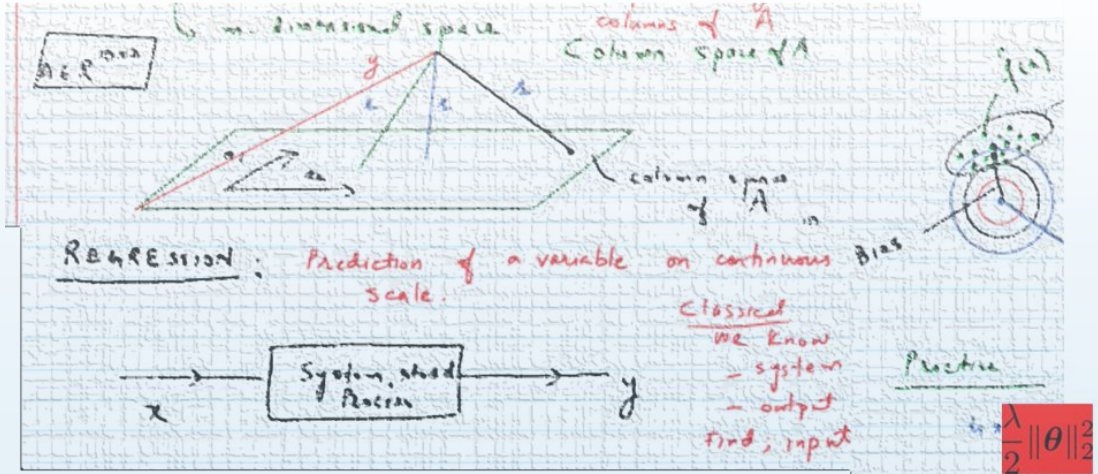
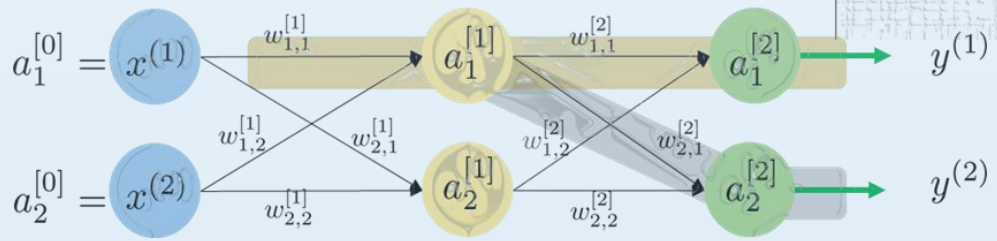
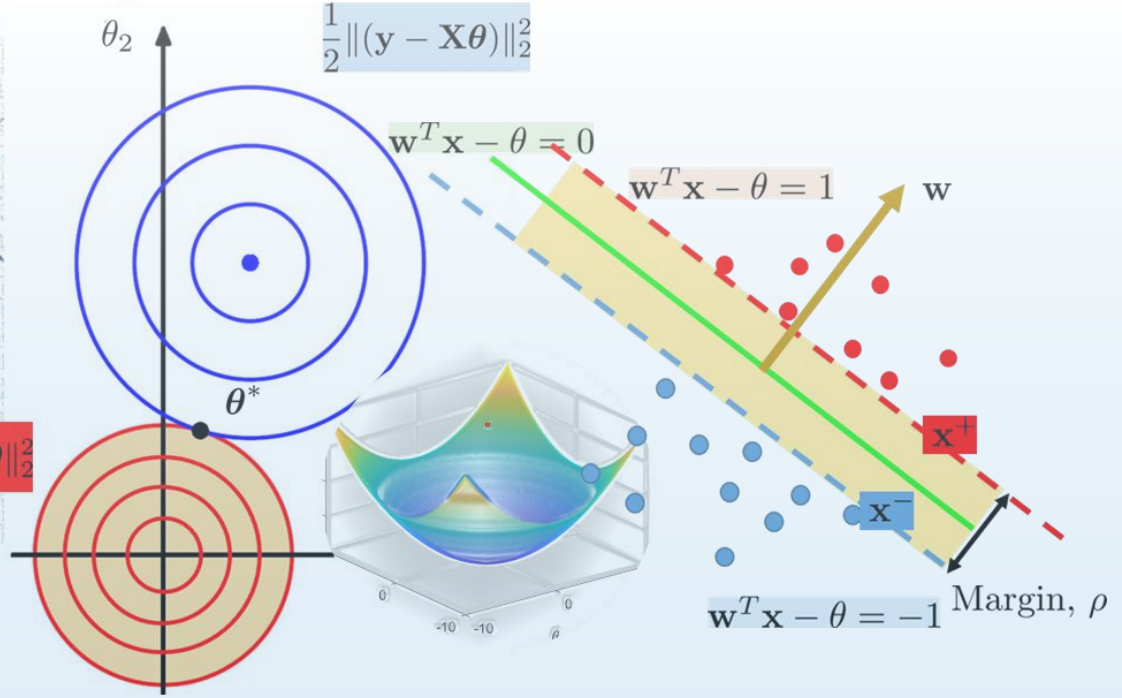






Machine Learning

Support Vector Machines (SVM)

School of Science and Engineering

https://www.zubairkhalid.org/ee514_2025.html

LUMS
A Not-for-Profit University

Outline

- Support Vector Machines (SVM) Overview
- Hard SVM
- Soft SVM
- Kernel Trick
- OvR SVM classifier for Multi-class classification

Support Vector Machines (SVM)

Why SVM?

• **kNN** creates local decision boundaries based on nearest neighbors, which can be *computationally expensive* and *sensitive to noise.*

• **Logistic Regression** works well for linear boundaries but needs *feature engineering for non-linear boundary*. However, it provides *interpretation in terms of probability.*

•**Perceptron Classifier** finds a separating hyperplane but *does not guarantee the best one.*

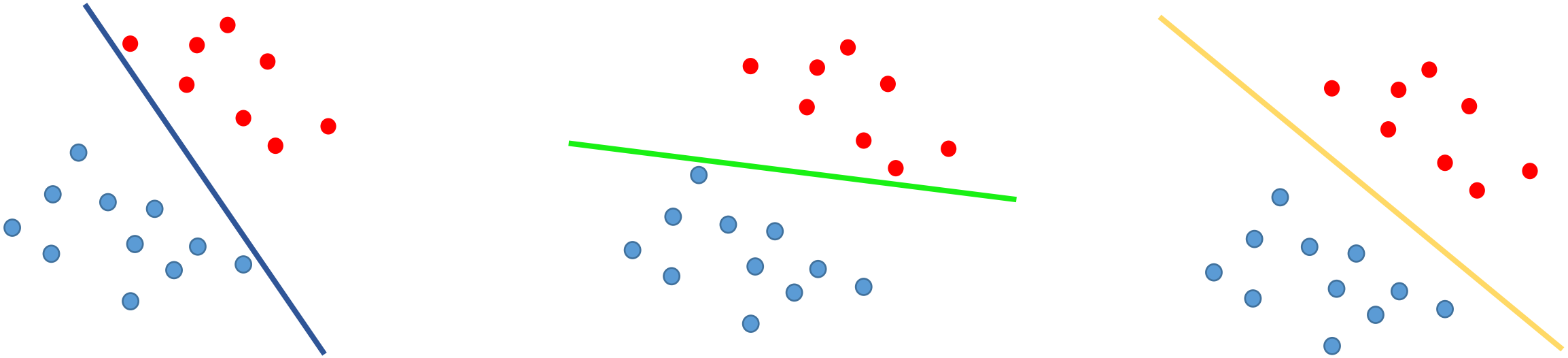
SVMs address these limitations:

- finding the boundary that not only separates the classes but also
- maximizes the margin, making it more generalizable to new data, and
- provides a capability to handle non-linear decision boundaries

Support Vector Machines (SVM)

Maximum Margin Classifier – Overview:

- We have linearly separable classes.
- We can use perceptron classifier to learn the decision boundary (hyper-plane), that separates the classes.
- We can have multiple hyper-planes separating the classes (see illustration).



Q: Which one is the best decision boundary?

A: Maximum Margin Classifier (e.g., Support Vector Machine)

Support Vector Machines (SVM)

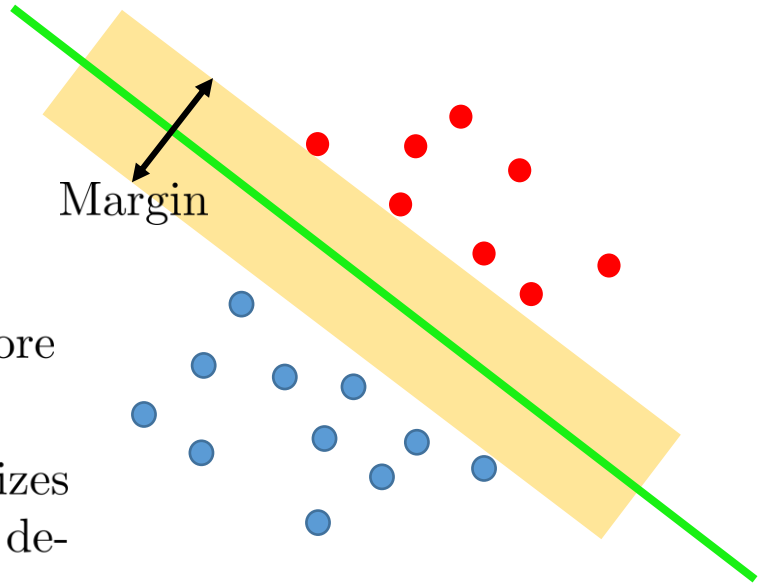
Maximum Margin Classifier – Overview:

- Margin of a data point is defined as the distance from the data point to the decision boundary.

Maximum margin classifier Idea:

Choose a fat separator;

maximize classification margin.

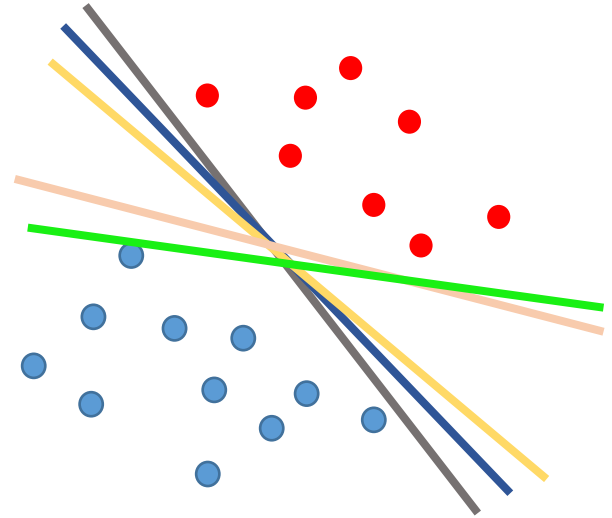


- The margin is the width that the boundary could be increased by before hitting a datapoint.
- The **best** boundary is the one that maximizes this margin or maximizes the distance between the boundary and the "difficult points" close to decision boundary.
- Choose a hyper-plane that is approximately half-way between the nearest positive and negative data points.
- Classification margin is twice the width of the distance of the boundary from the nearest points.
- Support Vector Machine (SVM) classifier gives us the separating hyper-plane that maximizes margin.

Support Vector Machines (SVM)

Overview and Intuition:

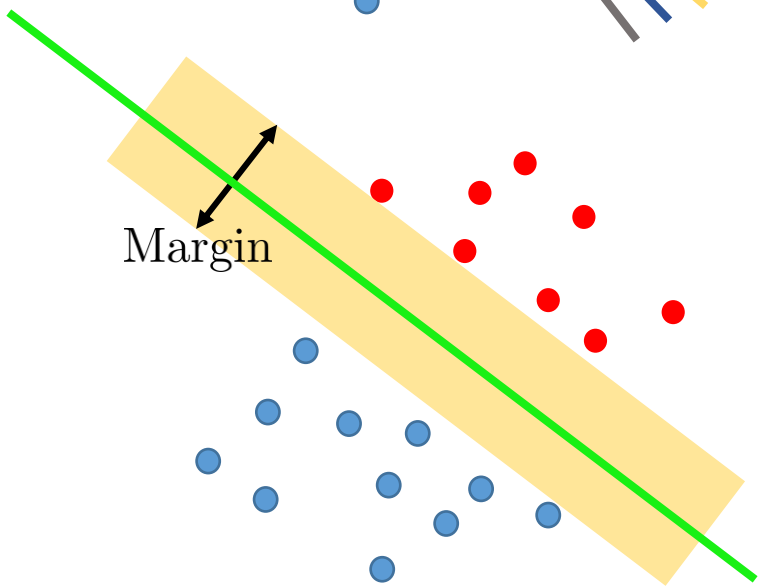
- We have linearly separable classes.
- We can use perceptron classifier to learn the decision boundary (hyper-plane), that separates the classes.
- We can have multiple hyper-planes separating the classes (see illustration).



Q: Which one is the best decision boundary?

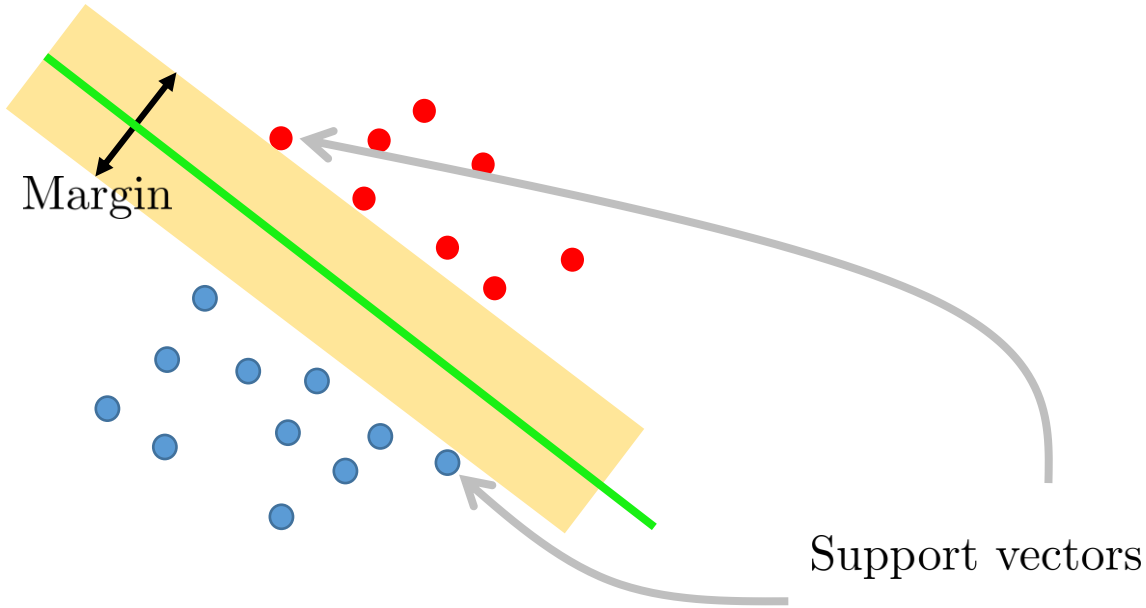
A: Support Vector Machine (Maximum Margin Classifier)

- Idea: Choose a fat separator.
- Choose a hyper-plane that is approximately half-way between the nearest positive and negative data points.
- Margin is twice the width of the distance of the boundary from the nearest points.
- The **best** boundary is the one that maximizes this margin or maximizes the distance between the boundary and the "difficult points" close to decision boundary.



Support Vector Machines (SVM)

SVM – Overview:

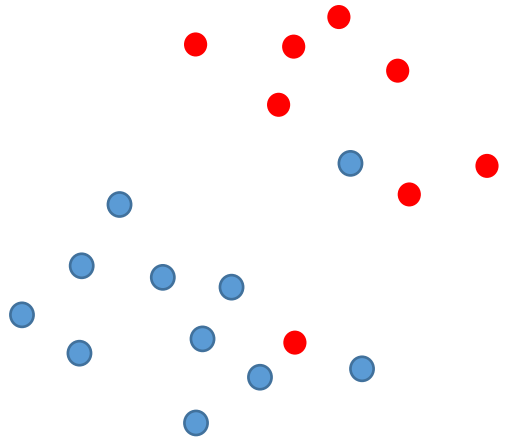


- A support-vector machine is a model that constructs hyper-plane as maximum margin classifier. It can be used for classification, regression, or other tasks like outliers detection.
- Margin pushes against the data points are called **support vectors**.
- How do we mathematically formulate the problem of finding maximum margin decision boundary?
  - We will formulate an optimization, aka quadratic program (QP).
  - But before this, let's look at some other variants.

Support Vector Machines (SVM)

Hard vs Soft Margin – Overview:

- Consider the following classification problem.
- We have a problem here:
  - Classes are linearly separable with some noise.



Q: How do we find the maximum margin decision boundary in this case?

A: Instead of using hard margin, we can use so-called soft margin.

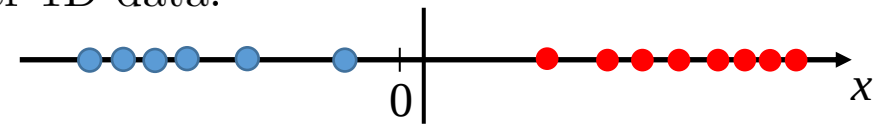
Hard margin idea: **Find maximum margin classifier with no errors on the training data.**

Soft margin idea: **Find maximum margin classifier while minimizing number of training errors.**

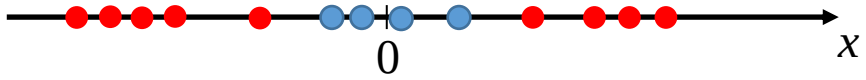
Support Vector Machines (SVM)

The Kernel Trick – Overview:

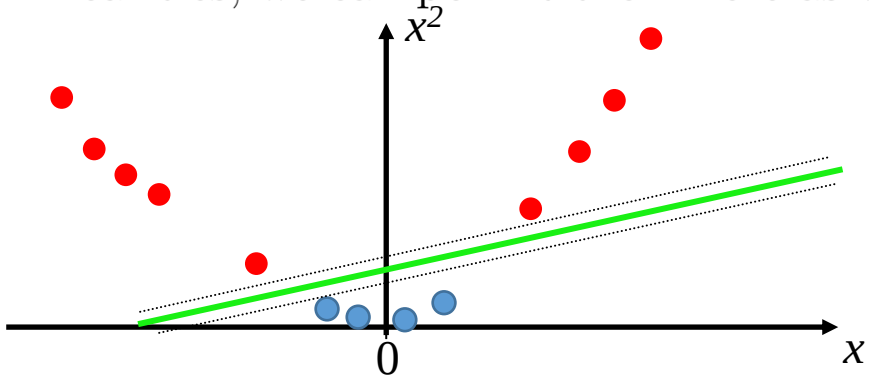
- For simplicity, consider 1D data.



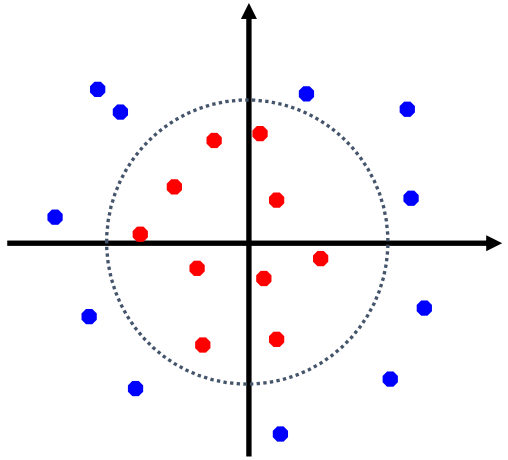
- How can we use SVM for this data?

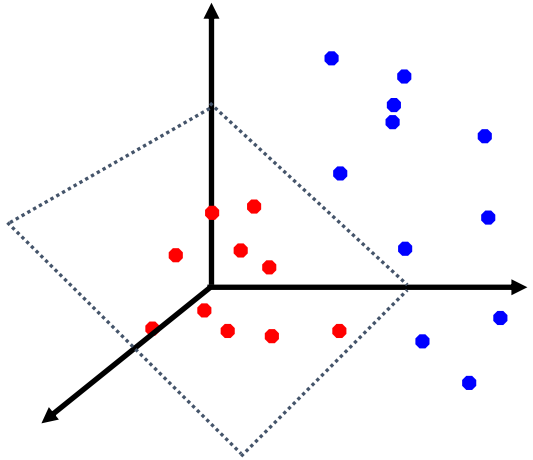


- Since the use of polynomial (non-linear)functions facilitated us in linear regression to model non-linearities, we can permit them here as well.



- 2D to 3D.





General Idea:

Project to original feature space to higher dimensional space to make the classes linearly separable. This mapping function is referred to as the kernel trick and the mapping function is known as the kernel function.

Outline

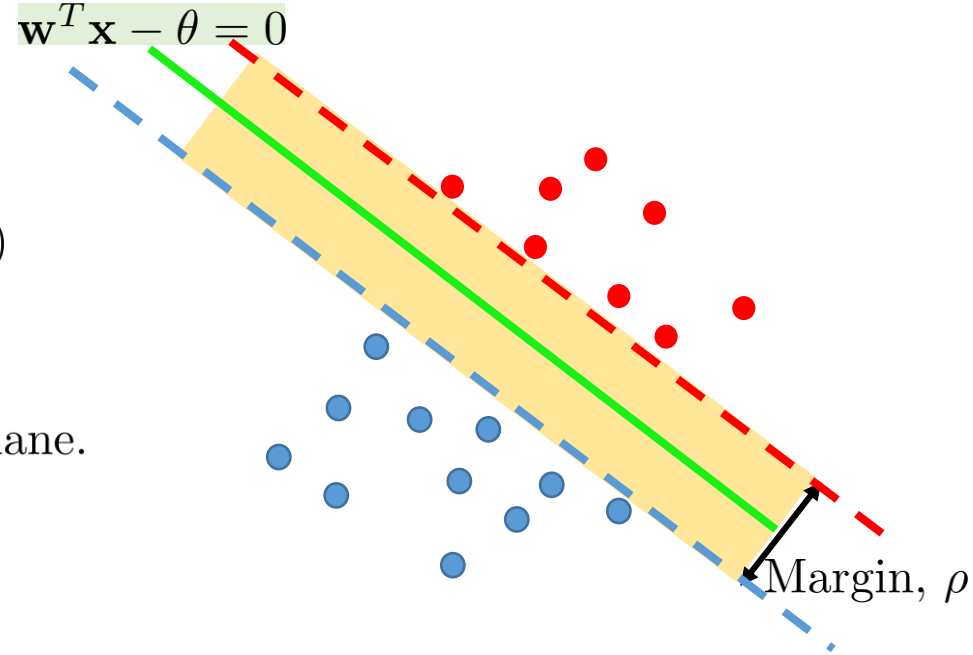
- Support Vector Machines (SVM) Overview
- Hard SVM
- Soft SVM
- Kernel Trick
- OvR SVM classifier for Multi-class classification

Support Vector Machines (SVM)

Hard SVM – Mathematical Formulation:

- n data-points, d number of real-valued features $\mathbf{x} = [x^{(1)}, \ldots, x^{(d)}]$.
- Boolean output, $y \in \{-1, 1\}$.
- Perceptron classifier, characerized by $\mathbf{w} = [w_1, \ldots, w_d]$:

$$y = \begin{cases} 1 \quad \text{if} & \sum_{i=1}^{d} w_i\, x^{(i)} - \theta \geq 0 \text{ or } \sum_{i=0}^{d} w_i\, x^{(i)} \geq 0 \\ -1 \quad \text{if} & \sum_{i=1}^{d} w_i\, x^{(i)} - \theta < 0 \text{ or } \sum_{i=0}^{d} w_i\, x^{(i)} < 0 \end{cases} = \text{sign}(\mathbf{w}^T\mathbf{x} - \theta)$$



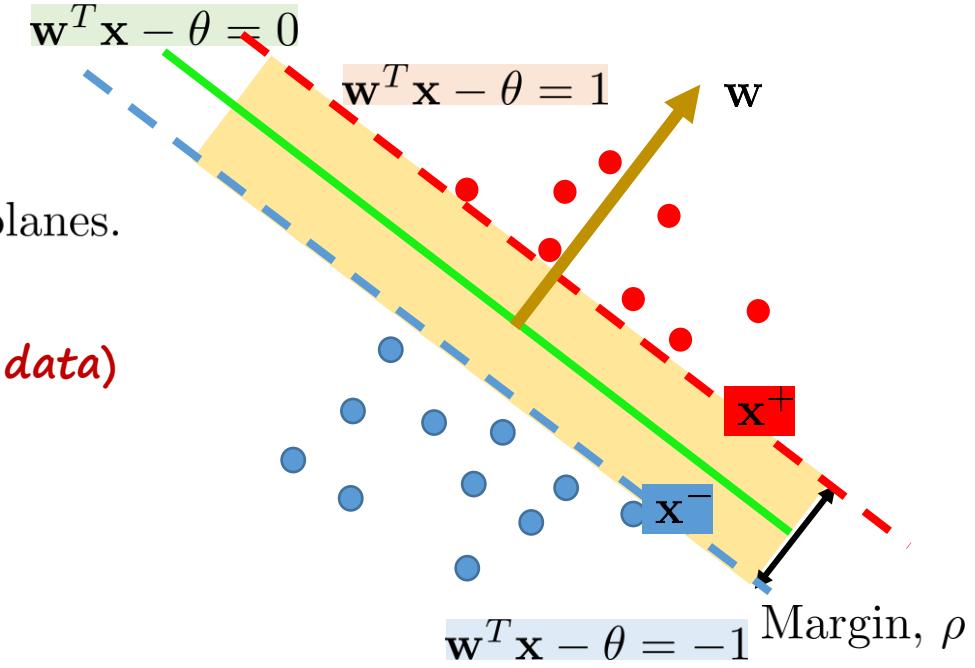
- Support vector is the data-point for each class closest to the hyper-plane.
- Margin, denoted by ρ, is the distance between the support vectors.
- We wish to find the hyper-plane for which this margin is maximized.
- Maximization depends on the support vectors; other training data points are ignorable.
- Hard SVM: We require all the training points to be classified correctly. (No misclassification)

Support Vector Machines (SVM)

Hard SVM – Mathematical Formulation:

- Since the support vectors are equidistant from the decision boundary.
- Define:
  - Plus plane: $\mathbf{w}^T\mathbf{x} - \theta = 1$
  - Minus plane: $\mathbf{w}^T\mathbf{x} - \theta = -1$

Note: $\mathbf{w}$ is perpendicular to both the decision boundary and plus/minus planes.



- Take $\mathbf{x}^-$ as any point on the minus plane. *(Not necessarily from the data)*
- Take $\mathbf{x}^+$ point on the plus plane that is closest to $\mathbf{x}^-$.
- We can relate $\mathbf{x}^-$, $\mathbf{x}^+$ and $\mathbf{w}$ as

$$\mathbf{x}^+ = \mathbf{x}^- + \beta\mathbf{w}$$

- Since the margin ρ is the distance between plus and minus planes:

$$\rho = \|\mathbf{x}^+ - \mathbf{x}^-\| = \|\beta\mathbf{w}\| = \beta\|\mathbf{w}\|$$

Support Vector Machines (SVM)

Hard SVM – Mathematical Formulation:

- Noting $\mathbf{x}^+ = \mathbf{x}^- + \beta\mathbf{w}$ $\quad \mathbf{w}^T\mathbf{x}^+ - \theta = 1$ $\quad \mathbf{w}^T\mathbf{x}^- - \theta = -1$

  we obtain

$$\mathbf{w}^T\mathbf{x}^+ - \theta = \mathbf{w}^T\mathbf{x}^- + \beta\mathbf{w}^T\mathbf{w} - \theta = 1$$

$$\beta\mathbf{w}^T\mathbf{w} = 2 \quad \Rightarrow \beta = \frac{2}{\mathbf{w}^T\mathbf{w}} = \frac{2}{\|\mathbf{w}\|^2}$$

- Since $\rho = \|\mathbf{x}^+ - \mathbf{x}^-\| = \|\beta\mathbf{w}\| = \beta\|\mathbf{w}\|$, we have

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

- Now we have expressed margin in terms of $\mathbf{w}$, we can now formulate optimization problem.
- We note here that the maximization of $\rho = \frac{2}{\|\mathbf{w}\|}$ is equivelent to the minimization of $\|\mathbf{w}\|$ or $\|\mathbf{w}\|^2$.
- Interpretation: We want to minimize the norm of the vector $\mathbf{w}$ (normal to the decision boundary hyper-plane).

Support Vector Machines (SVM)

Hard SVM – Optimization Problem Formulation:

- We want to minimize $\|\mathbf{w}\|^2$.
- At the same time, we should ensure that the training data-points are classified correctly.
- For training data $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$

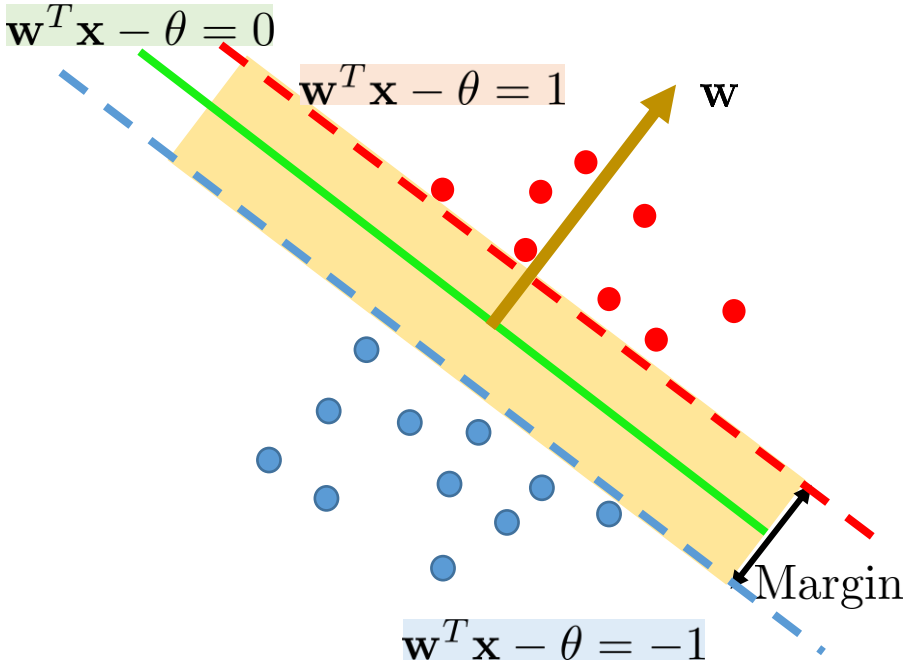
$\mathbf{w}^T\mathbf{x}_i - \theta \geq 1$ if $y_i = 1$ (Plus plane)

$\mathbf{w}^T\mathbf{x}_i - \theta \leq -1$ if $y_i = -1$ (Minus plane)

$\Rightarrow y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1$

- This can be formulated as a following optimization problem.

$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1 \quad i = 1, 2, \ldots, n \end{aligned}$$

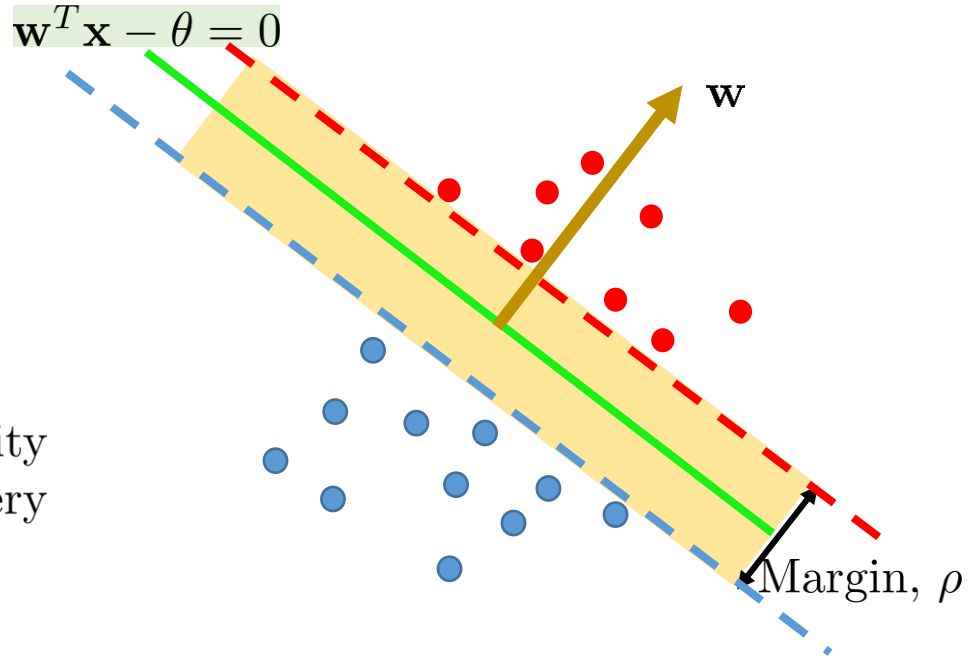


Support Vector Machines (SVM)

Hard SVM – Optimization Problem Formulation:

- Optimization problem for learning SVM parameters:

$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1 \quad i = 1, 2, \ldots, n \end{aligned}$$



Interpretation: Minimize norm of the normal vector defining the separating hyperplane while ensuring all the training points are classified correctly.

- We have a qudratic (convex) objective function and n linear inequality constraints; optimization problem, aka quadratic program (QP), is very well studied and can be solved very efficiently.
- Gives us maximum margin classifier, hyper-plane, $\mathbf{w}^T\mathbf{x} - \theta = 0$.
- Classification margin is $\rho = \frac{2}{\|\mathbf{w}\|}$.

Support Vector Machines (SVM)

Formulation of Dual Problem using Lagrange Multipliers:

- For the following convex optimization problem:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f_o(\boldsymbol{\theta}) \\ \text{subject to} \quad & f_i(\boldsymbol{\theta}) \leq 0, \quad i = 1, 2, \ldots, m \end{aligned}$$

- Convex optimization problem: if f_o and $f_i, \quad i = 1, 1, 2, \ldots, m$ are convex.

- The Lagrangian dual problem is formulated as:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \inf_{\boldsymbol{\theta}} \left(f_o(\boldsymbol{\theta}) + \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{\theta}) \right) \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, 2, \ldots, m \end{aligned}$$

- $\mathcal{L}(\boldsymbol{\alpha})$ is the Lagrange dual function.

- Connection between two problems: Under certain conditions (Slater's constraint), the optimal value is same for both the problems.

Support Vector Machines (SVM)

Formulation of Dual Problem using Lagrange Multipliers:

- For optimization problem,

$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \tfrac{1}{2}\,\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) - 1 \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned}$$

- We formulate Lagrangian as

$$\mathcal{L}(\mathbf{w}, \theta, \boldsymbol{\alpha}) = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i \big(y_i(\mathbf{w}^T\mathbf{x}_i - \theta) - 1\big)$$

- $\alpha_i \geq 0$ is the Lagrange multiplier associated with the i-th constraint.

Support Vector Machines (SVM)

Formulation of Dual Problem using Lagrange Multipliers:

- Now we compute derivative of Lagrangian with respect to $\mathbf{w}$ and θ.

Lagrangian

$$\mathcal{L}(\mathbf{w}, \theta, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i \big(y_i(\mathbf{w}^T\mathbf{x}_i - \theta) - 1\big)$$

- Partial derivative of Lagrangian with respect to $\mathbf{w}$.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- Substituting it equal to zero yields:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Interpretation: Normal vector $\mathbf{w}$ is a linear combination of the data points.

- Partial derivative of Lagrangian with respect to θ and equating it to zero.

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

Support Vector Machines (SVM)

Formulation of Dual Problem using Lagrange Multipliers:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \qquad \frac{\partial \mathcal{L}}{\partial \theta} = \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \mathcal{L}(\mathbf{w}, \theta, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i \big(y_i(\mathbf{w}^T\mathbf{x}_i - \theta) - 1\big)$$

- We can combine these to obtain:

$$\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \; - \; \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i y_i \alpha_j y_j \mathbf{x}_j^T\mathbf{x}_i + \theta\sum_{i=1}^{n}\alpha_i y_i + \sum_{i=1}^{n}\alpha_i$$

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

- This is the Lagrange dual function.
- This is a concave function.

- We have minimzed over $\mathbf{w}$ and θ and now we maximize the Lagrange dual function with respect to $\boldsymbol{\alpha}$.

Support Vector Machines (SVM)

Formulation of Dual Problem using Lagrange Multipliers:

- Dual optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned}$$

- This is also a QP.
- It turns out that $\alpha_i = 0$ when $\mathbf{x}_i$ is not a support vector.
- If prior information about support vectors is known, this can be used to reduce the complexity of the optimization problem.
- We use SV to denote the set of support vectors, that is, the points in the feature space for which α_i is non-zero.

Support Vector Machines (SVM)

Formulation of Dual Problem using Lagrange Multipliers:

- Maximization of Lagrange dual function gives us $\boldsymbol{\alpha}$ using which we can determine $\mathbf{w}$ as

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i$$

- Here we have only used the support vectors.

- Using this reformulation, we can write the decision boundary as

$$\sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} - \theta = 0$$

- For any test point, we only need to compute the inner product with the support vectors.

Remark:

- We do not need to determine $\mathbf{w}$ explicitly. We only need support vectors and associated α_i.

Support Vector Machines (SVM)

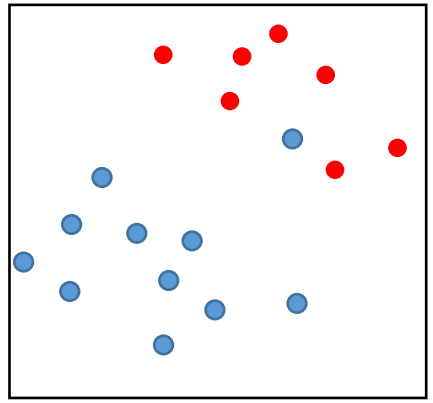
Hard SVM – Summary:

- We formulated optimization problem to learn maximum margin classifier using the training data.
- Since the optimization problem constraints represent the misclassification on the training data, the error loss on the training data is zero and therefore we this is referred to as Hard Margin SVM or Hard SVM.

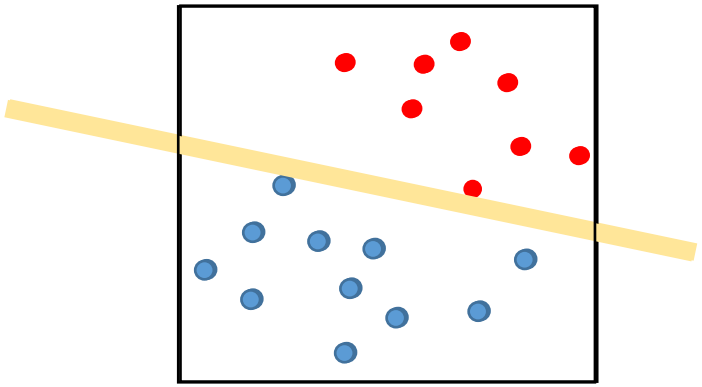
Issues with hard SVM:

It can overfit very easily and therefore cannot generalize.

- Due to outliers or noisy/erroneously label observation, optimization problem can be infeasible, that is no solution.
- Even the outliers within the boundaries can influence the margin.



Infeasible; due to one noisy or incorrectly labeled point.



Decision boundary and margin changed; due to one noisy or incorrectly labeled point.

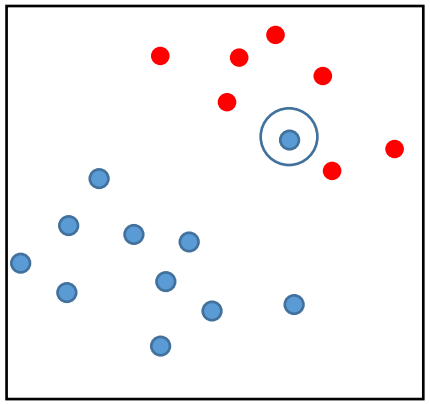
Outline

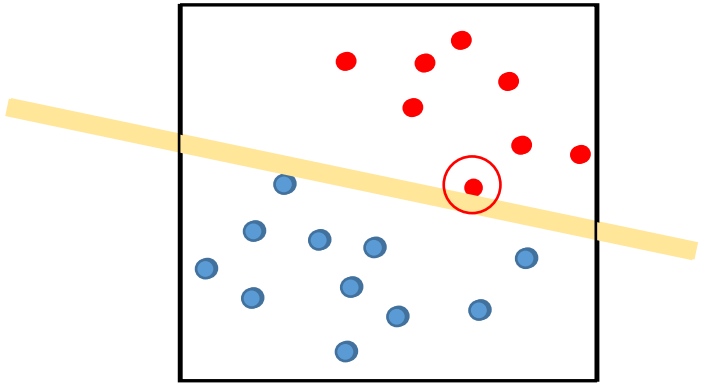
- Support Vector Machines (SVM) Overview
- Hard SVM
- Soft SVM
- Kernel Trick
- OvR SVM classifier for Multi-class classification

Support Vector Machines (SVM)

Soft SVM:

- Due to outliers or noisy/erroneously labeled observations, optimization problem to find hard SVM
  - can be infeasible.
  - does not return the maximum margin classifier.





- Q. What's the issue with hard SVM?
- A. The constraints in the following optimization problem must be satisfied.

$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1 \quad i = 1, 2, \ldots, n \end{aligned}$$

Zero loss (no misclassification) on the training data

Support Vector Machines (SVM)

Soft SVM:

Q. How do we learn maximum margin classifier to handle the noise in training data?

A. We allow misclassification of difficult (close to the boundary) or noisy examples and therefore allowing the margin to be soft, resulting in soft SVM.

Candidate 1:

- Minimize $\|\mathbf{w}\|^2$ and loss function (number of misclassifications on the training set).

  How do we minimize two quantities at the same time?

- We can minimize the sum of $\|\mathbf{w}\|^2$ and number of missclassfications, i.e.,

$$\underset{\mathbf{w},\theta}{\text{minimize}} \quad \|\mathbf{w}\|^2 + C\left(\text{no. of misclassifications on training set}\right)$$

$C \geq 0$ is the trade-off parameter that quantifies the relative trade-off.

1/0 loss function.

Issues with this:

1) *not a quadratic program, and*
2) *does not consider whether the misclassification (loss) is due to the points near the boundary or far from the boundary.*

Support Vector Machines (SVM)

Soft SVM – Optimization Problem:

Candidate 2:

- Minimize $\|\mathbf{w}\|^2$ and distance of the error points to their correct place.
- We achive this by introducing slack variables to allow misclassification of difficult or noisy data points.

Hard SVM – Optimization Problem:

$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1 \quad i = 1, 2, \ldots, n \end{aligned}$$

Soft SVM – Optimization Problem with Slack Variables:

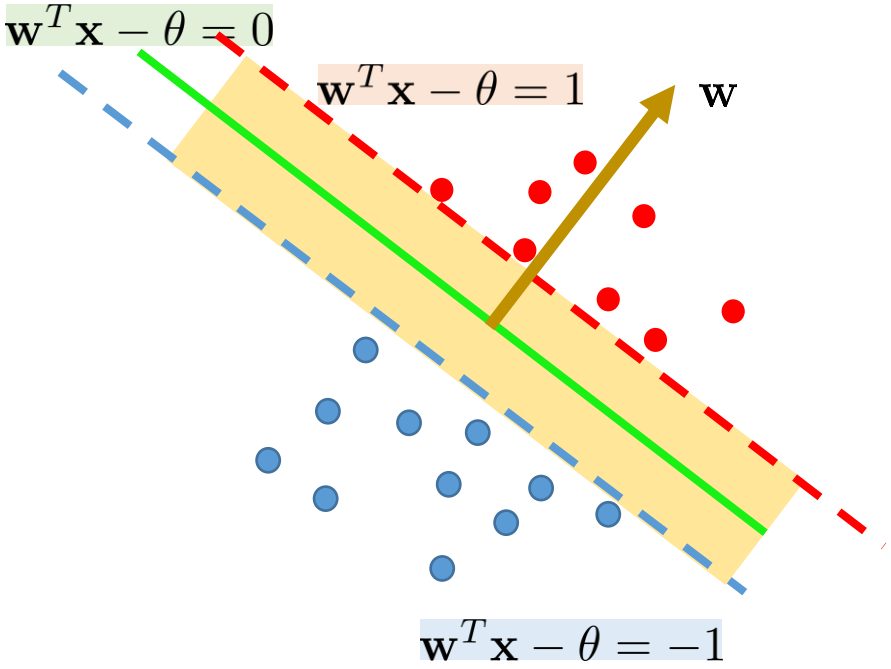
$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq (1 - \xi_i) \quad i = 1, 2, \ldots, n \\ & \xi_i \geq 0 \quad i = 1, 2, \ldots, n \end{aligned}$$

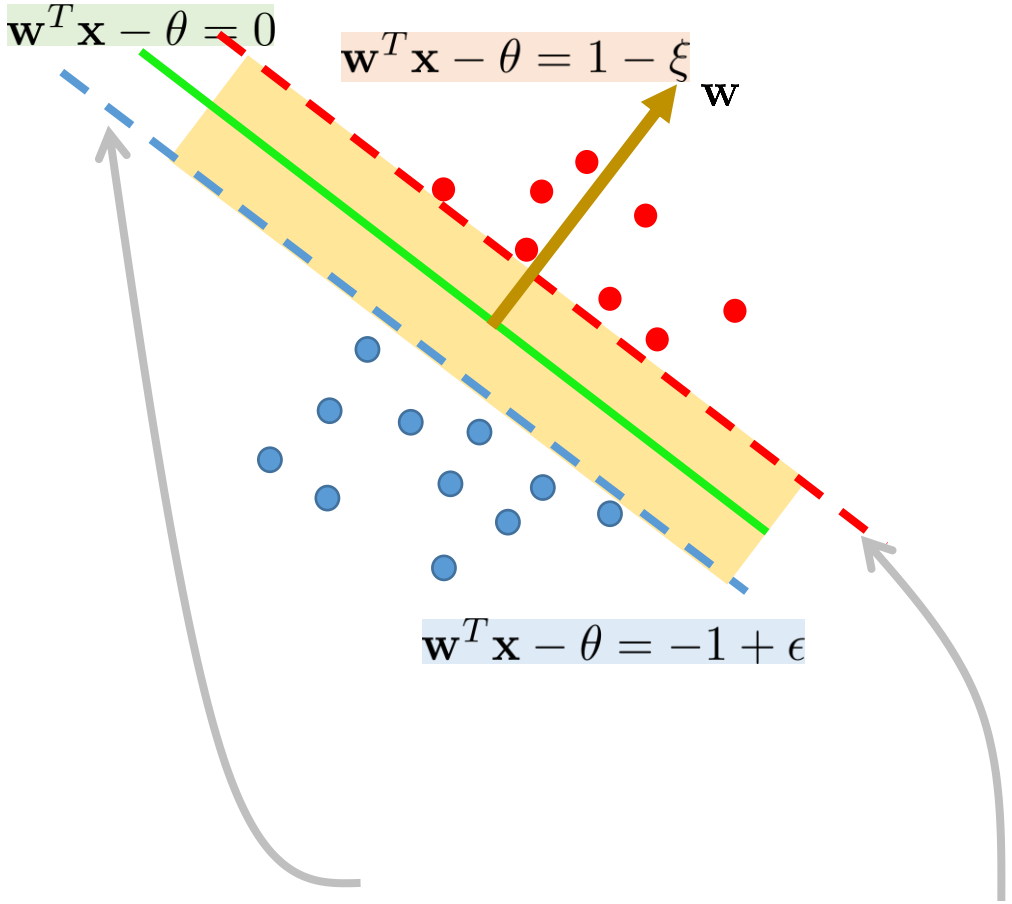
We dislike the points to be misclassified.

We **allow** the points to be misclassified.

Support Vector Machines (SVM)

Soft SVM – Visualization:





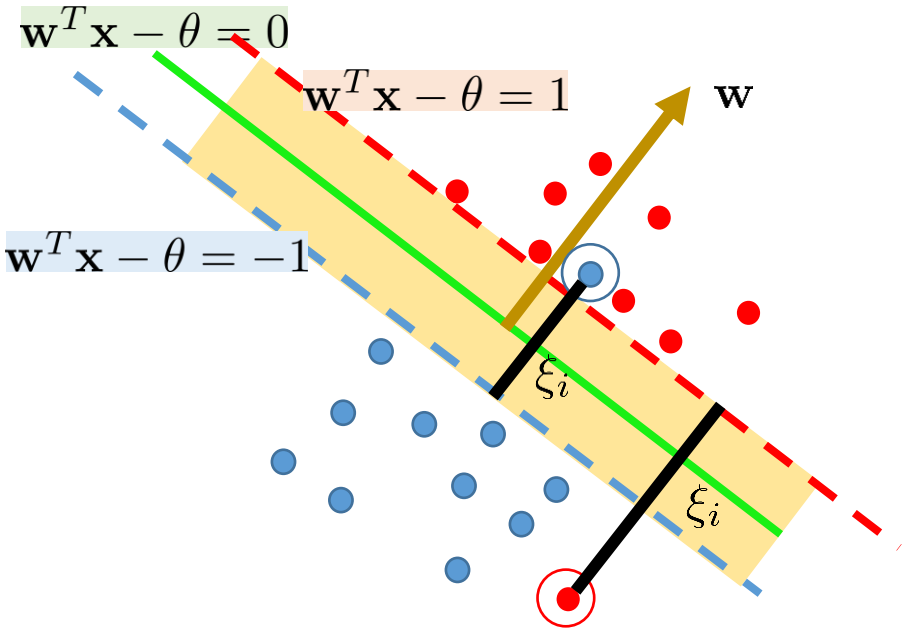
We allow the points to violate these support vectors.

In fact, we have different ξ_i for each data point.

Support Vector Machines (SVM)

Soft SVM – Visualization:

- As an example, consider the following data. We do not have a linear separability.



- Hard SVM will not be able to find the separating hyper-plane.
- Becasue we satisfy the following constraint for hard SVM.

$$y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1 \quad \Rightarrow \quad \begin{matrix} \mathbf{w}^T\mathbf{x}_i - \theta \geq 1 \text{ if } y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i - \theta \leq -1 \text{ if } y_i = -1 \end{matrix}$$

- The constraint is satisfied by all but the encircled data points.
- We have allowed the following constraint in soft SVM.

$$y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq (1 - \xi_i) \Rightarrow \begin{matrix} \mathbf{w}^T\mathbf{x}_i - \theta \geq 1 - \xi_i \text{ if } y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i - \theta \leq -1 + \xi_i \text{ if } y_i = -1 \end{matrix}$$

- $\xi_i \geq 0$ indicated for encircled points.
- Using ξ_i for each data-point, we have allowed the points to be misclassified by the separating hyper-plane.

Support Vector Machines (SVM)

Soft SVM – Optimization Problem:

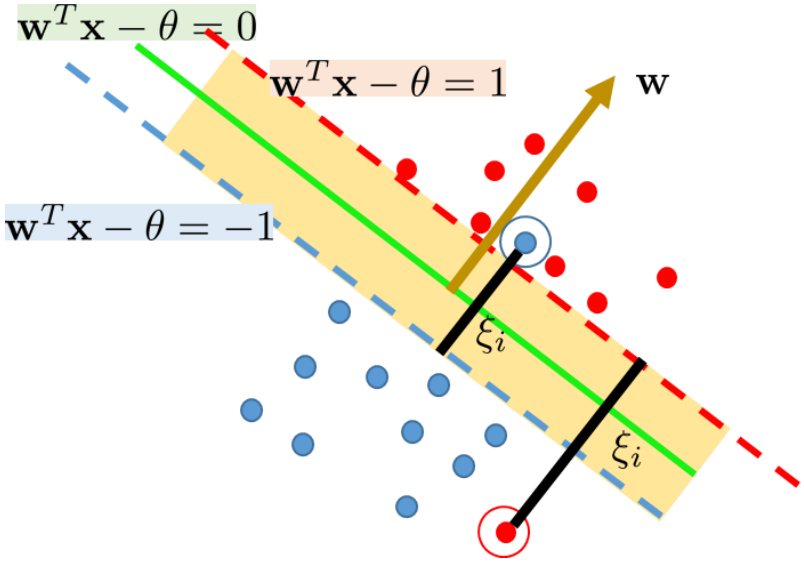
- Revisit the problem:

$$\underset{\mathbf{w},\theta}{\text{minimize}} \quad \|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right)$$

$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq (1 - \xi_i) \qquad i = 1, 2, \ldots, n$$

$$\xi_i \geq 0 \qquad i = 1, 2, \ldots, n$$

- Optimization problem is still a quadratic program in $d + 1 + n$ variables. For hard SVM, QP had $d + 1$ variables.
- The slack variable ξ_i allows the input $\mathbf{x}_i$ to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack".
- $\xi_i \geq 0$ is the distance from the correct boundary.
- $\xi_i = 0$ for the points that are on the correct side of the separating hyperplane. This is ensured by non-negativity constraint on ξ_i.
- The misclassified points far from the boundary are penalized more compared to the points close to the boundary as we have incorporated the distances in the objective function.



Support Vector Machines (SVM)

Soft SVM – Optimization Problem:

- Revisit the problem:

$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq (1 - \xi_i) \qquad i = 1, 2, \ldots, n \\ & \xi_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned}$$

- Parameter C maintains the relative trade-off between the importance of maximizing the margin and fitting the training data.

- If C is very large, the SVM becomes very strict and tries to get all points to be classified correctly, that is, on the correct side of the hyperplane.

- If C is very small, the SVM may allows some points to be misclassified to obtain a simpler (i.e., lower $\|\mathbf{w}\|^2$) solution.

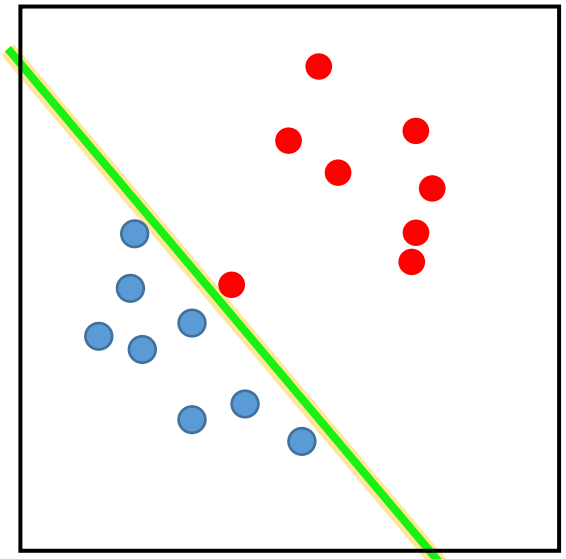
Source: https://www.cs.cornell.edu/courses/cs4780/2018sp/lectures/lecturenote09.html

Support Vector Machines (SVM)

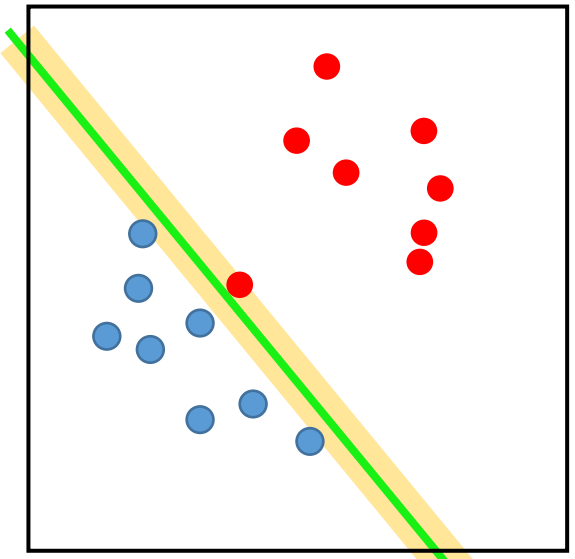
Soft SVM – Dislike of points to be misclassified vs Importance of margin

The larger the value of C, the more we dislike misclassification.

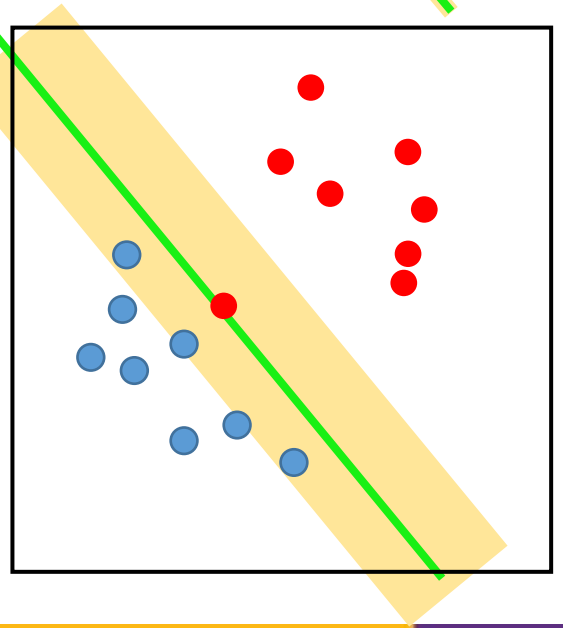
- $C = 100$



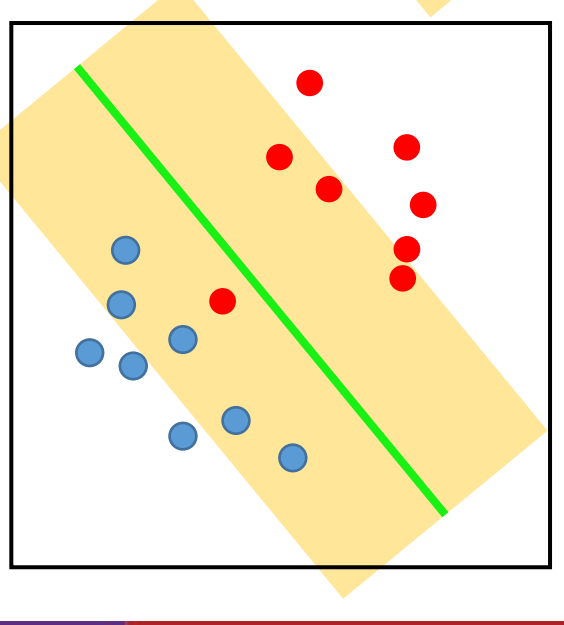
- $C = 10$



- $C = 1$



- $C = 0.1$



Support Vector Machines (SVM)

Soft SVM – Optimization Problem Reformulation:

- We could solve constrained QP. We can also reformulate the problem as an unconstrained optimization problem.
- For each $i = 1, 2, \ldots, n$, we have $y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq (1 - \xi_i) \quad \xi_i \geq 0$
- When $y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq 1$ (no misclassification), we have $\xi_i = 0$.
- When $y_i(\mathbf{w}^T\mathbf{x}_i - \theta) < 1$, we have $y_i(\mathbf{w}^T\mathbf{x}_i - \theta) = 1 - \xi_i \quad \Rightarrow \xi_i = 1 - y_i(\mathbf{w}^T\mathbf{x}_i - \theta)$
- Interpretation:
  - No penalty if data point is classifier correctly.
  - Penalty = distance from the correct boundary if point is misclassified.
- Combining these two, we can express ξ_i as

$$\xi_i = \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - \theta)\right)$$

Support Vector Machines (SVM)

Soft SVM – Optimization Problem Reformulation:

- We use this to obtain equivalent formulation of our objective function:

$$\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \quad \rightarrow \quad \|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - \theta)\right)$$

- Consequently, we have an unconstrained minimization problem in $d+1$ variables:

$$\underset{\mathbf{w},\theta}{\text{minimize}} \quad \|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - \theta)\right)$$

- This looks familiar.

Square of norm of the weights; we are restricting the norm of the weights to grow.

*We had been doing this before; **L_2 regularization!***

*This is sort of a **loss function:** It quantifies the misclassification distance for each point.*

*This loss is known as '**Hinge Loss**'.*

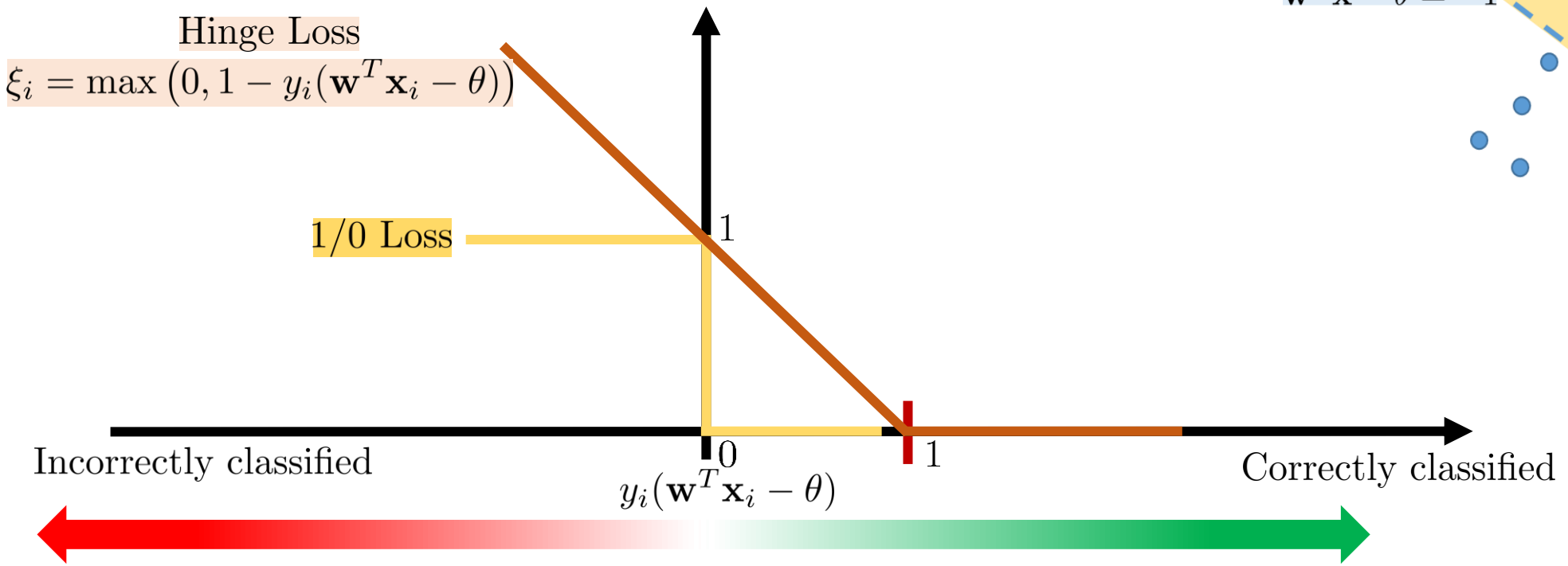
- The formulation is similar to the one we can develop for regularized logistic regression.
- Here we have hinge loss instead of the logistic loss.
- We can use gradient descent to minimize this.

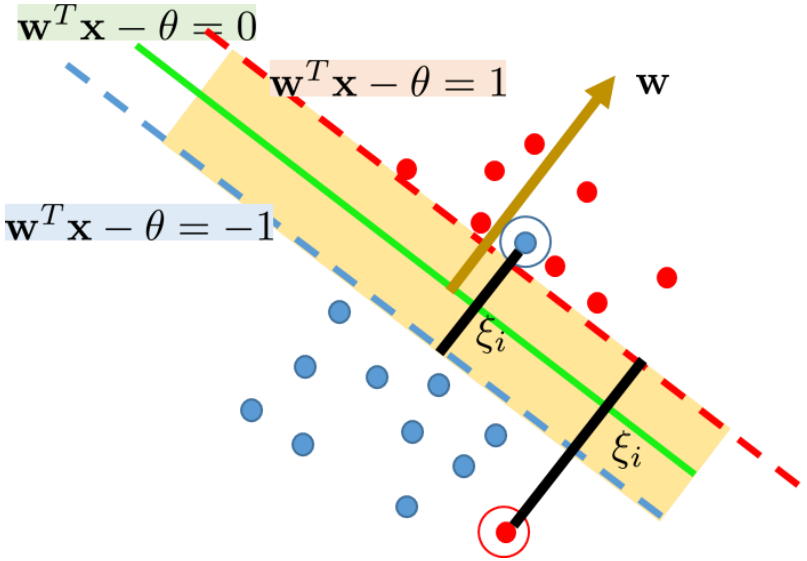
Support Vector Machines (SVM)

Hinge Loss vs 0/1 Loss:

- 1/0 Loss: no misclassification if $y_i(\mathbf{w}^T\mathbf{x}_i - \theta)) \geq 0$
- Hinge Loss: Penalize the point if

$$y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \leq 1$$





Outline

- Support Vector Machines (SVM) Overview
- Hard SVM
- Soft SVM
- Kernel Trick
- OvR SVM classifier for Multi-class classification

Support Vector Machines (SVM)

The Kernel Trick – Overview:

"If you only do what you can, you will never be more than what you are now."

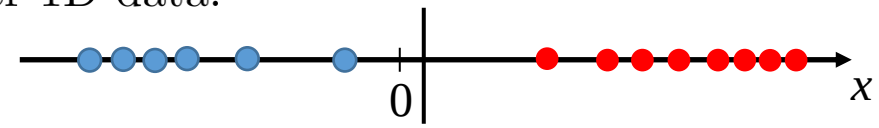
Master Shifu

(KungFu Panda)

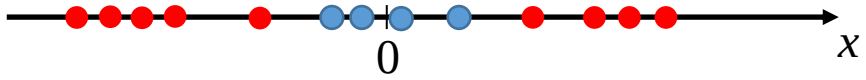
Support Vector Machines (SVM)

The Kernel Trick – Overview (Recap):

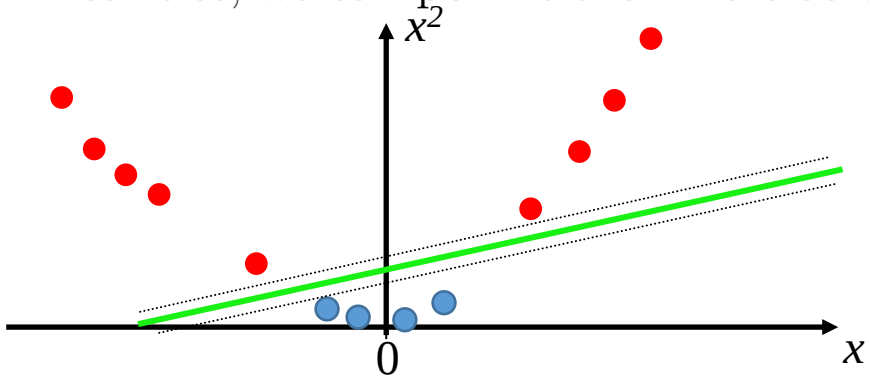
- For simplicity, consider 1D data.



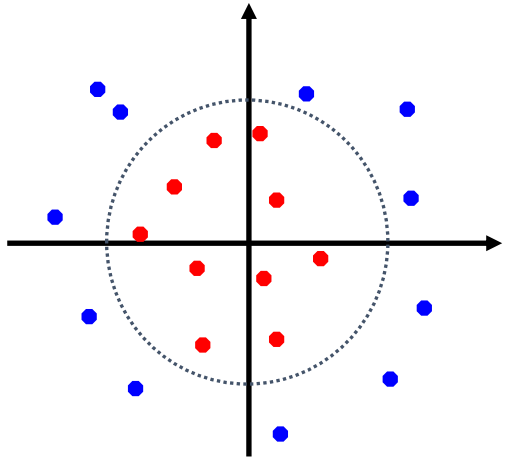
- How can we use SVM for this data?

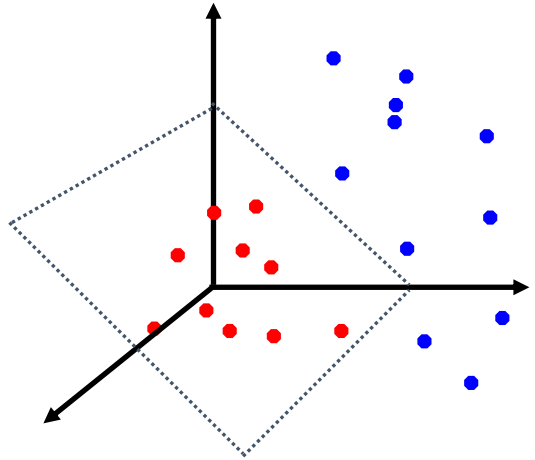


- Since the use of polynomial (non-linear)functions facilitated us in linear regression to model non-linearities, we can permit them here as well.



- 2D to 3D.



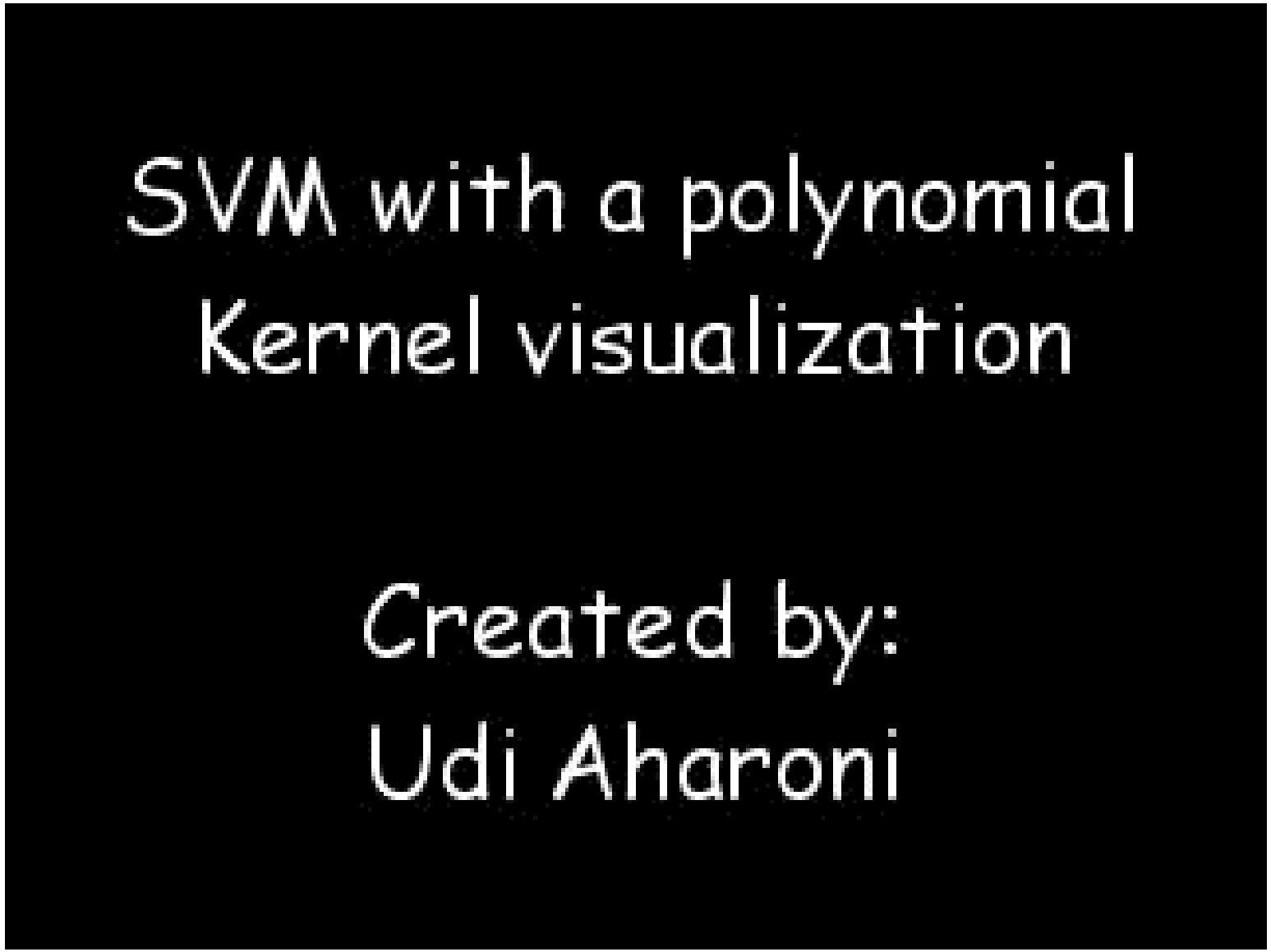


General Idea:

Project to original feature space to higher dimensional space to make the classes linearly separable. This mapping function is referred to as the kernel trick and the mapping function is known as the kernel function.

Support Vector Machines (SVM)

The Kernel Trick – Overview demo:



Support Vector Machines (SVM)

The Kernel Trick – Overview:

- We can project the features to a higher dimensional space to make the data linearly separable (increasing d).
  - Trade-off: Computationally expensive or intractable as we need to solve optimization problem in higher dimensional space in order to find maximum margin classifier.
- The so-called 'The Kernel Trick' allows us to keep computational tractability.
- First, we need to understand the computations we require to determine SVM classifier.

Lagrange Dual Optimization Problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned}$$

Decision Boundary:

$$\sum_{i \in \text{SV}} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} - \theta = 0$$

We require the computation of these inner products.

Support Vector Machines (SVM)

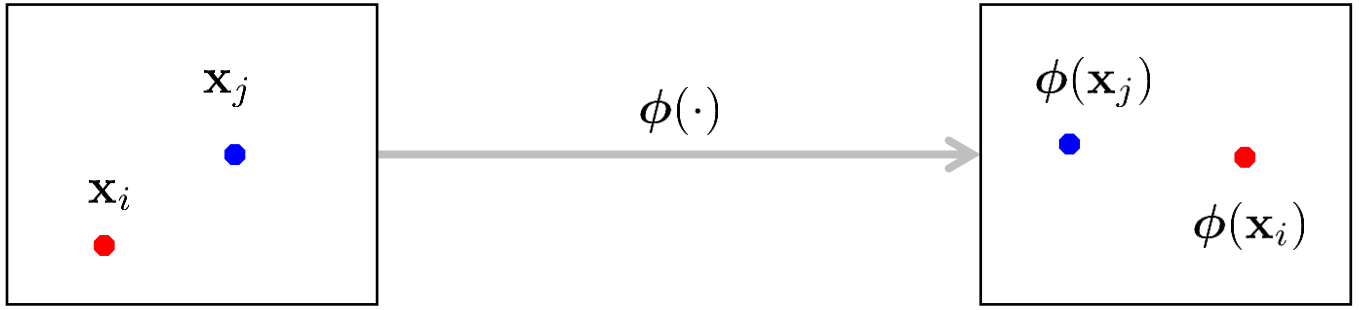
The Kernel Trick – Overview:

- The computation of inner products depends on the dimensionality of the feature space.
- If we can find a computationally efficient method to compute the inner products, we can learn SVM classifier in higher dimensional space.
- This efficient computation of inner products is, in fact, enabled by the kernel trick.
- Idea: We choose mapping to high dimensional space in a way that supports fast computation of scalar products.

Support Vector Machines (SVM)

The Kernel Trick – Overview:

- We use $\phi(\mathbf{x})$ to denote the mapping to the higher dimensional space.



low dimensional space

high dimensional space

- We need to compute the inner product in the high dimensional space:

$$\phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j)$$

- For certain (not all) mapping functions, this inner product can be computed as a simple operation in the low dimensional space using a function known as Kernel function, that is,

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

Computation using kernel

Computationally expensive

Support Vector Machines (SVM)

The Kernel Trick – Example:

- For $d = 2$, we have a two dimensinal feature vector $\mathbf{x} = [x^{(1)} \quad x^{(2)}]$.

- We project to 6 dimensional space by including quadratic features using the mapping function:

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 & \left(x^{(1)}\right)^2 & \sqrt{2}\, x^{(1)} x^{(2)} & \left(x^{(2)}\right)^2 & \sqrt{2}\, x^{(1)} & \sqrt{2} x^{(2)} \end{bmatrix}$$

- Inner product in higher dimensional space $\boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)$ can be computed using the following kernel function:

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

- Thus, a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ enables us to compute inner products in high dimensional space efficiently using the data points in the low dimensional space.

Support Vector Machines (SVM)

The Kernel Trick – Examples of Kernel Functions:

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
  - Mapping function is feature itself: $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$.

- Polynomial of power m: $K(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \mathbf{x}_i^T \mathbf{x}_j\right)^m$.
  - Mapping function: $\boldsymbol{\phi}(\mathbf{x})$ degree m polynomial of the features.

    mapping from d dimensional space to $\binom{d+m}{m}$ dimensional space.

- Gaussian (Radial-basis) function:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \equiv \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

  - Mapping function: $\boldsymbol{\phi}(\mathbf{x})$ is infinite dimensional and every point is mapped to a Gaussian.

- It must be noted that the intrinsic dimensionality of the data remains d in the high dimensional space.

Support Vector Machines (SVM)

The Kernel Trick – More on Kernel Functions:

- For a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, checking manually $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)$ can be complicated and tedious.
- Mercer's Theorem: We can use every positive semi-definite symmetric function as a kernel.
- For symmetric function such that $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$, the kernel is positive semi-definite (PSD) if the following gram matrix is PSD.

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

Support Vector Machines (SVM)

The Kernel Trick – Optimization Problem:

- Dual optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j) \\ \text{subject to} \quad & \alpha_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned}$$

- Incorporating kernel function yields:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & \alpha_i \geq 0 \qquad i = 1, 2, \ldots, n \end{aligned}$$

- Consequently, we have the decision boundary as:

$$\sum_{i \in \text{SV}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - \theta = 0$$

- For any test point $\mathbf{z}$, label is 1 if

$$\sum_{i \in \text{SV}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) - \theta \geq 0$$

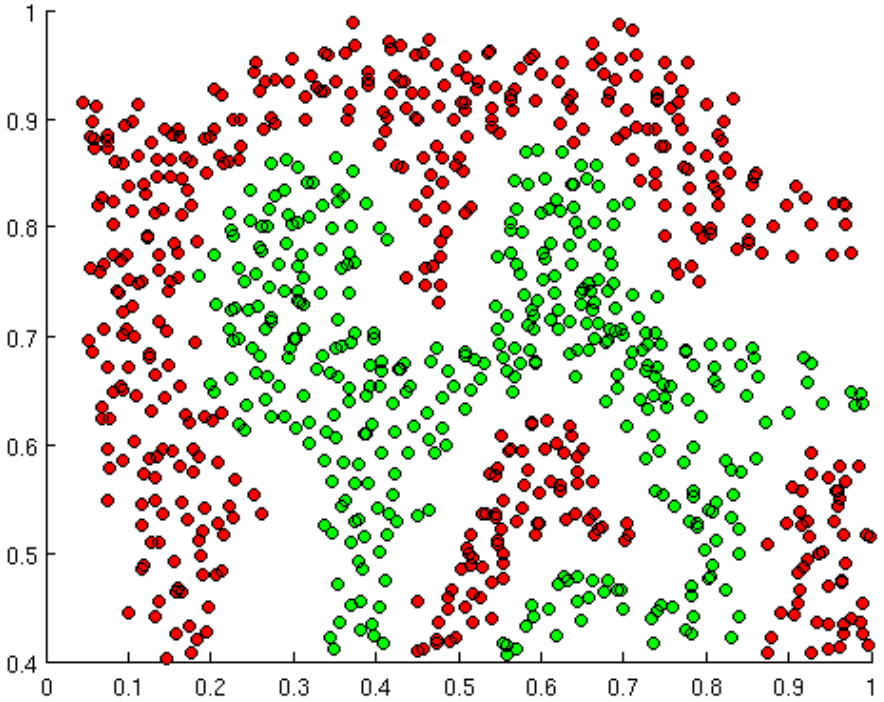
We are living in a low dimensional space but we are manipulating the data in high dimensional space. We obtain a non-linear decision boundary in the low dimensional space.

Support Vector Machines (SVM)

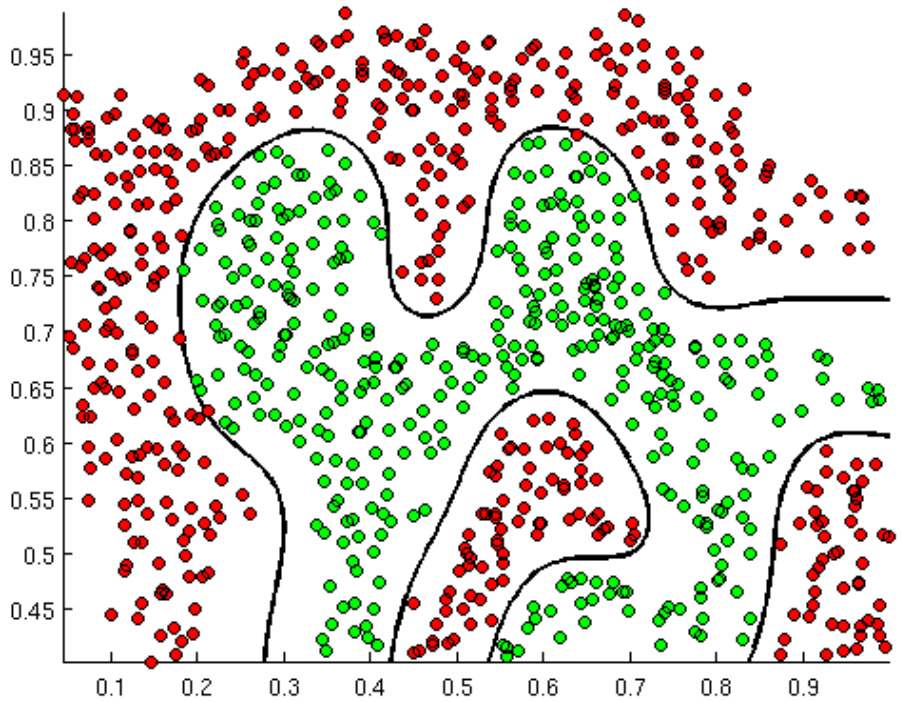
The Kernel Trick – Example:

- Gaussian (Radial-basis) function:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \equiv \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$



$\gamma = 100$

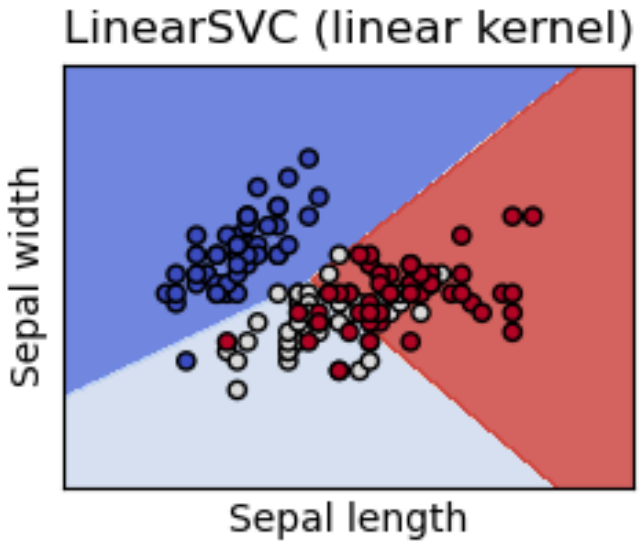


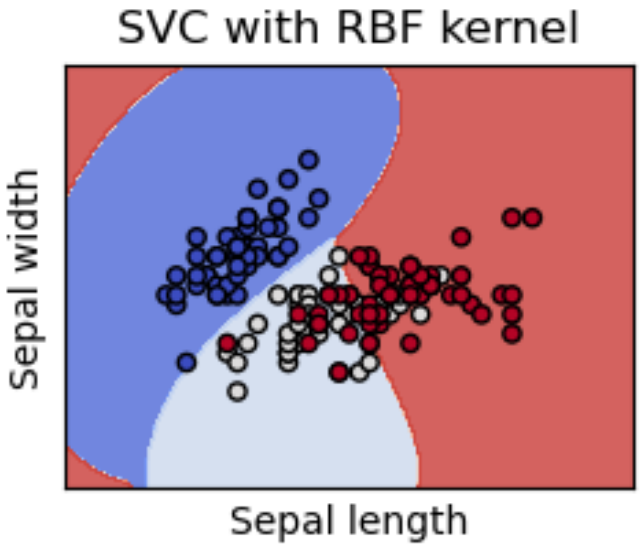
LUMS
A Not-for-Profit University

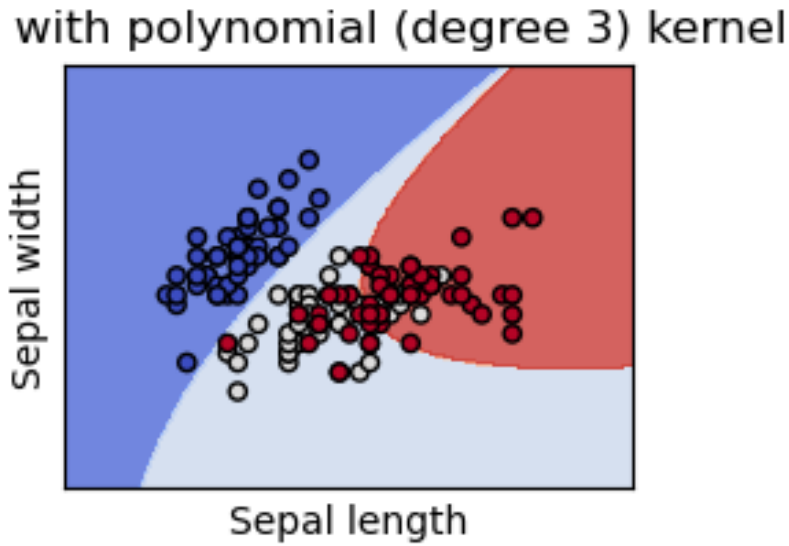
Source: http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html

Support Vector Machines (SVM)

The Kernel Trick – Example:







Source: https://scikit-learn.org/stable/modules/svm.html

Outline

- Support Vector Machines (SVM) Overview
- Hard SVM
- Soft SVM
- Kernel Trick
- OvR SVM classifier for Multi-class classification

Support Vector Machines (SVM)

Multi-Class (Multinomial) Classification:

- $\mathcal{Y} = \{0, 1, 2, \ldots, M-1\}$ (M-class classification)

Build a one-vs-rest (OvR) classifier:

- Train M different SVM classifiers $h_0(\mathbf{x}), h_1(\mathbf{x}), \ldots, h_{M-1}(\mathbf{x})$.
- Classifier $h_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} - \theta_i$ is trained to classify if $\mathbf{x}$ belongs to i-th class or not.
- For a new test point $\mathbf{z}$, get scores for each classifier, that is, $s_i = h_i(\mathbf{z})$.
- s_i represents the classification margin of the test point from the boundary separating i-th class and the rest of the classes.
- Predict the label as $\hat{y} = \max_{i=0,1,2,\ldots,M-1} s_i$

Support Vector Machines (SVM)

Summary:

- Support Vector Machine classifiers are very widely used.

- SVM in standard setting assumes the data is linearly separable and it locates a separating hyperplane in the feature space which can be used to classify points.

- Hyper-plane (decision boundary) can be determined by solving convex optimization problem.

- Hard SVM can be infeasible due to noisy or erroneous training data; Soft SVM version allows the points to be misclassified on the training data.

- To handle non-linear separability, we project the data to higher dimensional space.
  - Using the kernel trick, we do not need to project the data explicitly,
  - We only specify a kernel function and incorporate it in the optimization problem such that the kernel function computes the inner product effectively in the high dimensional space.

- For classification, we only need to store the support vectors and associated weights.

- Issues: Choice of kernel function and kernel parameters, choice of parameter C in soft SVM requires cross-validation, computational complexity is significant for large training data.

Support Vector Machines (SVM)

References:

- KM – Section 14.5.2
- CB – Section 7.1
- https://www.cs.cornell.edu/courses/cs4780/2018sp/lectures/lecturenote09.html (Prof. Kilian Weinberger)
- https://scikit-learn.org/stable/modules/svm.html