = \Sigma_i \alpha_i - 1/2\, \Sigma_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$
- Subject to
  - $\alpha_i \geq 0$
  - $\Sigma_i \alpha_i y_i = 0$
- Decision function
  - $f(x) = \text{sign}(\Sigma_i \alpha_i y_i \langle x, x_i \rangle + b)$

These are kernels!

$(\langle x_i, x_j \rangle + 1)^4$

So by the kernel trick,
we just replace them!

Non-linear SVM using kernel

1. Select a kernel function.
2. Compute pairwise kernel values between labeled examples.
3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.
4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Multi-Class Problem

Instead of just two classes, we now have C classes

- E.g. predict which movie genre a viewer likes best
- Possible answers: action, drama, indie, thriller, etc.

Two approaches:

- One-vs-all
- One-vs-one

Multi-Class Problem

One-vs-all (a.k.a. one-vs-others)

- Train C classifiers
- In each, pos = data from class *i*, neg = data from classes other than *i*
- The class with the most confident prediction wins
- Example:
  - You have 4 classes, train 4 classifiers
  - 1 vs others: score 3.5
  - 2 vs others: score 6.2
  - 3 vs others: score 1.4
  - 4 vs other: score 5.5
  - Final prediction: class 2
- Issues?

Multi-Class Problem

One-vs-one (a.k.a. all-vs-all)

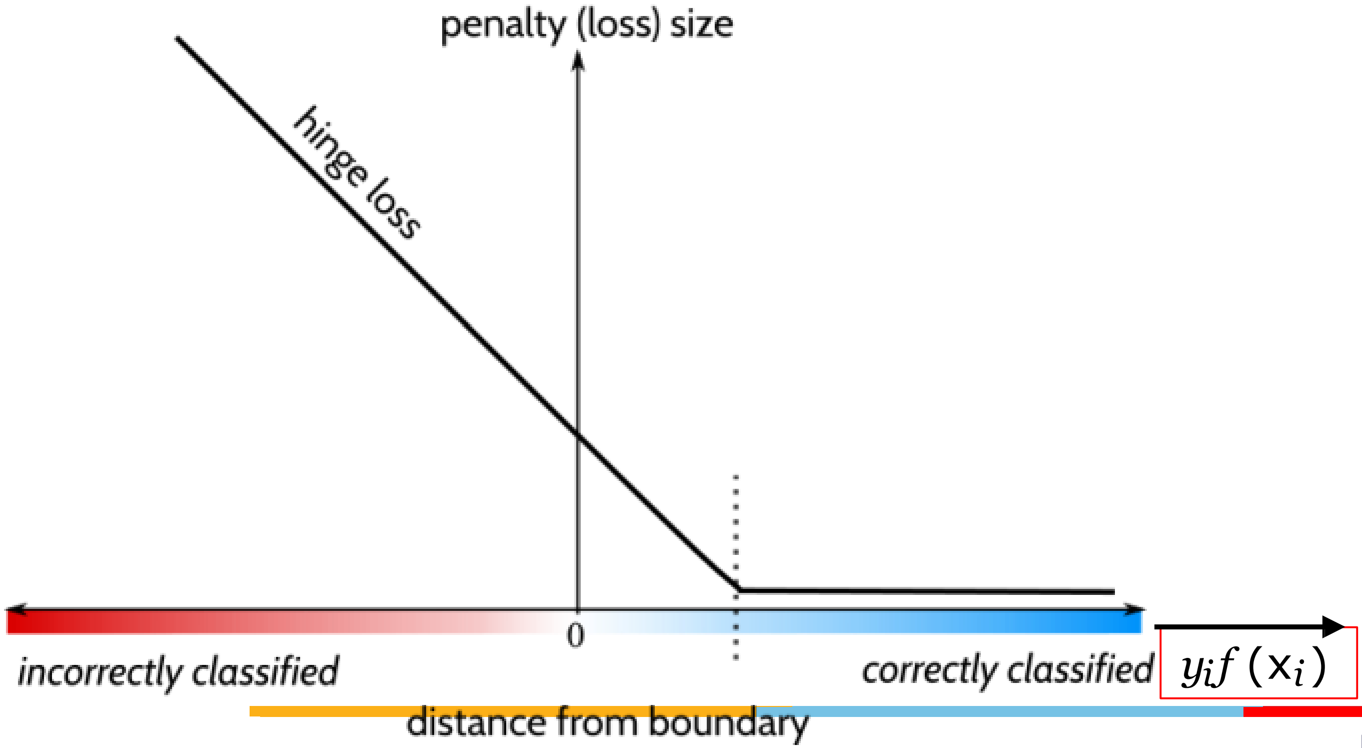
- Train C(C-1)/2 binary classifiers (all pairs of classes)
- They all vote for the label
- Example:
  - You have 4 classes, then train 6 classifiers
  - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
  - Votes: 1, 1, 4, 2, 4, 4
  - Final prediction is class 4

Slack variable-Hinge loss

$$\xi_j = \text{loss}(f(x_j), y_j)$$

$$f(x_j) = \text{sgn}(\mathbf{w} \cdot \mathrm{x_j} + \mathrm{b})$$

$$\xi_j = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$



SVM versus Logistic Regression

- When viewed from the point of view of regularized empirical loss minimization, SVM and logistic regression appear quite similar:

SVM: $$\sum_{i=1}^{n} \left(1 - y_i \left[w_0 + \mathbf{x}_i^T \mathbf{w}_1\right]\right)^+ + \quad \|\mathbf{w}_1\|^2/2$$

Logistic: $$\sum_{i=1}^{n} \overbrace{-\log \sigma\left(y_i \left[w_0 + \mathbf{x}_i^T \mathbf{w}_1\right]\right)}^{-\log P(y_i|\mathbf{x},\mathbf{w})} + \quad \|\mathbf{w}_1\|^2/2$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logistic function.

SVM versus Logistic Regression

- The difference comes from how we penalize "errors":

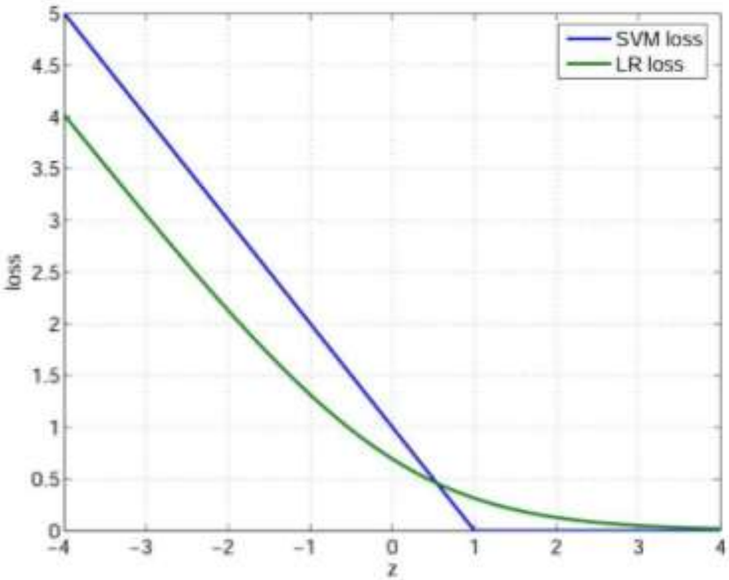
Both: $$\sum_{i=1}^{n} \text{Loss}\Big(\overbrace{y_i\left[w_0 + \mathbf{x}_i^T \mathbf{w}_1\right]}^{z}\Big) + \quad \|\mathbf{w}_1\|^2/2$$

- SVM:

$$\text{Loss}(z) = (1 - z)^+$$

- Regularized logistic reg:

$$\text{Loss}(z) = \log(1 + \exp(-z))$$



SVM versus Logistic Regression

- Logistic regression focuses on maximizing the probability of the data. The farther the data lies from the separating hyperplane (on the correct side), the happier LR is
- An SVM tries to find the separating hyperplane that maximizes the distance of the closest points to the margin (the support vectors). If a point is not a support vector, it doesn't really matter.

SVM versus Logistic Regression

- LR gives calibrated probabilities that can be interpreted as confidence in a decision.
- LR gives us an unconstrained, smooth objective
- LR can be (straightforwardly) used within Bayesian models.
- SVMs don't penalize examples for which the correct decision is made with sufficient confidence. This may be good for generalization.
- SVMs have a nice dual form, giving sparse solutions when using the kernel trick (better scalability).

Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - Only support vectors are used to specify the separating hyperplane
  - Therefore SVM also called sparse kernel machine.
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection

Strengths of SVMs

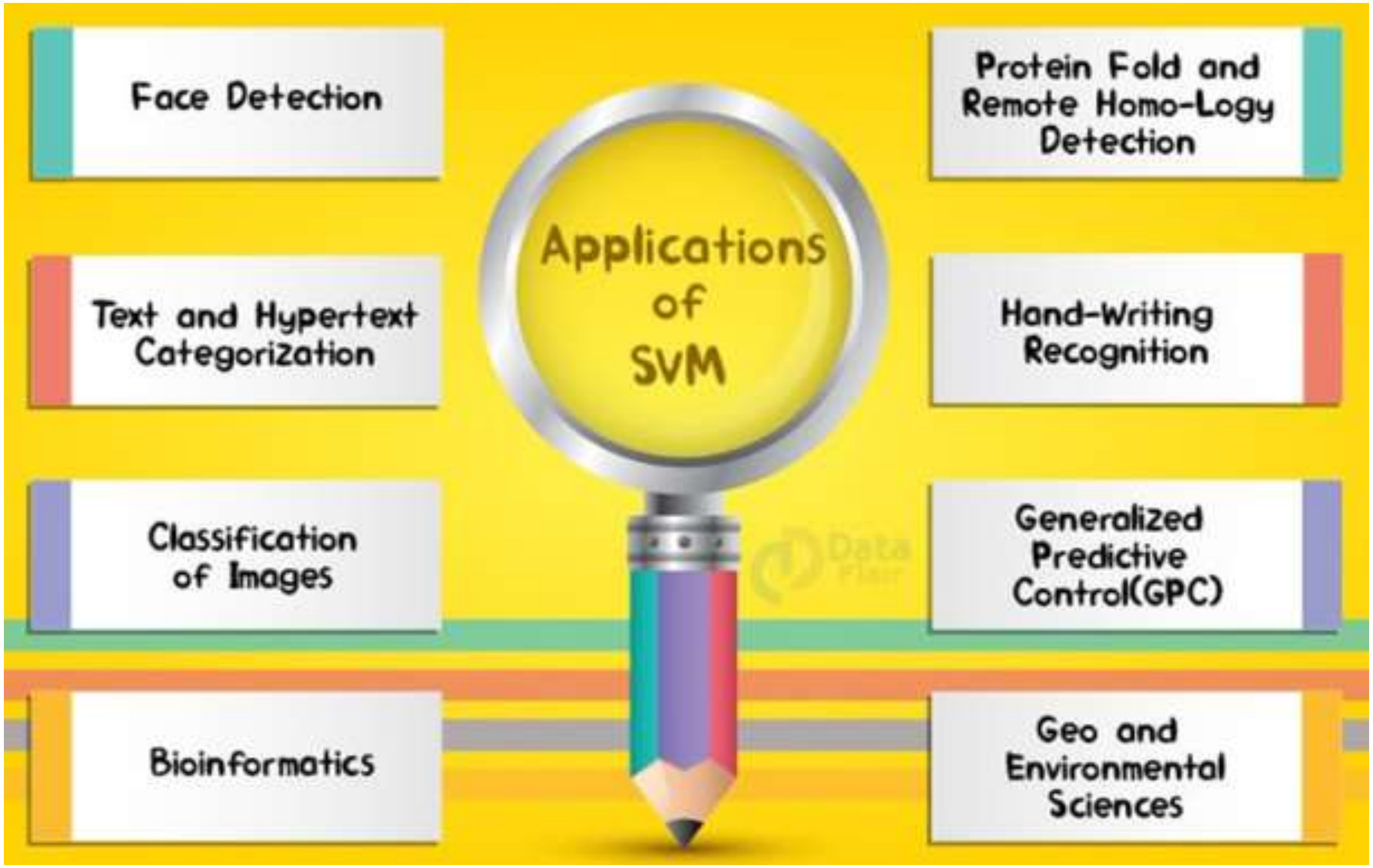
- Good generalization in theory
- Good generalization in practice
- Work well with few training instances
- Find globally best model
- Efficient algorithms
- Amenable to the kernel trick
- Although the SVM based classification (i.e., training time) is extremely slow, the result, is however highly accurate. Further, testing an unknown data is very fast.

Some Issues

- **Sensitive to noise**
  - A relatively small number of mislabeled examples can dramatically decrease the performance
- **Choice of kernel**
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures
- **Choice of kernel parameters**
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- **Optimization criterion – Hard margin v.s. Soft margin**
  - a lengthy series of experiments in which various parameters are tested
- **Large data sets.**
  - Calculating the kernel is expensive.

SVM Applications

SVM has been used successfully in many real-world problems



Application : Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.

  A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

Text Categorization using SVM

- **The distance between two documents is $\phi(x)\cdot\phi(z)$**

- **$K(x,z) = \phi(x)\cdot\phi(z)$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination.**

- **Why SVM?**

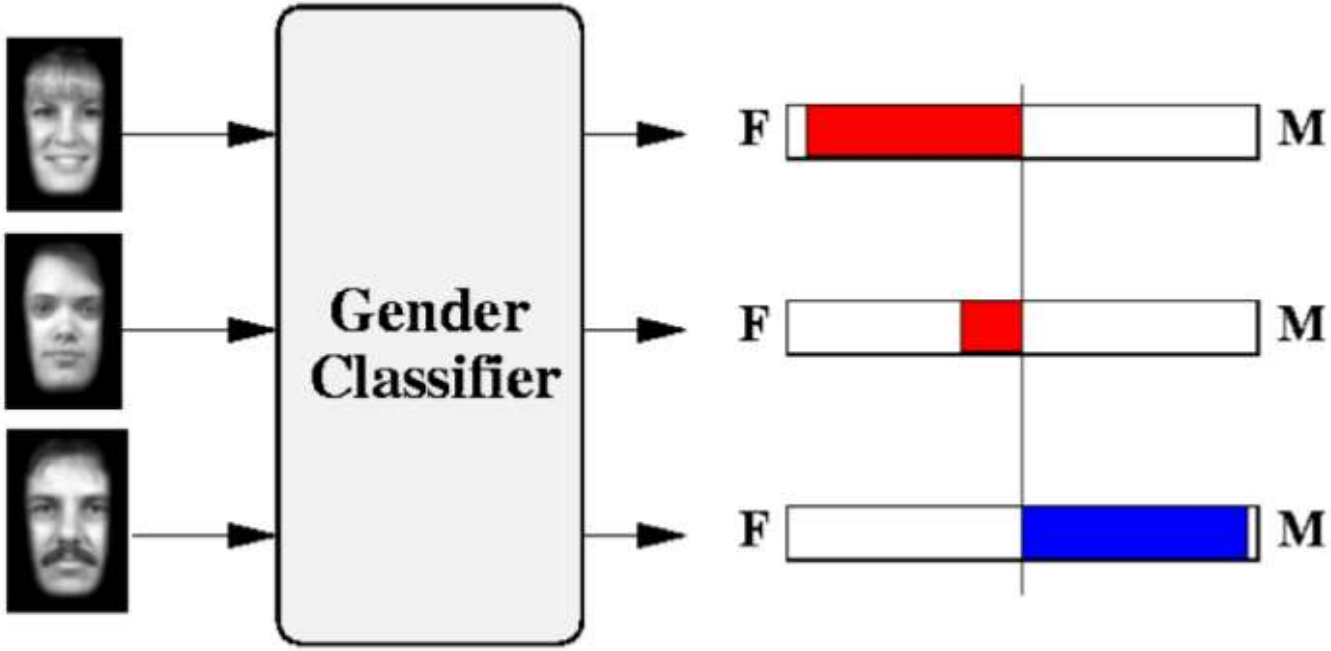
-High dimensional input space

-Few irrelevant features (dense concept)

-Sparse document vectors (sparse instances)

-Text categorization problems are linearly separable

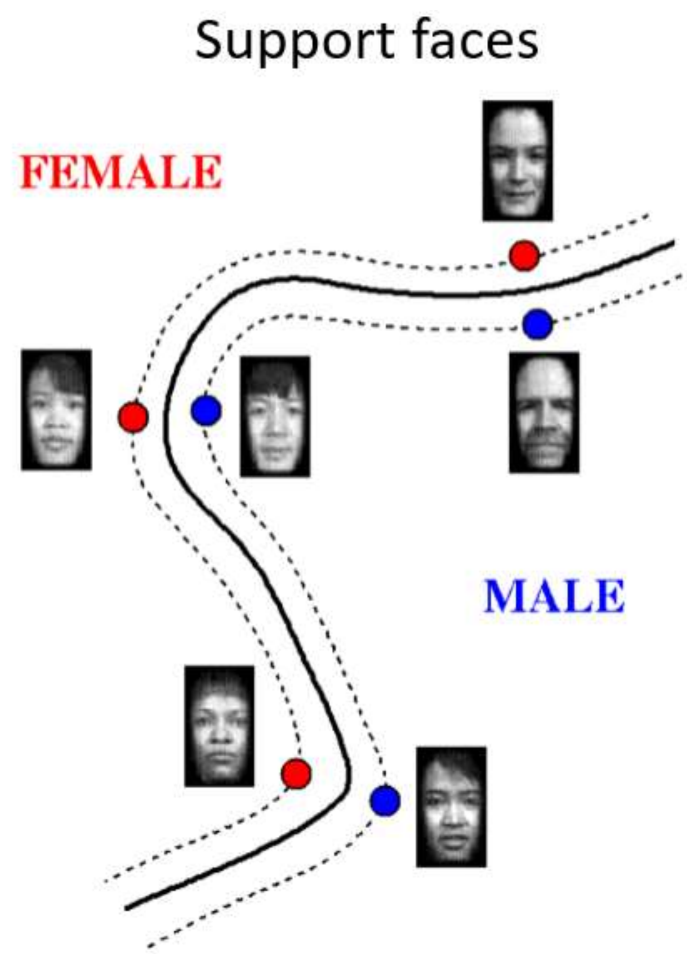
Learning Gender from image with SVM



Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002

Moghaddam and Yang, Face & Gesture 2000

Support faces



Summary

SVM Points to remember

- Primal and Dual optimization problems
- Kernel functions
- Support Vector Machines
  - Maximizing margin
  - Derivation of SVM formulation
  - Slack variables and hinge loss
- Relationship between SVMs and logistic regression
  - 0/1 loss
  - Hinge loss
  - Log loss

Hard Margin versus Soft Margin

- **Hard Margin:**

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

$\mathbf{\Phi(w)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w^T x_i} + \mathrm{b}) \geq 1$

- **Soft Margin incorporating slack variables:**

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

$\mathbf{\Phi(w)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\,\Sigma\,\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Corresponding Lagrangian function**

$$L = \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i + \sum_i \lambda_i (y_i(\vec{w}\,.\,\vec{x}_i + b) - 1 + \xi_i)$$

Soft Margin Vs Hard margin Classification – Solution

The dual problem

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) $\mathbf{0 \leq \alpha_i \leq C}$ **for all** $\boldsymbol{\alpha_i}$

Solution to the dual problem

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

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$

$\boldsymbol{b = y_k(1- \xi_k) - \mathrm{w}^{\mathrm{T}}\mathrm{x}_k}$ **where** $\boldsymbol{k = \mathrm{argmax}\ \alpha_k{}'}$

Find $\alpha_1 \ldots \alpha_N$ such that

The dual problem

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \right)$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\boldsymbol{\alpha_i \geq 0}$ **for all** $\boldsymbol{\alpha_i}$

Solution to the dual problem

$\mathrm{w} = \Sigma\, \alpha_i y_i \mathrm{x_i}$

$\mathbf{b} = y_i - \mathbf{W.x_i}$

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

Reference

- **Support Vector Machine Classification of Microarray Gene Expression Data**, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- **Text categorization with Support Vector Machines: learning with many relevant features**

  T. Joachims, ECML - 98
- Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition
- **A Tutorial on Support Vector Machines for Pattern Recognition**, Kluwer Academic Publishers - Christopher J.C. Burges

Good Web References for SVM

- http://www.cs.utexas.edu/users/mooney/cs391L/
- https://www.coursera.org/learn/machine-learning/home/week/7
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://data-flair.training/blogs/svm-kernel-functions/
- MIT 6.034 Artificial Intelligence, Fall 2010
- https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior
- https://www.sciencedirect.com/science/article/abs/pii/S0893608006002796
- https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be
- Radial basis kernel
- http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

Thank You