ECE 1513: Introduction to Machine Learning

Lecture 6: Support Vector Machine

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

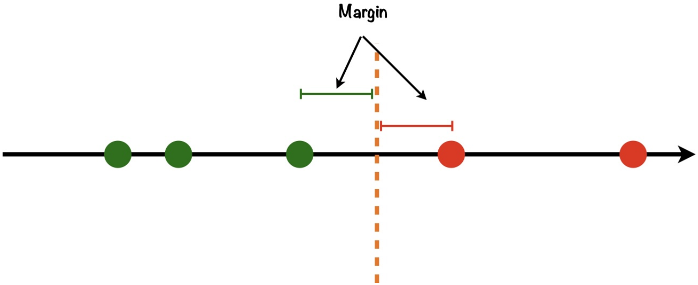
University of Toronto

Winter 2025

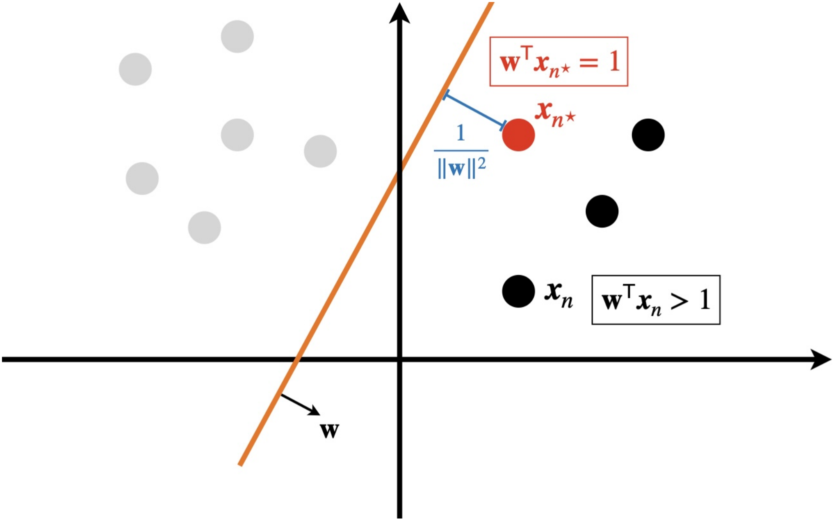
Quick Recap: *Classification with Confidence*

Classifying with Maximal Margin

We try to find linear classifier with maximal margin to support vectors



Quick Recap: *Support Vector Classifier*



Quick Recap: *SVC Formulation*

Training with maximal margin then looks like

$$\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|} \qquad \textit{subject to} \ \ v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n \geqslant 1 \textit{ for all } n$$

which can be alternatively written as

$$\textit{maximum margin} \rightarrow \boxed{\min_{\mathbf{w}} \|\mathbf{w}\|^2} \ \textit{subject to} \ \boxed{v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n \geqslant 1} \leftarrow \textit{no error}$$

Today's Agenda: *Support Vector Machine*

Today, we find the solution to SVC and through that introduce

Support Vector Machine and Concept of Kernels

In this way, we discuss the following topics

- *We find the solution to SVC*
  - ↳ *We review the method of Lagrange multipliers*
- *Understanding how SVC applies cross-validation*
- *Extend the idea of SVC to nonlinear patterns via the kernel trick*
- *Support vector machines*
  - ↳ *Nonlinear classification*

Looking at SVC: *Constrained Optimization*

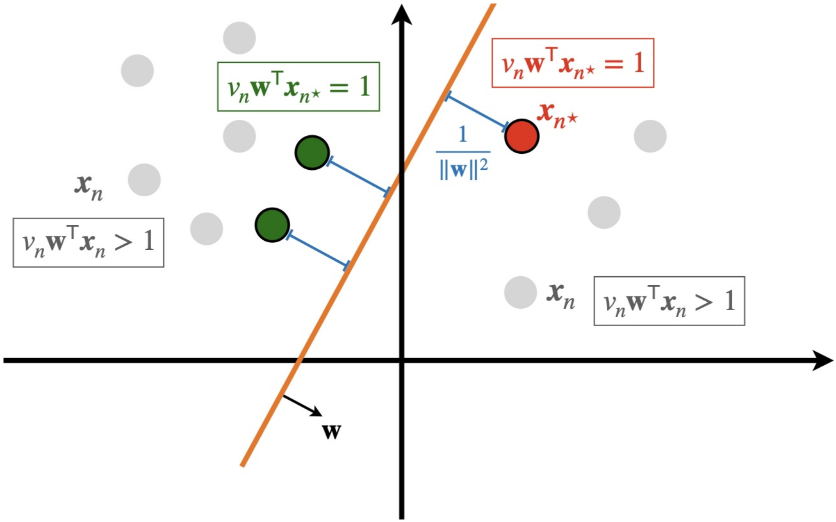
We ended up with the following training for SVC

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 \qquad \textit{subject to } \; v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n \geqslant 1 \textit{ for all } n$$

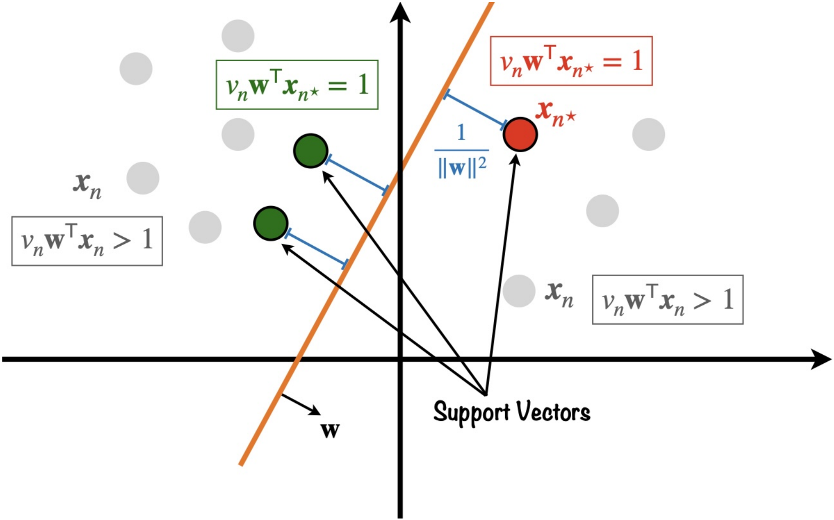
This is a constrained optimization

- *We want to find minimum of* $\|\mathbf{w}\|^2$
  - ↳ *This is the objective function*
  - ↳ *With no constraint it's obvious:* $\mathbf{w} = \mathbf{0}$
- *But, we are constrained by* $v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n \geqslant 1$ *for all samples*
  - ↳ *With* N *samples, we have* N *constraints*
  - ↳ *Obvious solution is not valid, since* $v_n \mathbf{0}^\mathsf{T} \boldsymbol{x}_n = 0 \ngeqslant 1$

Looking at SVC: *Visual Illustration*



Looking at SVC: *Visual Illustration*



Optimization with Inequality Constraints

We need to develop some approach that lets us solve

$$\min_{\mathbf{w}} f(\mathbf{w}) \qquad \textit{subject to } \; g_n(\mathbf{w}) \leqslant 0 \textit{ for all } n$$

In SVC, we have

$$f(\mathbf{w}) = \|\mathbf{w}\|^2$$
$$g_n(\mathbf{w}) = 1 - v_n \mathbf{w}^{\mathsf{T}} \boldsymbol{x}_n$$

? *How we can solve this constrained optimization?*

Lagrange Dual Objective

We make the dual Lagrangian function: *let*

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix}$$

be a vector of auxiliary variables; then, the dual Lagrangian is

$$\ell(\mathbf{w}, \boldsymbol{\lambda}) = f(\mathbf{w}) + \sum_{n=1}^{N} \lambda_n g_n(\mathbf{w})$$

where we have all Lagrange dual variables positive

$$\lambda_n \geqslant 0$$

Lagrange Dual Objective: *Key Property*

? *Why do we define such a function?*

Let's look at its maximum value over $\boldsymbol{\lambda}$ for an arbitrary $\mathbf{w}$

$$\ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = f\left(\mathbf{w}\right) + \sum_{n=1}^{N} \lambda_n g_n\left(\mathbf{w}\right)$$

and recall that $\lambda_n > 0$ for all n

- *At feasible points: $g_n\left(\mathbf{w}\right) \leqslant 0$ for all n*

$$\ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = f\left(\mathbf{w}\right) + \sum_{n=1}^{N} \lambda_n g_n\left(\mathbf{w}\right) \leqslant f\left(\mathbf{w}\right)$$

↳ *we could thus say that*

$$\max_{\boldsymbol{\lambda}} \ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = f\left(\mathbf{w}\right)$$

Lagrange Dual Objective: *Key Property*

? *Why do we define such a function?*

Let's look at its maximum value over $\boldsymbol{\lambda}$ for an arbitrary $\mathbf{w}$

$$\ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = f\left(\mathbf{w}\right) + \sum_{n=1}^{N} \lambda_n g_n\left(\mathbf{w}\right)$$

and recall that $\lambda_n > 0$ for all n

- *At infeasible points*: $g_n\left(\mathbf{w}\right) > 0$ *for all* n

$$\ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = f\left(\mathbf{w}\right) + \sum_{n=1}^{N} \lambda_n g_n\left(\mathbf{w}\right) \geqslant f\left(\mathbf{w}\right)$$

↳ *we could say that*

$$\max_{\boldsymbol{\lambda}} \ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = +\infty$$

Lagrange Dual Objective: *Key Property*

? *Why do we define such a function?*

So, we have

$$\max_{\boldsymbol{\lambda} \geqslant 0} \ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = \begin{cases} f\left(\mathbf{w}\right) & g_n\left(\mathbf{w}\right) \leqslant 0 \\ +\infty & g_n\left(\mathbf{w}\right) > 0 \end{cases}$$

This means that

$$\min_{\mathbf{w}} \max_{\boldsymbol{\lambda} \geqslant 0} \ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = \min_{\mathbf{w}} f\left(\mathbf{w}\right) \quad \textit{subject to} \quad g_n\left(\mathbf{w}\right) \leqslant 0$$

! *We can solve this unconstrained problem instead!*

Solving Dual Problem

Primal Problem

This alternative form describes the primal problem

$$P = \min_{\mathbf{w}} \max_{\boldsymbol{\lambda} \geqslant 0} \ell\left(\mathbf{w}, \boldsymbol{\lambda}\right)$$

However, it is easier to solve the dual problem

Dual Problem

Dual problem solves the unconstrained minimization first

$$D = \max_{\boldsymbol{\lambda} \geqslant 0} \min_{\mathbf{w}} \ell\left(\mathbf{w}, \boldsymbol{\lambda}\right)$$

Solving Dual Problem: *Strong Duality*

The key property is that the

dual value always bounds the primal from below, i.e., $D \leqslant P$

Strong Duality

Under Slater's conditions dual and primal values meet

- $f(\mathbf{w})$ *is convex*
- $g_n(\mathbf{w})$ *are all convex*
- *There is at least one* $\mathbf{w}_0$ *such that* $g_n(\mathbf{w}_0) < 0$ *for all* n

Solving Dual Problem: *KKT Conditions*

? *Say we have strong duality; then, how to find the solution?*

We need to find the point that satisfies KKT conditions

- *to be a stationary point*

$$\nabla_{\mathbf{w}} \ell \left(\mathbf{w}^\star, \boldsymbol{\lambda}^\star \right) = \mathbf{0}$$

- *to be feasible*

$$g_n \left(\mathbf{w}^\star \right) \leqslant 0 \qquad\qquad \lambda_n^\star \geqslant 0$$

- *to satisfy supplementary slackness*

$$\lambda_n^\star g_n \left(\mathbf{w}^\star \right) = 0$$

Solving Dual Problem: *KKT Conditions*

! *The key point is supplementary slackness*

$$\lambda_n^\star g_n\left(\mathbf{w}^\star\right) = 0$$

- *If the optimal point is on boundary; then, the dual variable is active*

$$g_n\left(\mathbf{w}^\star\right) = 0 \rightsquigarrow \lambda_n^\star > 0$$

- *But if it's not on the boundary; then, dual variable is inactive*

$$g_n\left(\mathbf{w}^\star\right) < 0 \rightsquigarrow \lambda_n^\star = 0$$

Example: *Minimizing Paraboloid I*

Let's find the solution to

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant 1$$

Before, we start

- *without constraint the optimal point is at* $\mathbf{w}^\star = \mathbf{0}$
- *it's already in the feasible region, since* $w_1^\star = 0 \leqslant 1$

Example: *Minimizing Paraboloid I*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant 1$$

Slater's conditions hold, so we find dual objective

$$\ell\left(\mathbf{w}, \lambda\right) = w_1^2 + w_2^2 + \lambda\left(w_1 - 1\right)$$

First, we find the stationary points

$$\nabla \ell\left(\mathbf{w}^\star, \lambda^\star\right) = \begin{bmatrix} 2w_1^\star + \lambda \\ 2w_2^\star \end{bmatrix} = \mathbf{0} \rightsquigarrow \begin{bmatrix} w_1^\star \\ w_2^\star \end{bmatrix} = \begin{bmatrix} -\lambda^\star/2 \\ 0 \end{bmatrix}$$

Example: *Minimizing Paraboloid I*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant 1$$

Slater's conditions hold, so we find dual objective

$$\ell(\mathbf{w}, \lambda) = w_1^2 + w_2^2 + \lambda(w_1 - 1)$$

Next, we check the feasibility

$$w_1^\star = -\frac{\lambda^\star}{2} \leqslant 1 \rightsquigarrow \lambda^\star \geqslant -2 \qquad \text{(Primal)}$$

$$\lambda^\star \geqslant 0 \qquad \text{(Dual)}$$

so, we should have $\lambda^\star \geqslant 0$

Example: *Minimizing Paraboloid I*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant 1$$

Slater's conditions hold, so we find dual objective

$$\ell(\mathbf{w}, \lambda) = w_1^2 + w_2^2 + \lambda(w_1 - 1)$$

Finally, we check the supplementary slackness

$$\lambda^\star (w_1^\star - 1) = 0 \rightsquigarrow -\lambda^\star \left(1 + \frac{\lambda^\star}{2}\right) = 0$$

Since we look for $\lambda^\star \geqslant 0$, the only solution is

$$\lambda^\star = 0$$

Example: *Minimizing Paraboloid I*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant 1$$

The dual solution is

$$\begin{bmatrix} w_1^\star \\ w_2^\star \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

and $\lambda^\star = 0$ which says that the dual variable is inactive

This makes sense, since the constraint does not really impact!

Example: *Minimizing Paraboloid II*

Now, let's solve this problem

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant -1$$

Before, we start

- *without constraint the optimal point is at* $\mathbf{w}^\star = \mathbf{0}$
- *it's infeasible, since* $w_1^\star = 0 \not\leqslant -1$

For this simple example, we can easily see that the solution is on boundary then

Example: *Minimizing Paraboloid II*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \ \textit{subject to} \ w_1 \leqslant -1$$

Slater's conditions hold, so we find dual objective

$$\ell(\mathbf{w}, \lambda) = w_1^2 + w_2^2 + \lambda(w_1 + 1)$$

First, we find the stationary points

$$\nabla \ell(\mathbf{w}^\star, \lambda^\star) = \begin{bmatrix} 2w_1^\star + \lambda \\ 2w_2^\star \end{bmatrix} = \mathbf{0} \rightsquigarrow \begin{bmatrix} w_1^\star \\ w_2^\star \end{bmatrix} = \begin{bmatrix} -\lambda^\star/2 \\ 0 \end{bmatrix}$$

Example: *Minimizing Paraboloid II*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant -1$$

Slater's conditions hold, so we find dual objective

$$\ell(\mathbf{w}, \lambda) = w_1^2 + w_2^2 + \lambda(w_1 + 1)$$

Next, we check the feasibility

$$w_1^\star = -\frac{\lambda^\star}{2} \leqslant -1 \rightsquigarrow \lambda^\star \geqslant 2 \qquad \text{(Primal)}$$

$$\lambda^\star \geqslant 0 \qquad \text{(Dual)}$$

so, we should have $\lambda^\star \geqslant 2 \rightsquigarrow$ the dual variable cannot be inactive!

Example: *Minimizing Paraboloid II*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \text{ subject to } w_1 \leqslant -1$$

Slater's conditions hold, so we find dual objective

$$\ell\left(\mathbf{w}, \lambda\right) = w_1^2 + w_2^2 + \lambda\left(w_1 + 1\right)$$

Finally, we check the supplementary slackness

$$\lambda^\star\left(w_1^\star + 1\right) = 0 \rightsquigarrow \lambda^\star\left(1 - \frac{\lambda^\star}{2}\right) = 0$$

Since we look for $\lambda^\star \geqslant 2$, the only solution is

$$\lambda^\star = 2$$

Example: *Minimizing Paraboloid II*

Now, we solve it using the Lagrange multipliers method

$$\min w_1^2 + w_2^2 \ \textit{subject to} \ w_1 \leqslant -1$$

The dual solution is

$$\begin{bmatrix} w_1^\star \\ w_2^\star \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

and $\lambda^\star = 2$ which says that the solution is on the boundary

This makes sense, since the constraint is now actively impacting!

Back to SVC: *Dual Problem*

In SVC, we want to solve the following problem

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 \qquad \textit{subject to } \; v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n \geqslant 1 \textit{ for all } n$$

Maybe we can write it in the standard form as

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 \qquad \textit{subject to } \; 1 - v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n \leqslant 0 \textit{ for all } n$$

We can see that Slater's conditions hold: *we can solve the dual problem*

$$\ell\left(\mathbf{w}, \boldsymbol{\lambda}\right) = \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \lambda_n \left(1 - v_n \mathbf{w}^\mathsf{T} \boldsymbol{x}_n\right)$$

Solving Dual Problem: *Stationary Points*

First, we find the stationary points

$$\nabla \ell \left(\mathbf{w}^\star, \boldsymbol{\lambda}^\star \right) = \mathbf{0} \rightsquigarrow 2\mathbf{w}^\star - \sum_{n=1}^{N} \lambda_n^\star v_n \boldsymbol{x}_n = \mathbf{0}$$

$$\rightsquigarrow \mathbf{w}^\star = \frac{1}{2} \sum_{n=1}^{N} \lambda_n^\star v_n \boldsymbol{x}_n$$

The optimal model is a weighted average of data-points

Solving Dual Problem: *Find Dual Optimal*

We could find the dual objective by replacing $\mathbf{w}^\star$

$$\ell\left(\mathbf{w}^\star, \boldsymbol{\lambda}\right) = \sum_{n=1}^{N} \lambda_n - \frac{1}{4} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m v_n v_m \boldsymbol{x}_n^\mathsf{T} \boldsymbol{x}_m$$

and maximize it, i.e.,

$$\max_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \sum_{n=1}^{N} \lambda_n - \frac{1}{4} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m v_n v_m \boldsymbol{x}_n^\mathsf{T} \boldsymbol{x}_m$$

Or alternatively find feasible solutions to the complementary slackness

Key Observation

Optimal model is characterized only via cross-correlations

Solving Dual Problem: *Complementary Slackness*

We can find $\lambda_n^\star$ also via complementary slackness: at all n we need to see

$$\lambda_n^\star \left(1 - v_n \mathbf{w}^{\star\mathsf{T}} \boldsymbol{x}_n\right) = 0$$

It describes system of equations that is solved uniquely via feasibility constraints

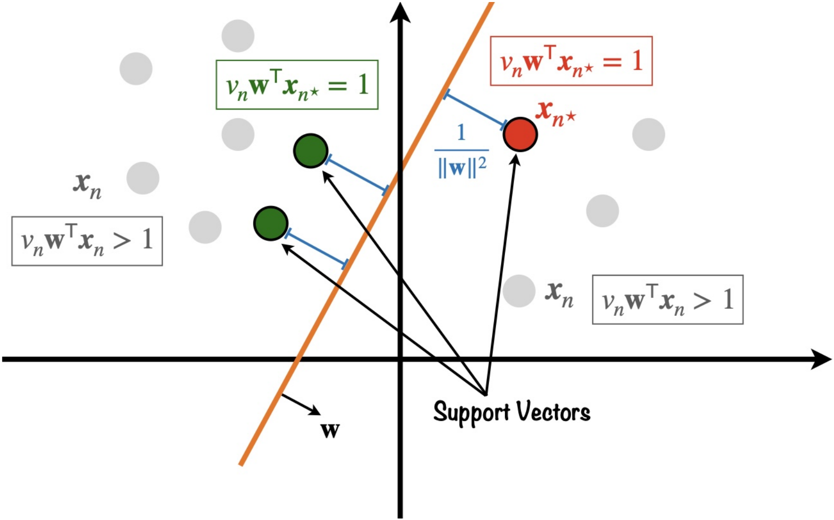
---

But, without solving it, we can say that

- *If the constraint is active, i.e., $v_n \mathbf{w}^{\star\mathsf{T}} \boldsymbol{x}_n = 1$; then, $\lambda_n > 0$*
- *If the constraint is inactive, i.e., $v_n \mathbf{w}^{\star\mathsf{T}} \boldsymbol{x}_n > 1$; then, $\lambda_n = 0$*

? *Which constraints are active in SVC?!*

Active Samples in Solution of SVC



Complementary Slackness: *Only Support Vectors Matter*

We can hence write the solution of SVC as

$$\mathbf{w}^\star = \sum_{n=1}^{N} \underbrace{\lambda_n^\star}_{\neq 0 \text{ at Support Vectors}} v_n \boldsymbol{x}_n$$
$$= \sum_{n \in \mathbb{S}} \lambda_n^\star v_n \boldsymbol{x}_n$$

This is why we call it support vector classifier

Conclusion

To find SVC, we first compute all cross-correlations $\boldsymbol{x}_m^\mathsf{T} \boldsymbol{x}_n$

- *Using cross-correlations we can find $\lambda_n^\star$*
  - ↳ *They are non-zero only of support vectors*
- *We set $\mathbf{w}$ to be the weighted average of support vectors*

SVC: *How We Classify*

Say we found the support vector and their dual variables

We set our model to

$$\mathbf{w} = \sum_{n \in \mathbb{S}} \lambda_n v_n \boldsymbol{x}_n$$

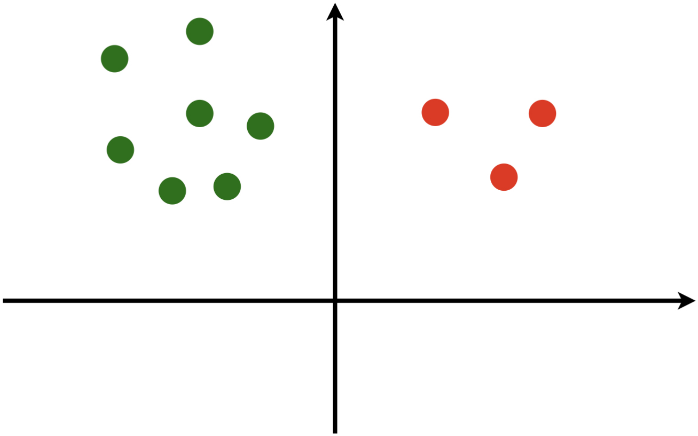
? *How do we classify a new sample $\boldsymbol{x}$?*

We should check the sign of

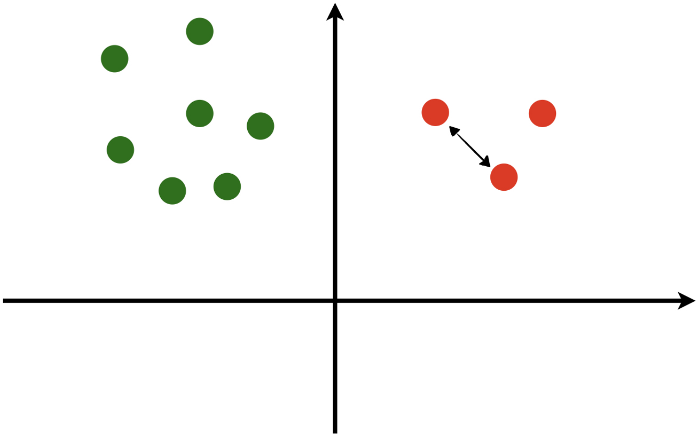
$$y = \mathbf{w}^{\mathsf{T}} \boldsymbol{x} = \sum_{n \in \mathbb{S}} \lambda_n v_n \boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{x}$$

We again need only to know the cross-correlations!

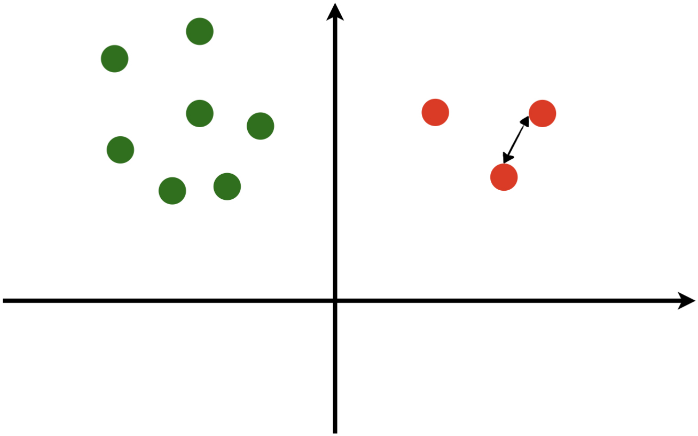
SVC Training: *Visualization*



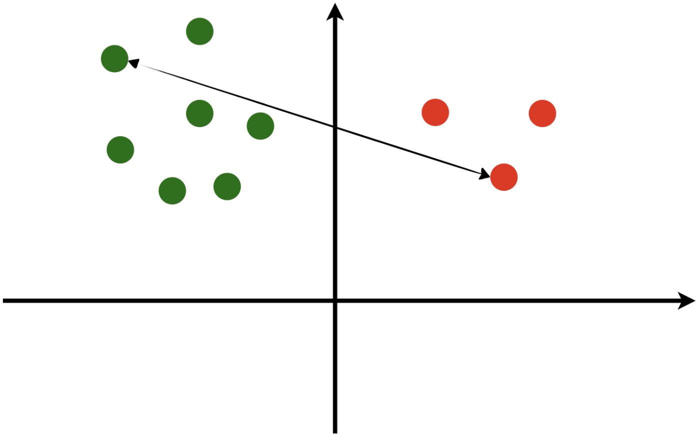
SVC Training: *Visualization*



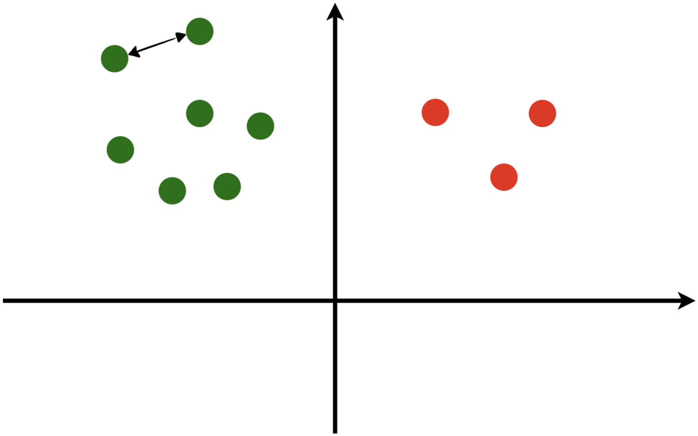
SVC Training: *Visualization*



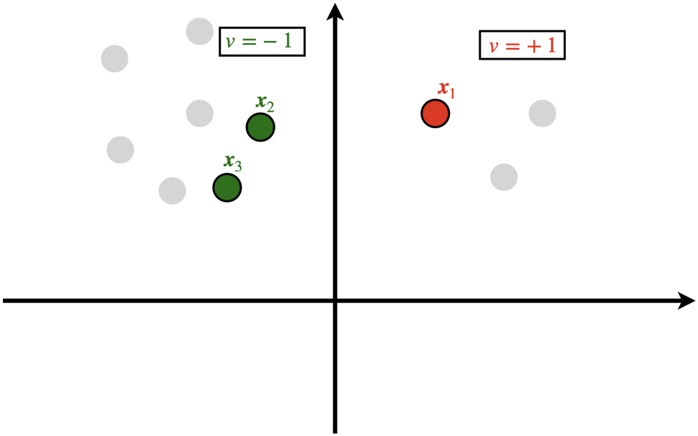
SVC Training: *Visualization*



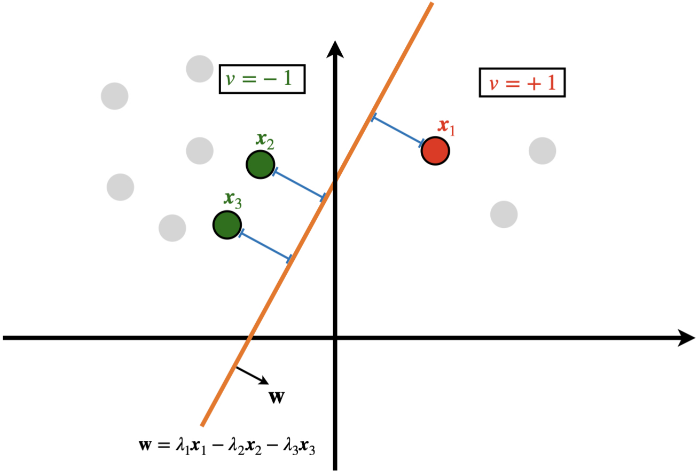
SVC Training: *Visualization*



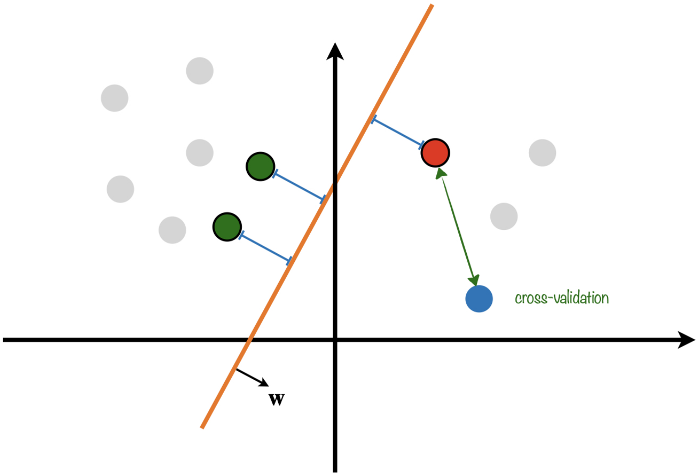
SVC Training: *Visualization*



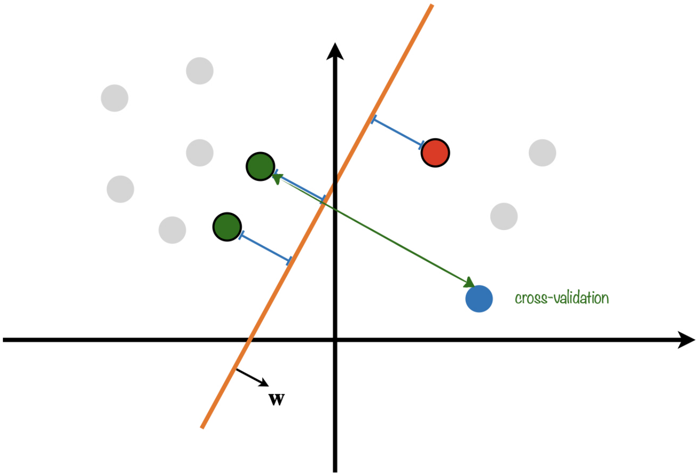
SVC Training: *Visualization*



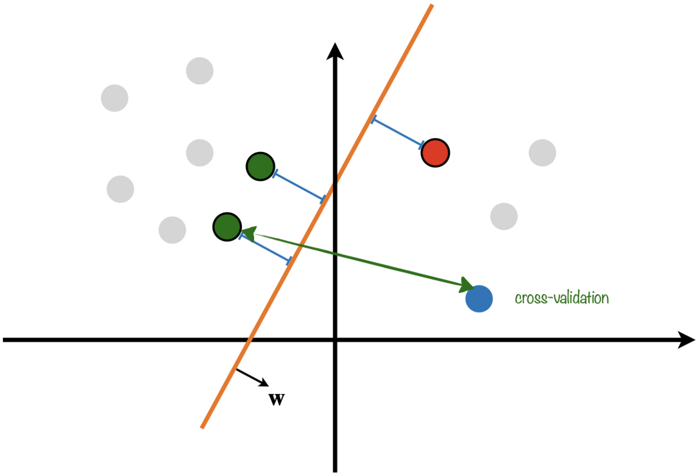
SVC Inference: *Visualization*



SVC Inference: *Visualization*



SVC Inference: *Visualization*

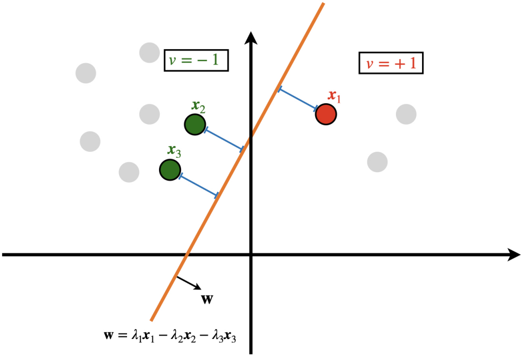


Further Read

- Bishop
  - ↳ Chapter 6: *Section 6.1* ***SVC Solution***
- ESL
  - ↳ Chapter 12: *Section 12.1 – 12.2* ***SVC***

SVC is Still Linear!

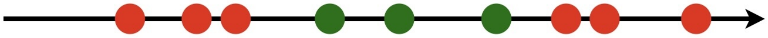
SVC can classify if it is linearly separable



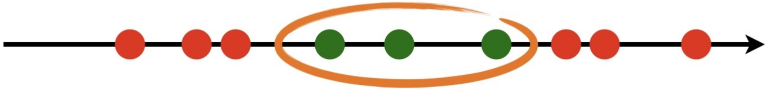
? *What's going to happen if data is not so simply structures?*

Example I: *Cat or Dog*

? *What if in our Cat or Dog example, the weights are sorted like this?*



! *Well, we need a nonlinear classifier then*



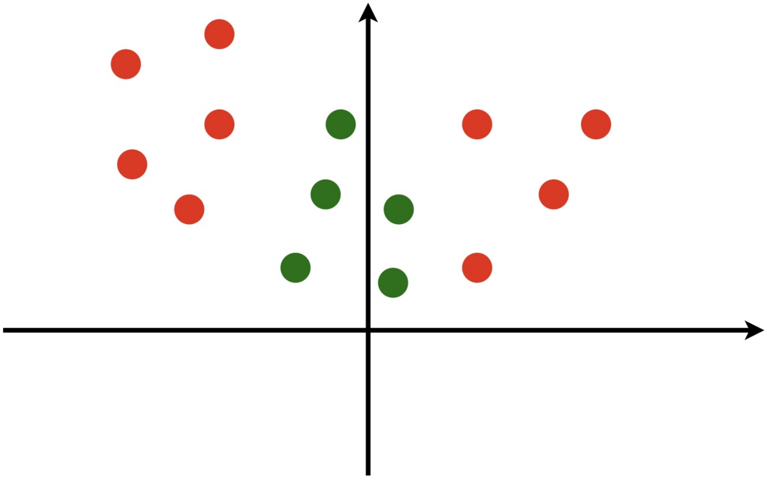
Nonlinear Classifier

Classifier that infers class of samples from nonlinear computations, e.g.,

$$z_n = w_1 x_n + w_2 x_n^2 \rightsquigarrow y_n = \mathrm{sign}\left(z_n\right)$$

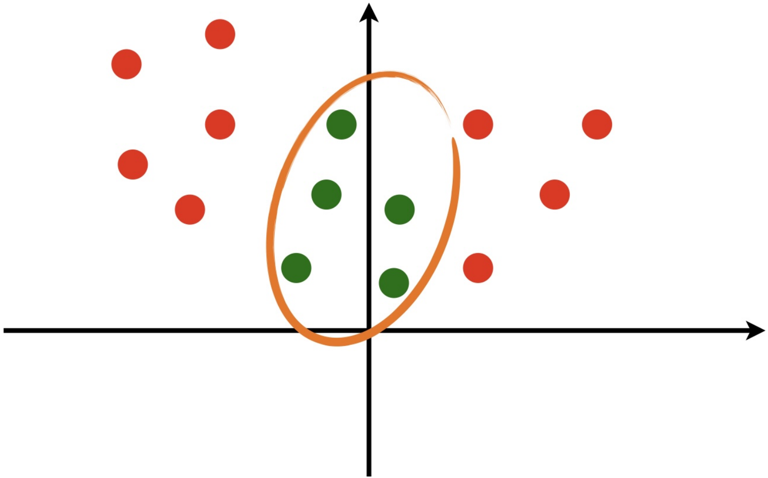
Example II: *Curved Boundary*

! *Such classifier should be needed in many problems*



Example II: *Curved Boundary*

! *Such classifier should be needed in many problems*

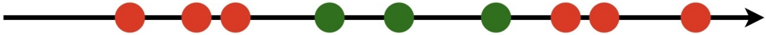


Example I: *Cat or Dog*

Let's get back to our Cat or Dog example: *We have a dataset*

$$\mathbb{D} = \{(x_n, v_n) : n = 1, \dots, N\}$$

But now we cannot simply divide the green points from red ones by thresholding



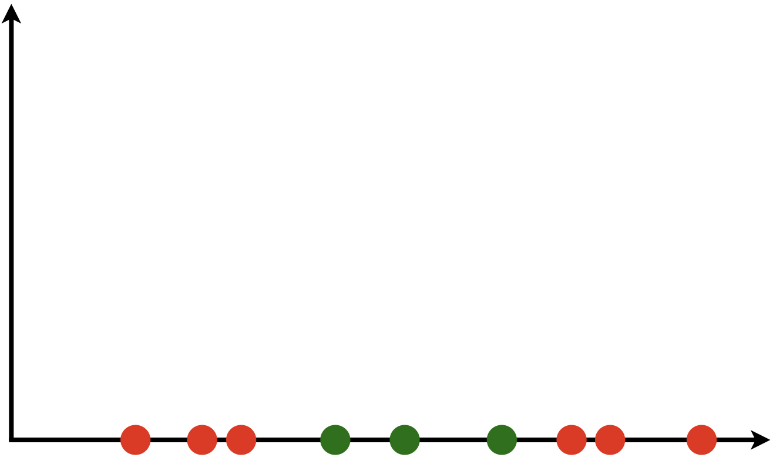
? *How can we do it in a systematic way then?*

Example I: *Cat or Dog*

Let's try a trick

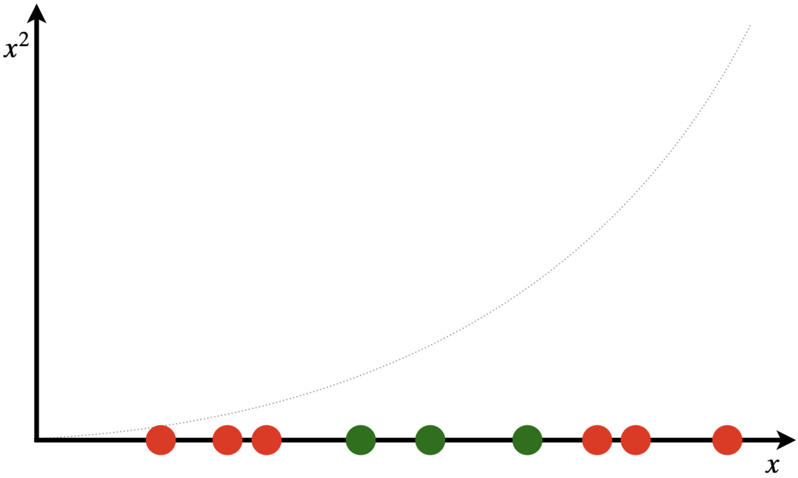
Example I: *Cat or Dog*

Let's try a trick: we add a second dimension to our data



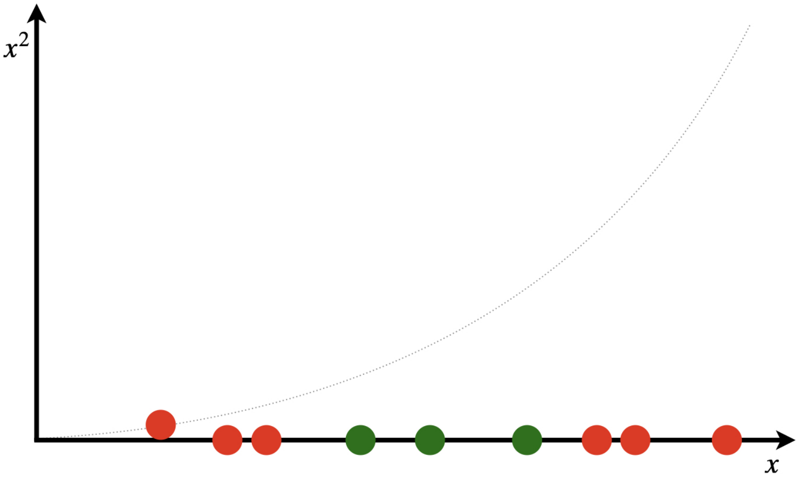
Example I: *Cat or Dog*

Let's try a trick: for this dimension we compute the square of samples



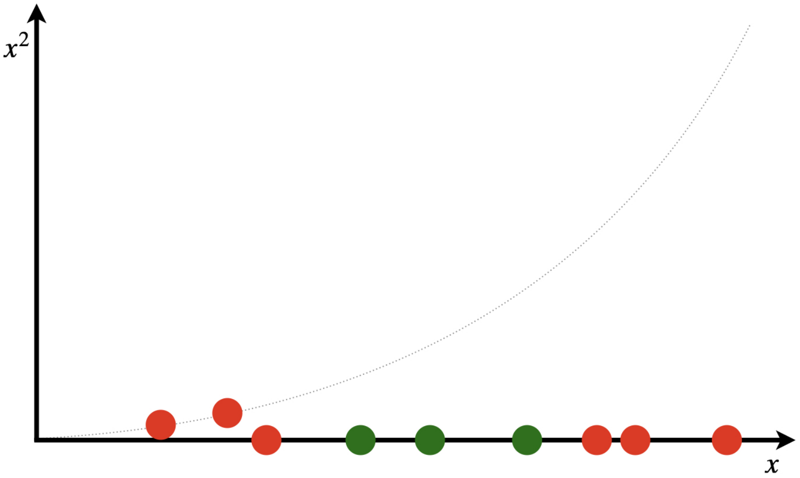
Example I: *Cat or Dog*

Let's try a trick: now we represent each point with a new vector $\tilde{\boldsymbol{x}} = [x, x^2]$



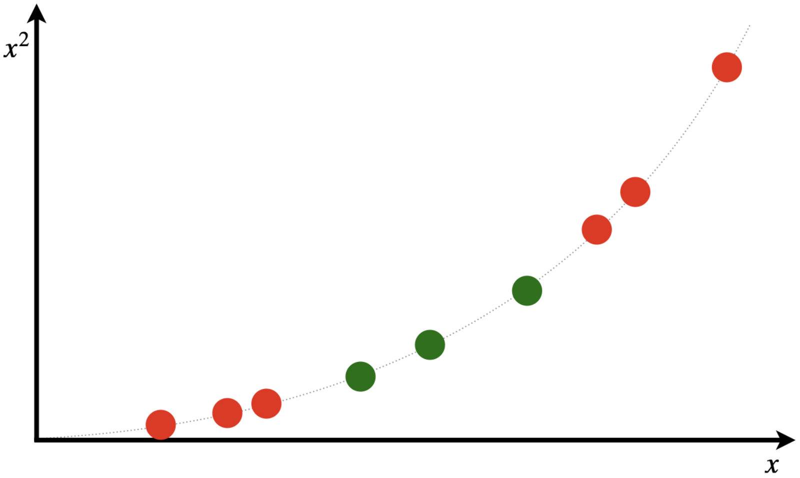
Example I: *Cat or Dog*

Let's try a trick: now we represent each point with a new vector $\tilde{\boldsymbol{x}} = [x, x^2]$



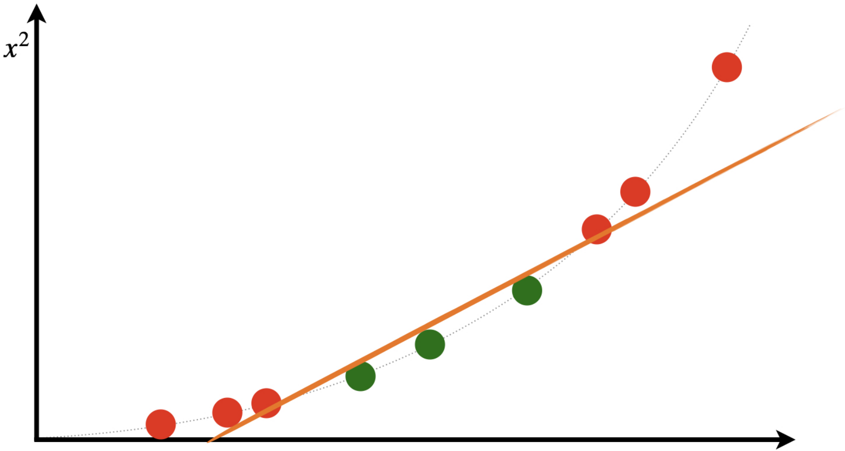
Example I: *Cat or Dog*

Let's try a trick: we transform the whole dataset



Example I: *Cat or Dog*

This transformed dataset can be linearly classified in 2D space!



Going Higher Dimensions

? *What is happening here?*

We see that the dataset cannot be classified perfectly by a linear model

We extract high-dimensional features as

$$x \mapsto \tilde{\boldsymbol{x}} = \varphi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

and see that

$$\mathbb{D} = \{(\tilde{\boldsymbol{x}}_n, v_n) : n = 1, \ldots, N\}$$

is classified by linear model!

We could make a nonlinear problem linear in higher dimensions

Nonlinear Low-Dimension $\leftrightsquigarrow$ Linear High-Dimension

This suggests a generic recipe: *for a nonlinear problem with data*

$$\mathbb{D} = \{(\boldsymbol{x}_n, v_n) : n = 1, \ldots, N\}$$

first extract high-dimensional features

Feature Extraction: *Embedding*

Use an embedding function φ to map samples to high-dimensional features

$$\boldsymbol{x}_n \in \mathbb{R}^d \mapsto \varphi(\boldsymbol{x}_n) \in \mathbb{R}^D$$

where $D > d$

Nonlinear Low-Dimension $\leftrightsquigarrow$ Linear High-Dimension

This suggests a generic recipe: *for a nonlinear problem with data*

$$\mathbb{D} = \{(\boldsymbol{x}_n, v_n) : n = 1, \ldots, N\}$$

then find a linear classifier for high-dimensional features

Feature Classification

Find a SVC with weight $\mathbf{w} \in \mathbb{R}^D$ *for* $\varphi(\boldsymbol{x}_n)$

$$y_n = \operatorname{sign}\left(\mathbf{w}^{\mathsf{T}} \varphi(\boldsymbol{x}_n)\right)$$

Nonlinear Low-Dimension $\leftrightsquigarrow$ Linear High-Dimension

? *How to use the model for inferring class of a new data?*

We infer based on the high-dimensional features

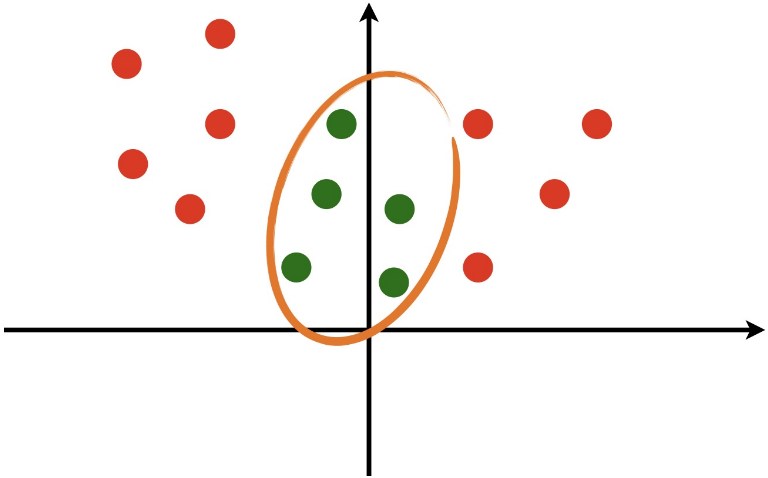
Inference

The label of the new sample $\boldsymbol{x}$ is given as

$$y = \operatorname{sign}\left(\mathbf{w}^{\mathsf{T}} \varphi\left(\boldsymbol{x}\right)\right)$$

Example II: *Curved Boundary*

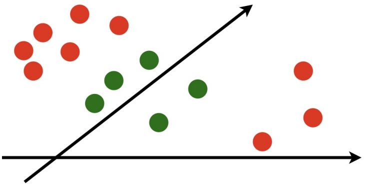
Let's look at the round boundary visual example



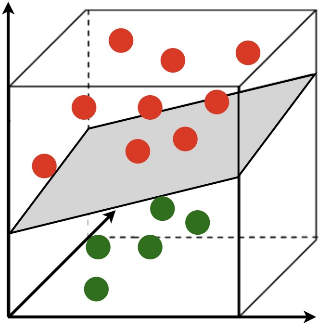
Example II: *Curved Boundary*

The points can be linearly separable in 3D

get the points $\boldsymbol{x}_n$ in 2D



maps them to 3D as $\varphi(\boldsymbol{x}_n)$



Support Vector Machine: *Nonlinear SVC*

Support Vector Machine

Support vector machine (SVM) is an SVC that learns from high-dimensional features extract by an embedding function $\varphi : \mathbb{R}^d \mapsto \mathbb{R}^D$

↳ *It can learn nonlinear patterns*

Though sounds promising, it seems a bit challenging

- ? *What should be the embedding function?*
- ? *What is the model need super high-dimensional features?*
  - ↳ *This means that we should work in very high dimensions*
  - ↳ *We even don't know how large it should be!*

Recall: *SVC Training and Inference*

To train an SVC, we solve the dual problem

$$\max_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \sum_{n=1}^{N} \lambda_n - \frac{1}{4} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m v_n v_m \boldsymbol{x}_n^\mathsf{T} \boldsymbol{x}_m$$

Once we found the dual values $\lambda_n^\star$, we classify as

$$\mathbf{w}^\star = \sum_{n=1}^{N} \lambda_n^\star v_n \boldsymbol{x}_n \rightsquigarrow y = \operatorname{sign}\left(\sum_{n=1}^{N} \lambda_n^\star v_n \boldsymbol{x}_n^\mathsf{T} \boldsymbol{x} \right)$$

Recall: Only Cross-Validations

Recall that we only need the correlations $\boldsymbol{x}_m^\mathsf{T} \boldsymbol{x}_n$

SVC with Embedded Features

For SVM, we would do the same: *we find the dual values through*

$$\max_{\boldsymbol{\lambda}\geqslant\mathbf{0}} \sum_{n=1}^{N} \lambda_n - \frac{1}{4}\sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n\lambda_m v_n v_m \varphi\left(\boldsymbol{x}_n\right)^{\mathsf{T}} \varphi\left(\boldsymbol{x}_m\right)$$

We then infer the class of a new sample as

$$\mathbf{w}^\star = \sum_{n=1}^{N} \lambda_n^\star v_n \boldsymbol{x}_n \rightsquigarrow y = \operatorname{sign}\left(\sum_{n=1}^{N} \lambda_n^\star v_n \varphi\left(\boldsymbol{x}_n\right)^{\mathsf{T}} \varphi\left(\boldsymbol{x}\right)\right)$$

Again: Only Cross-Validations

Here again we only need the correlations $\varphi\left(\boldsymbol{x}_m\right)^{\mathsf{T}} \varphi\left(\boldsymbol{x}_n\right)$

Kernel Trick

We don't really need to work in high-dimensional space:

It's enough to know how the embeddings are correlated!

Kernel

A function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ that computes the cross-correlation between high-dimensional features of two samples

$$\mathcal{K}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) = \varphi\left(\boldsymbol{x}_n\right)^{\mathsf{T}} \varphi\left(\boldsymbol{x}_m\right)$$

We don't really need the embedding function φ:

we only need the kernel!

Kernel Trick: *Extended Cross-Validation*

This is called the kernel trick: *choose a kernel $\mathcal{K}$ and solve*

$$\max_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \sum_{n=1}^{N} \lambda_n - \frac{1}{4} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m v_n v_m \mathcal{K}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$$

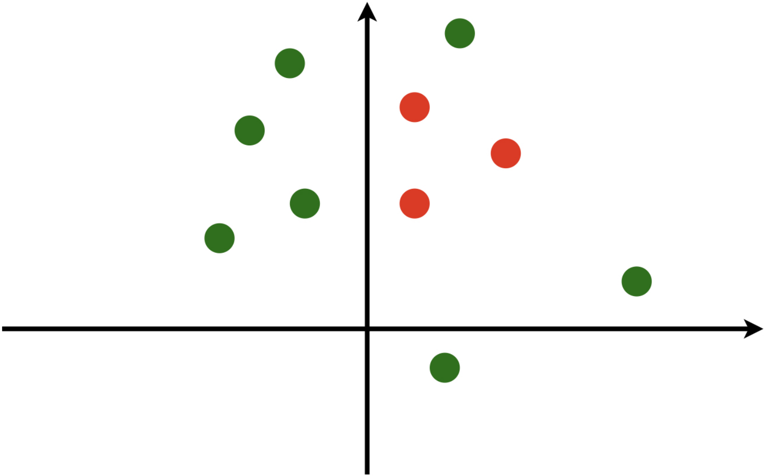
Once $\lambda_n^\star$ found classify by the same kernel $\mathcal{K}$ as

$$\mathbf{w}^\star = \sum_{n=1}^{N} \lambda_n^\star v_n \boldsymbol{x}_n \rightsquigarrow y = \operatorname{sign}\left(\sum_{n=1}^{N} \lambda_n^\star v_n \mathcal{K}\left(\boldsymbol{x}_n, \boldsymbol{x}\right)\right)$$

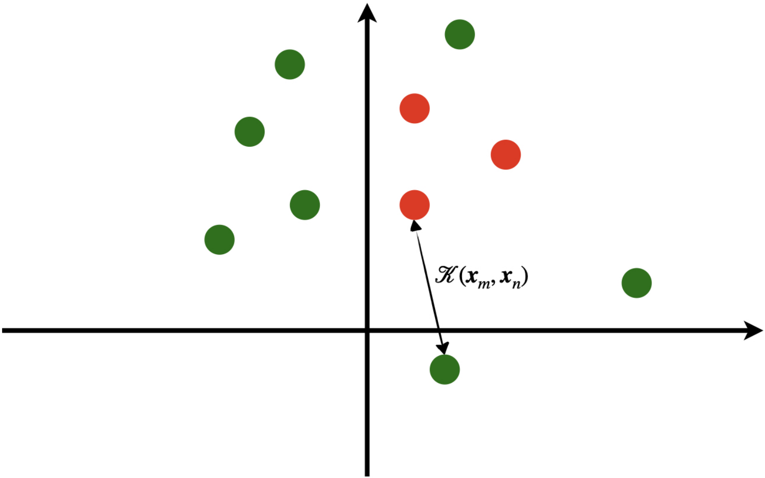
Moral of Story

SVM does what SVC do using a nonlinear (potentially very complicated) kernel

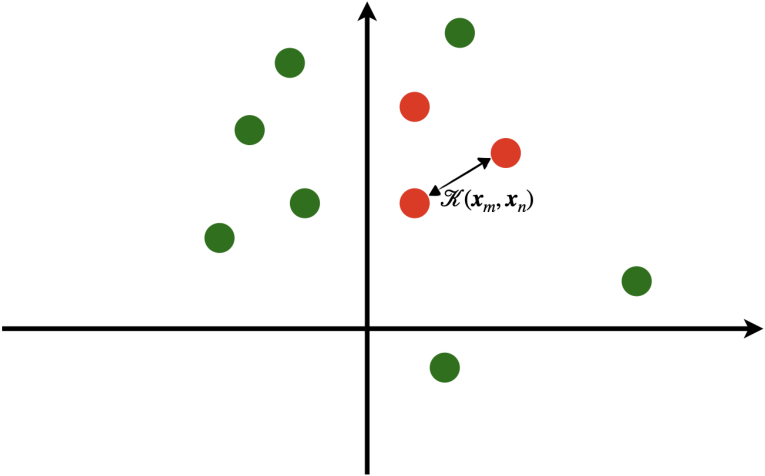
Support Vector Machines: *Visualization*



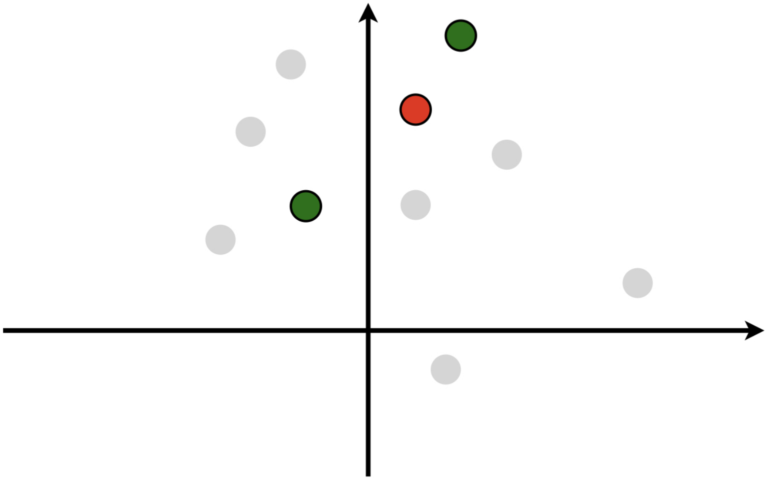
Support Vector Machines: *Visualization*



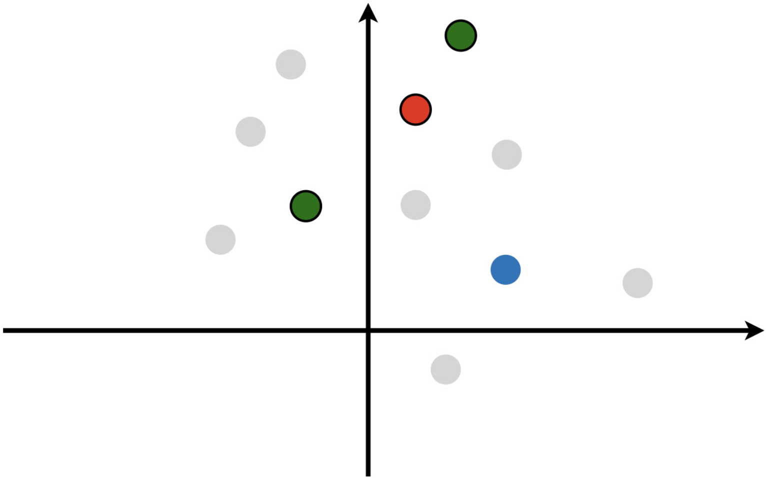
Support Vector Machines: *Visualization*



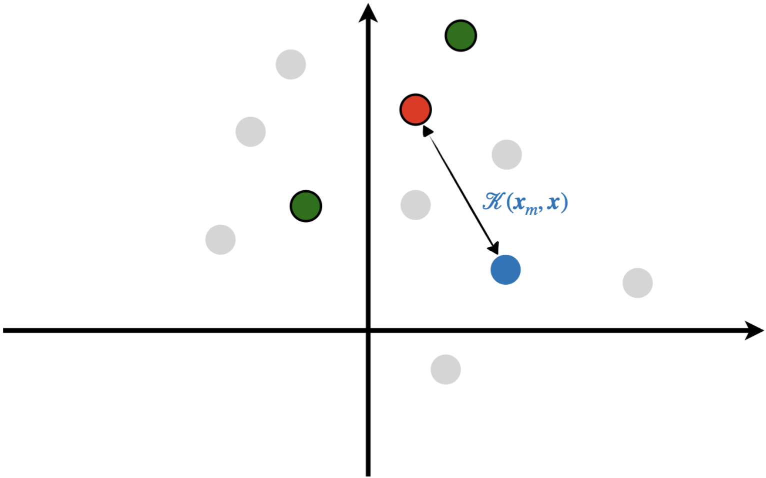
Support Vector Machines: *Visualization*



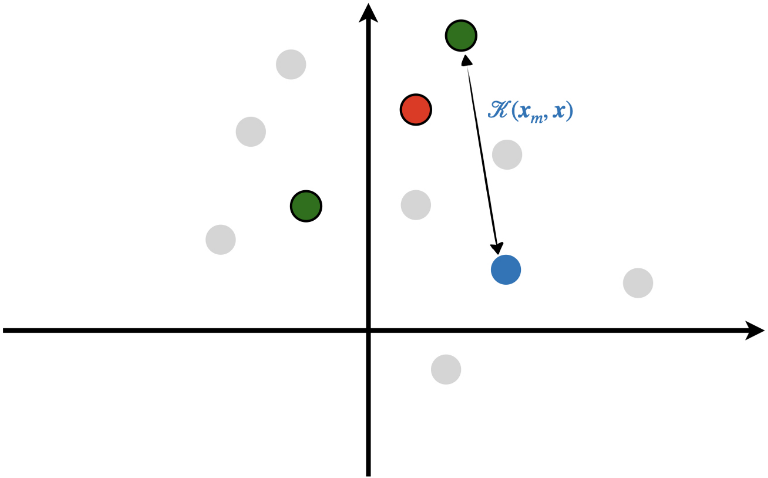
Support Vector Machines: *Visualization*



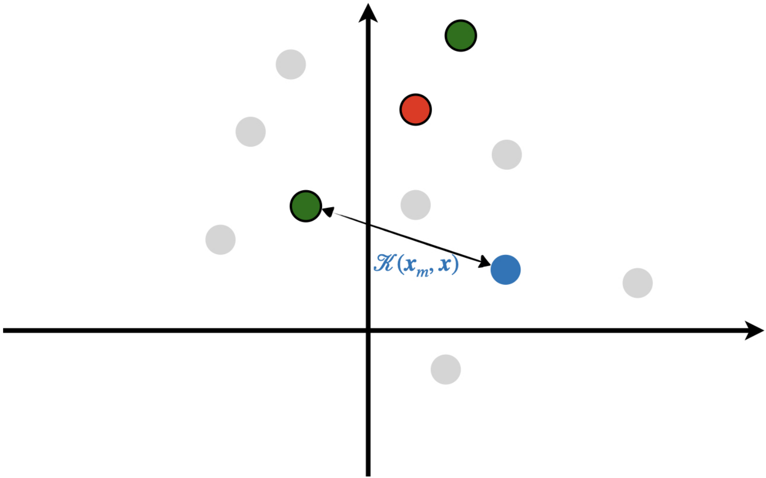
Support Vector Machines: *Visualization*



Support Vector Machines: *Visualization*



Support Vector Machines: *Visualization*



Famous Kernels: *Polynomial*

- **?** *What should we choose as the kernel?*
- **!** *There are some known choices*

Polynomial Kernel

The polynomial kernel of order p *is defined as*

$$\mathcal{K}\left(\boldsymbol{x}_n, \boldsymbol{x}_m\right) = \left(\boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{x}_m + c\right)^p$$

Corresponding embedding computes polynomial features

Famous Kernels: *Polynomial – Example*

Say order is 2, the samples are scalars and set $c = 1$

$$\begin{aligned}\mathcal{K}(x_n, x_m) &= (x_n x_m + 1)^2 \\ &= x_n^2 x_m^2 + 2x_n x_m + 1 \\ &= \begin{bmatrix} x_m^2 & \sqrt{2}x_m & 1 \end{bmatrix} \begin{bmatrix} x_n^2 \\ \sqrt{2}x_n \\ 1 \end{bmatrix}\end{aligned}$$

This is like our Cat or Dog example

$$\varphi(x) = \begin{bmatrix} x^2 \\ \sqrt{2}x \\ 1 \end{bmatrix}$$

Famous Kernels: *Gaussian Kernel*

Gaussian (Radial) Kernel

The polynomial kernel of order p is defined as

$$\mathcal{K}\left(\boldsymbol{x}_n, \boldsymbol{x}_m\right) = \exp\left\{-\frac{\|\boldsymbol{x}_n - \boldsymbol{x}_m{}^2\|}{\sigma}\right\}$$

The Gaussian kernel corresponds to

embedding in infinite-dimensional space!

Try to use Taylor series expansion of exponential function to see this

$$\exp\left\{x\right\} = 1 + x + \frac{x^2}{2!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Preset Kernel $\equiv$ *Feature Engineering*

- **?** *How do we know if we have chosen a good kernel?*
- **!** *Nobody knows!*

Feature Engineering

By choosing a predefined kernel, we are implicitly setting the embedding functions. This is often called feature engineering, since we indirectly use a fixed rule for extracting features

- **?** *But is there any other way?*
- **!** *Maybe, we can "let data speaks by itself"!*

Representation Learning: *Learning Kernels*

We can instead set the kernel to be a parameterized function

$$\mathcal{K}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$$

If we use the SVM, our final loss depends on both $\mathbf{w}$ and $\boldsymbol{\theta}$: *we can train both*

$$\min_{\boldsymbol{\theta}, \mathbf{w}} \hat{R}\left(\boldsymbol{\theta}, \mathbf{w}\right)$$

Representation Learning

We learn both the kernel and classifier together: this is like learning how to represent data first and then classify it

Example: *We can leave σ in Gaussian kernel undecided and find it jointly with $\mathbf{w}$*

Observation: *Neural Networks Give Excellent Representation*

? *How can we find the right form for the kernel?*

There is a rather rich literature on it

This is why it has its own name: Representation Learning

But it later turned out that

By repeating the linear model over and over we can build excellent kernels

This resembles what we know as *neural networks*:

deep neural networks can make us great kernels!

NNs are hence what we are going to study next!

Further Read

- Bishop
  - ↳ Chapter 6: *Sections 6.2 – 6.4* ***SVM***
- ESL
  - ↳ Chapter 12: *Section 12.3 – 12.4* ***SVM***