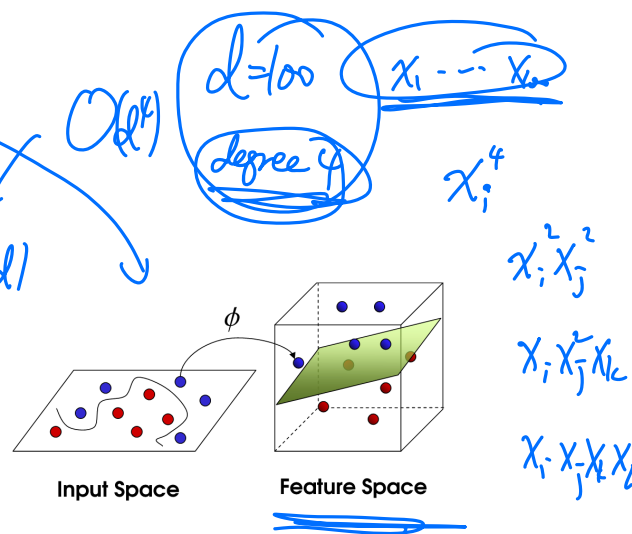
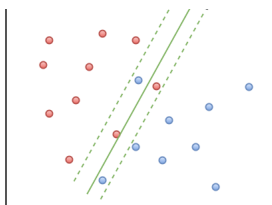
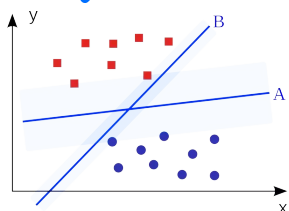


Section 13. Support vector machines and kernel methods

- Support Vector Machines
- Lagrange multiplier (KKT theorem)
- Regularization
- Kernel Methods



➤ Support Vector Machines (SVM)

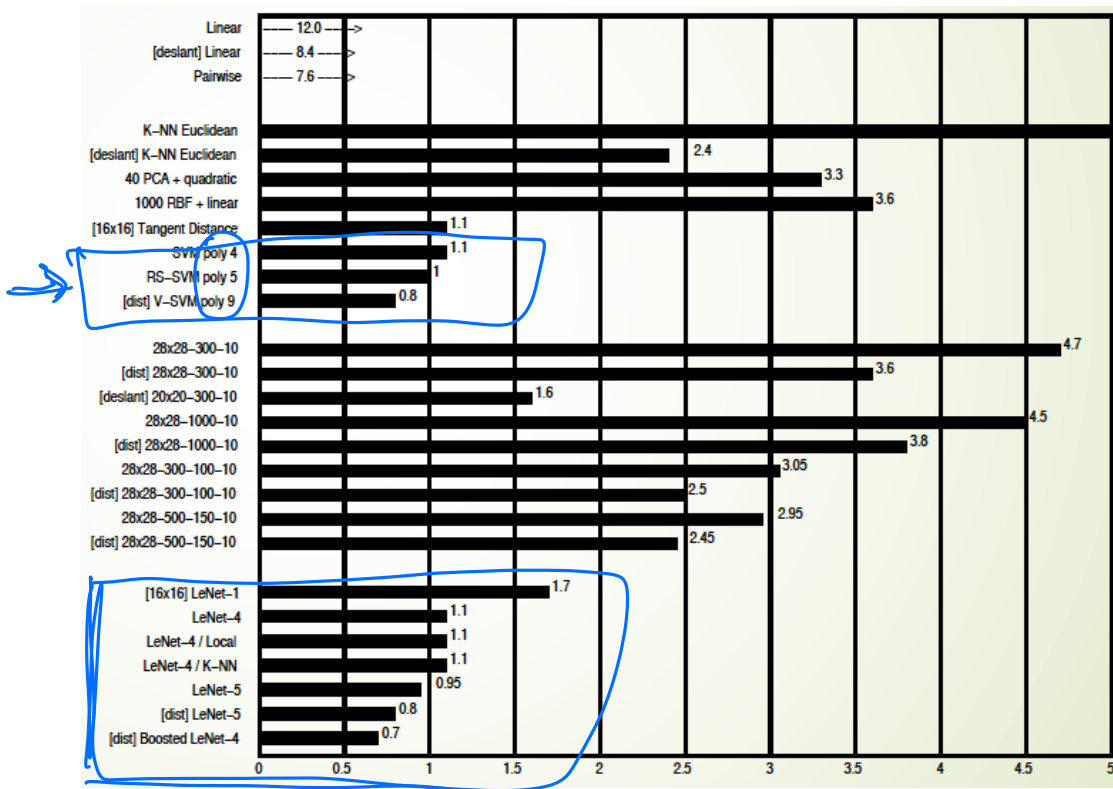
SVM was Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues in 1994. ^{1963 PhD}

- Support vector machine is one of the most popular machine learning methodologies.
- Empirically successful, with well developed theory.
- One of the best off-the-shelf methods.
- We mainly address classification.

SVM v.s. CNN:

Simple SVM performs as well as Multilayer Convolutional Neural Networks which need careful tuning (LeNets) (Second dark era for NN: 2000s)

MNIST Dataset Test Error: SVM vs. CNN



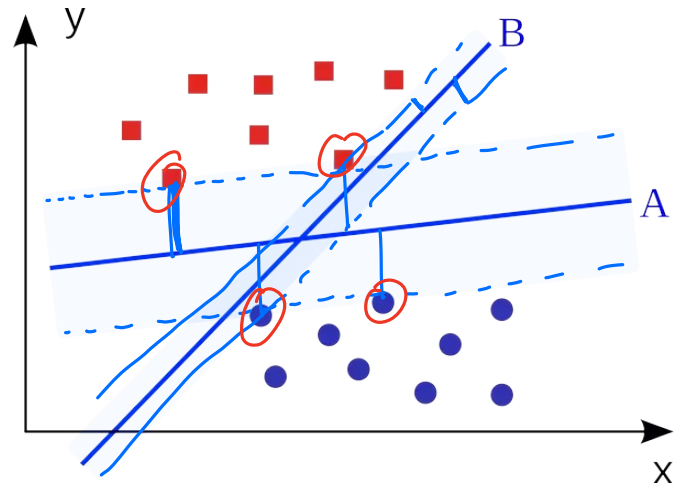
LeCun et al. 1998

➤ **Support Vector Machines (SVM)** for binary classification. (Max-Margin Classifier)

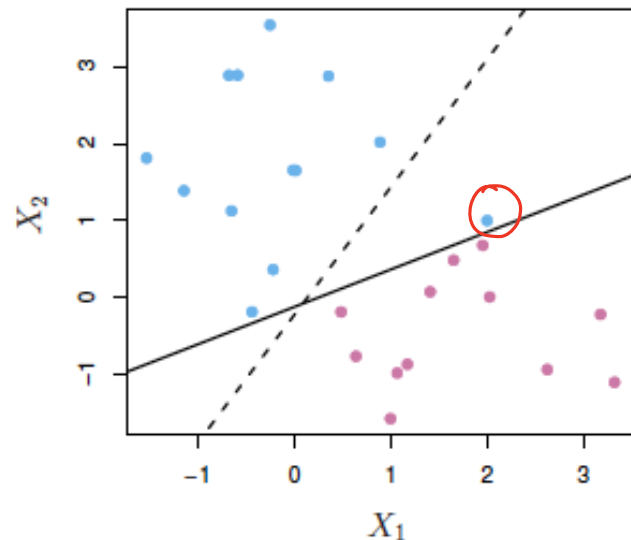
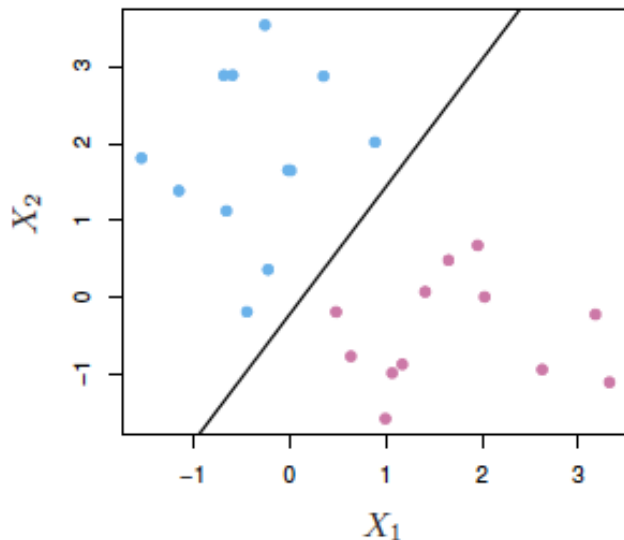
Assume the datasets are *linearly separable*.

Maximal margin hyperplane:

- The **optimal** separating hyperplane that is **farthest** from the training observations.
- The separating hyperplane such that the **minimum** distance of any training point to the hyperplane is the **largest**.
- Creates the **widest gap** between the two classes.
- Points on the boundary hyperplane, those with smallest distance to the max margin hyperplane, are called **support vectors**. They support the maximal margin hyperplane in the sense vector that if these points were moved slightly then the maximal margin hyperplane would move as well.



- Note that **margin** $M > 0$ is the **half** of the width of the strip separating the two classes.
- The eventual solution, the **max margin hyperplane** is determined by the **support vectors**.
- If $x^{(i)}$ on the correct side of the trip varies, the solution would remain same.
- The max margin hyperplane may vary a lot when the support vectors vary.



➤ SVM setup:

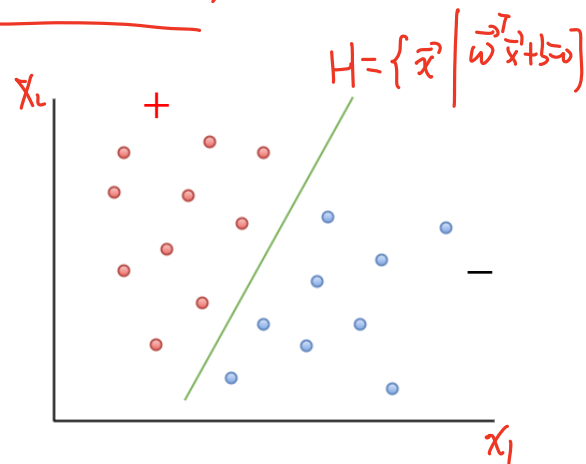
Binary Classification Data: $D = \{(\vec{x}^{(i)}, y^{(i)}), i = 1, \dots, n\}$ $y^{(i)} \in \{-1, 1\}$,

Assume the datasets are linearly separable. (more features than data)

Goal: Find a **linear classifier**:

$$h(\vec{x}) = \text{sign}(\vec{\theta}^T \vec{x}) = \text{sign}(\vec{w} \cdot \vec{x} + b)$$

$$h(\vec{x}) = \begin{cases} 1, & \text{if } \vec{w} \cdot \vec{x} + b \geq 0 \\ -1, & \text{if } \vec{w} \cdot \vec{x} + b < 0 \end{cases}$$



Notations: $\vec{\theta} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$ $\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$ or $\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$

Decision Boundary: Hyperplane H

$$H = \{ \vec{x} \mid \vec{\theta}^T \vec{x} = 0 \}$$

or:

$$w_1 x_1 + \cdots w_d x_d + b = 0$$

or:

$$\vec{w}^T \vec{x} + b = 0$$

Property:

\vec{w} is orthogonal to the hyperplane H .

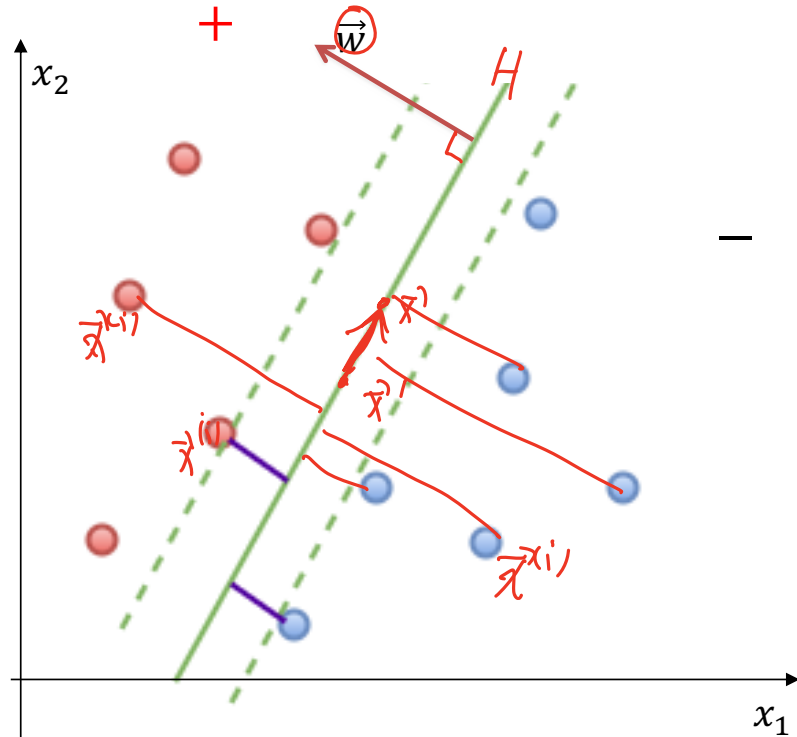
Reason:

Any two points \vec{x} and \vec{x}' on hyperplane,

$$\vec{w}^T \vec{x} + b = 0$$

$$\vec{w}^T \vec{x}' + b = 0$$

$$\text{So, } \vec{w} \cdot (\vec{x} - \vec{x}') = 0.$$



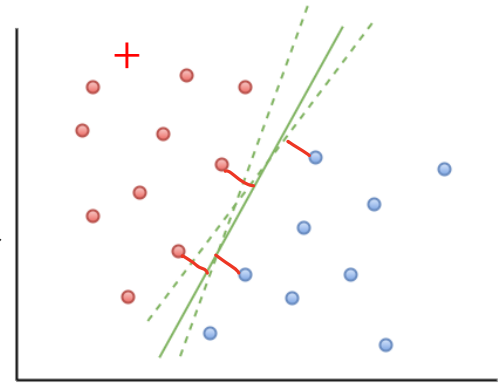
$$\text{Margin} = \min_{1 \leq i \leq n} \text{dist}(\vec{x}^{(i)}, H)$$

$$H = \{ \vec{x} \mid \vec{w}^T \vec{x} + b = 0 \}$$

➤ Maximal margin hyperplane (Hard-margin SVM classifier)

Initial Goal: Find hyperplane parameters \vec{w} and b such that for all $1 \leq i \leq n$

$$y^{(i)} = \begin{cases} 1, & \text{if } \vec{w} \cdot \vec{x}^{(i)} + b \geq 0 \\ -1, & \text{if } \vec{w} \cdot \vec{x}^{(i)} + b < 0 \end{cases}$$



Equivalently, find \vec{w} and b such that for all $1 \leq i \leq n$

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) > 0 \quad \text{(I)}$$

There are many different hyperplanes $H = \{\vec{x} \mid \vec{w}^T \vec{x} + b = 0\}$.

Question: What is the best separating hyperplane?

Updated SVM Goal: Find hyperplane with **largest margin**

Find \vec{w} and b such that (I) and

$$\max_{\vec{w}, b} \text{Margin} = \max_{\vec{w}, b} \min_{1 \leq i \leq n} \text{dist}(\vec{x}^{(i)}, H) \quad \text{(II)}$$

Property: The **distance** $\gamma^{(i)}$ between $\vec{x}^{(i)}$ and H is

$$\gamma^{(i)} = \text{dist}(\vec{x}^{(i)}, H)$$

$$= y^{(i)} \frac{1}{\|\vec{w}\|} (\vec{w} \cdot \vec{x}^{(i)} + b)$$

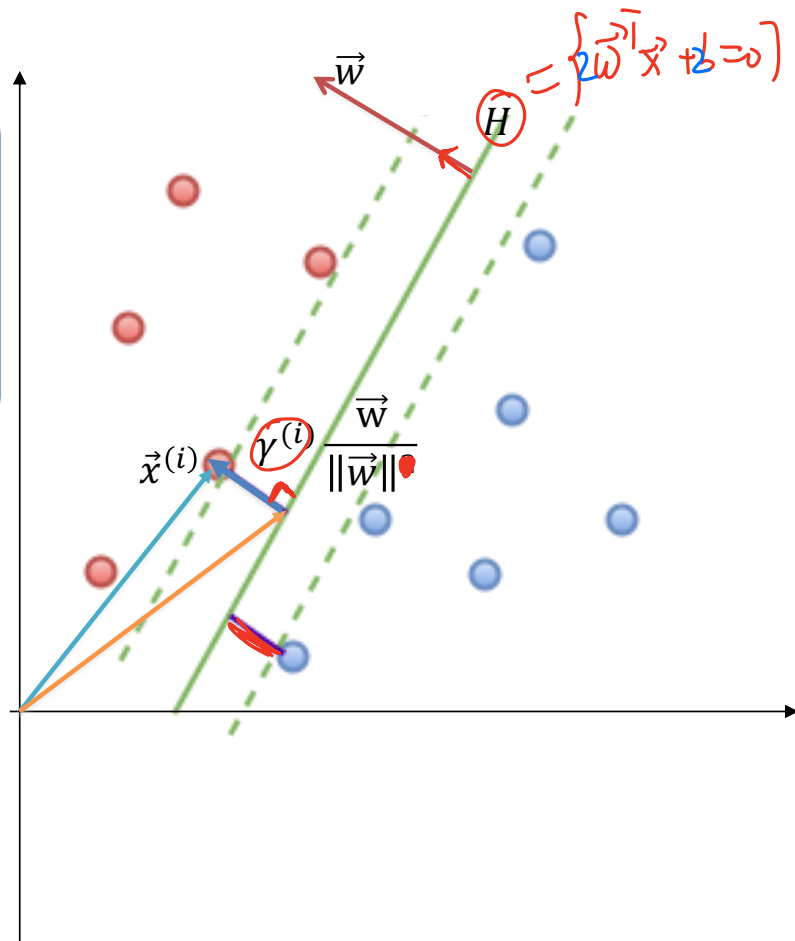
Proof: For positive point $(\vec{x}^{(i)}, y^{(i)} = 1)$,

$$\vec{w}^T \left(\vec{x}^{(i)} - \gamma^{(i)} \frac{\vec{w}}{\|\vec{w}\|} \right) + b = 0$$

Solve $\gamma^{(i)}$ we have

$$\gamma^{(i)} = \frac{1}{\|\vec{w}\|} (\vec{w} \cdot \vec{x}^{(i)} + b)$$

Similarly for negative label points, we have $\gamma^{(i)} = -\frac{1}{\|\vec{w}\|} (\vec{w} \cdot \vec{x}^{(i)} + b)$.



SVM Goal: Find \vec{w} and b such that $y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) > 0$ for all $1 \leq i \leq n$

and $\max_{\vec{w}, b} \left[\min_{1 \leq i \leq n} \text{dist}(\vec{x}^{(i)}, H) \right]$ $\text{Geometrical} = \text{margin}$

Equivalently, find \vec{w} and b

$$\max_{\vec{w}, b} \min_{1 \leq i \leq n} y^{(i)} \frac{1}{\|\vec{w}\|} (\vec{w} \cdot \vec{x}^{(i)} + b) \text{ such that } y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) > 0$$

Equivalently, find \vec{w} and b

$$\max_{\vec{w}, b} \left(\frac{1}{\|\vec{w}\|} \right) \min_{1 \leq i \leq n} y^{(i)} (\vec{w} \cdot \vec{x}^{(i)} + b) \text{ such that } y^{(i)} (\vec{w} \cdot \vec{x}^{(i)} + b) > 0$$

Denote $\lambda := \min_{1 \leq i \leq n} y^{(i)} (\vec{w} \cdot \vec{x}^{(i)} + b)$ $= \text{algebraic margin}$. Equivalently, find \vec{w} and b

$$\max_{\vec{w}, b} \frac{1}{\|\vec{w}\|} \lambda \text{ such that } y^{(i)} (\vec{w} \cdot \vec{x}^{(i)} + b) \geq \lambda \quad \text{for all } i$$

For the **same** hyperplane $H = \{\vec{x} \mid \vec{w}^T \vec{x} + b = 0\}$, we can scale \vec{w} and b anyway we want. So, we choose a 'smart' scale such that $\lambda = 1$, i.e., **margin** = $\frac{1}{\|\vec{w}\|}$

Equivalently, find \vec{w} and b

$$\max_{\vec{w}, b} \frac{1}{\|\vec{w}\|} \text{ such that } y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1$$

$$y^{(i)} \in \{-1, 1\}$$

Equivalently, find \vec{w} and b

$$\max_{\vec{w}, b} \frac{1}{\|\vec{w}\|^2} \text{ such that } 1 - y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 0$$

Equivalently, find \vec{w} and b

$$\min_{\vec{w}, b} \underbrace{\vec{w}^T \vec{w}}_{= w_1^2 + w_2^2 + \dots + w_d^2} \text{ such that } 1 - y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \leq 0$$

$$i = 1, \dots, n$$

The objective is a *quadratic term*, and the constraints are all *linear*, which is called a quadratic optimization problem. https://en.wikipedia.org/wiki/Quadratic_programming
It has a unique solution whenever a separating hyper plane exists

$$h(\vec{x}) = \vec{w}^T \vec{x} + b$$

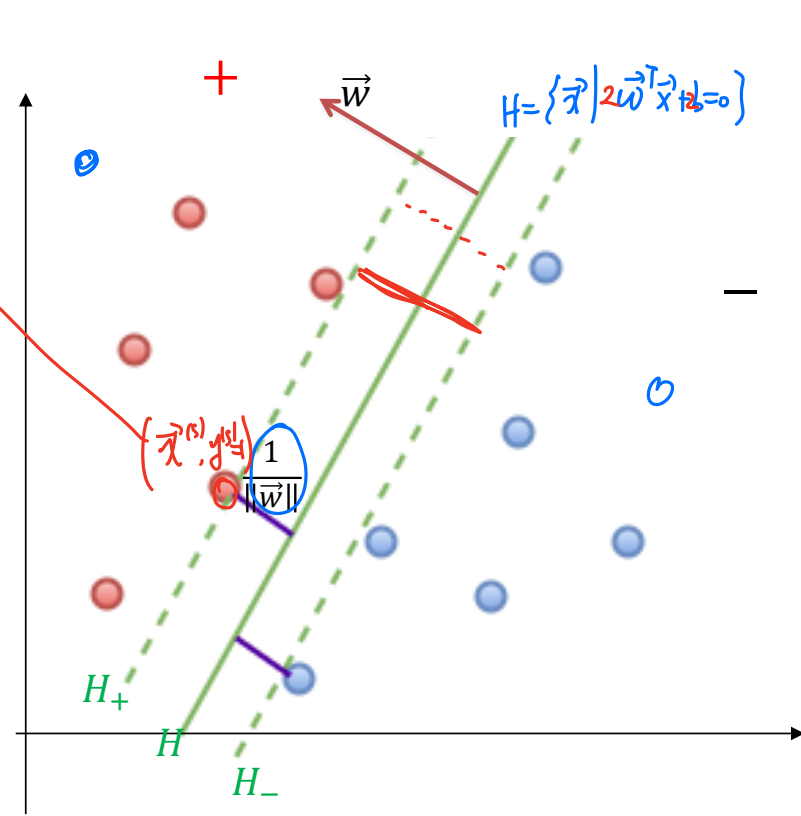
Remarks:

- The constraints $y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1$ for all i are equivalent to **margin** = $\frac{1}{\|\vec{w}\|}$
- The max-margin separating hyperplane, and two margin hyperplanes are:

$$H = \{\vec{x} \mid \vec{w} \cdot \vec{x} + b = 0\}$$

$$H_+ = \{\vec{x} \mid \vec{w} \cdot \vec{x} + b = 1\}$$

$$H_- = \{\vec{x} \mid \vec{w} \cdot \vec{x} + b = -1\}$$



- For optimal \vec{w}, b , the support vectors $(\vec{x}^{(i)}, y^{(i)})$ satisfy

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) = 1$$

➤ Optimization with Constraints.

Ridge / Lasso

Example: Consider the optimization problem

$$\text{Maximize (Minimize) } f(x, y) \text{ subject to } g(x, y) = c$$

Following J. Lagrange (1736–1813),

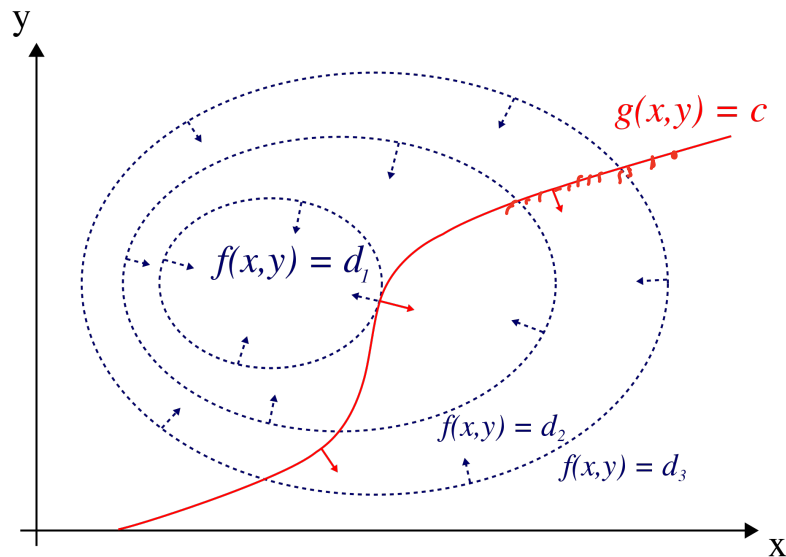
we can define Lagrangian

$$L(x, y, \lambda) := f(x, y) - \lambda(g(x, y) - c)$$

and calculate gradients

$$\nabla_{x,y,\lambda} L = 0$$

$$\begin{cases} \frac{\partial L}{\partial x} = 0 \\ \frac{\partial L}{\partial y} = 0 \\ g(x, y) = c \end{cases}$$



➤ Optimization with Constraints.

Optimization Question (★):

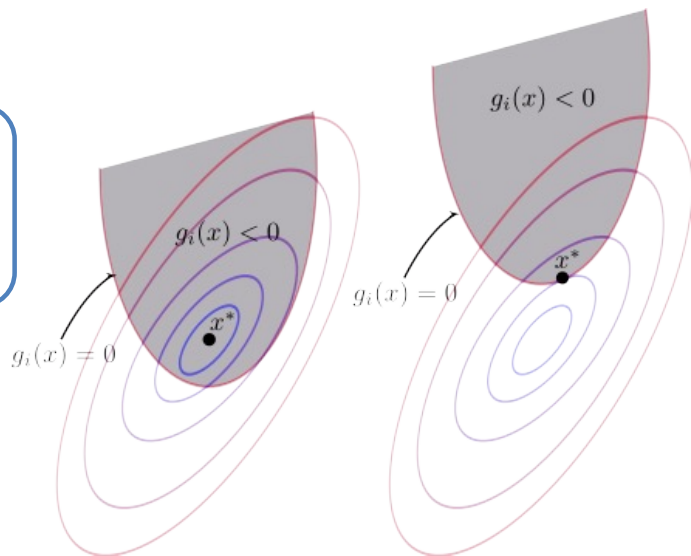
Optimize $\min_{\vec{w}} f(\vec{w})$

such that $\underline{g_i(\vec{w}) \leq 0}$ and $\underline{h_j(\vec{w}) = 0}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$

Define **Lagrangian**:

$$\underline{L(\vec{w}, \vec{\alpha}, \vec{\beta}) := f(\vec{w}) + \sum_{i=1}^m \alpha_i g_i(\vec{w}) + \sum_{j=1}^n \beta_j h_j(\vec{w})}$$

Here, $\vec{\alpha}$ and $\vec{\beta}$ are Lagrange multipliers



(KKT)

➤ Karush(1939)–Kuhn–Tucker (1951) Theorem

- Suppose $f(\vec{w})$ and $g_i(\vec{w})$ are "convex."
- Suppose $h_j(\vec{w})$ are affine.
- Suppose there exists \vec{w}_0 such that $g_i(\vec{w}_0) < 0$ for all $1 \leq i \leq n$.

e.g. $h(\vec{w}) = \vec{w}^T \vec{a} + b$

Under the above assumptions, the previous optimization question (★) has a solution \vec{w}^* if and only if there exist $\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*$ satisfying the following

Karush-Kuhn-Tucker (KKT) conditions:

$$\nabla_{\vec{w}} L(\vec{w}, \vec{\alpha}^*, \vec{\beta}^*) = 0$$

$$g_i(\vec{w}) \leq 0 \text{ for } 1 \leq i \leq m$$

$$h_j(\vec{w}) = 0 \text{ for } 1 \leq j \leq n$$

$$\vec{\alpha}_i^* \geq 0$$

$$\alpha_i^* g_i(\vec{w}^*) = 0 \text{ for } 1 \leq i \leq m \quad (\text{complementary slackness})$$

$$\begin{bmatrix} 2 & & \\ & \ddots & \\ & & 2 \end{bmatrix}$$

Application to SVM optimization: find \vec{w} and b

$$\min_{\vec{w}, b} \left(\frac{1}{2} \vec{w}^T \vec{w} \right) \text{ such that } \underline{1 - y^{(i)} (\vec{w}^T \vec{x}^{(i)} + b) \leq 0} \quad \text{for } 1 \leq i \leq n$$

Define Lagrangian:

$$L(\vec{w}, b, \vec{\alpha}) := \left(\frac{1}{2} \vec{w}^T \vec{w} \right) + \sum_{i=1}^n \alpha_i \left(1 - y^{(i)} (\vec{w}^T \vec{x}^{(i)} + b) \right)$$

Final Goal
 $\vec{w}^T \vec{x} + b = 0$

By KKT conditions: $\vec{w} - \sum_{i=1}^n \alpha_i y^{(i)} \vec{x}^{(i)} = 0$

• $\nabla_{\vec{w}} L = 0$ implies

$$\vec{w} = \sum_{i=1}^n \alpha_i y^{(i)} \vec{x}^{(i)}$$

• $\frac{\partial}{\partial b} L = 0$ implies

$$\sum_{i=1}^n \alpha_i y^{(i)} = 0$$

- $1 - y^{(i)}(\vec{w}^T \vec{x}^{(i)} + b) \leq 0$
- $\alpha_i \geq 0$
- $\alpha_i (1 - y^{(i)}(\vec{w}^T \vec{x}^{(i)} + b)) = 0$

From complementary slackness condition

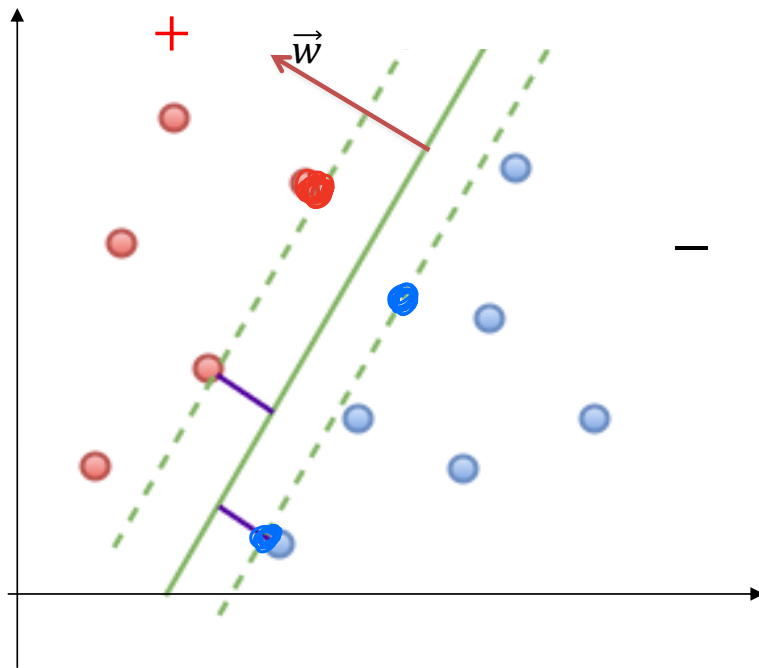
- If $\alpha_i > 0$, then $y^{(i)}(\vec{w}^T \vec{x}^{(i)} + b) = 1$

So, $\vec{x}^{(i)}$ is a support vector.

- If $y^{(i)}(\vec{w}^T \vec{x}^{(i)} + b) > 1$, then, $\alpha_i = 0$

So, if $\vec{x}^{(i)}$ is away from boundary, then we don't use those points.

Only support vectors matter!



Data: $\{(\vec{x}^{(i)}, \vec{y}^{(i)}) \mid i=1, \dots, n\}$

Plug the condition formulas back to the Lagrangian,

$$\vec{w} = \sum_{i=1}^n \alpha_i y^{(i)} \vec{x}^{(i)}$$

$$L(\vec{w}, b, \vec{\alpha}) := \frac{1}{2} \vec{w}^T \vec{w} + \sum_{i=1}^n \alpha_i (1 - y^{(i)} (\vec{w}^T \vec{x}^{(i)} + b))$$

$$= \frac{1}{2} \vec{w}^T \vec{w} + \sum_{i=1}^n \alpha_i - \vec{w}^T \vec{w}$$

$$= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle =: F(\vec{\alpha})$$

$$\vec{x}^{(i)} \cdot \vec{x}^{(j)} = \vec{x}^{(i)T} \vec{x}^{(j)}$$

The new question is to optimize the dual Lagrangian $F(\vec{\alpha})$ with constraints:

$$\max_{\vec{\alpha}} F(\vec{\alpha}) \text{ subject to } \sum_{i=1}^n \alpha_i y^{(i)} = 0 \text{ and } \alpha_i \geq 0$$

$$\text{e.g. } \alpha_1 + \alpha_2 - \alpha_3 - \dots = 0$$

Find $\vec{\alpha}$

There is a [Sequential minimal optimization \(SMO\) algorithm](#) for solving this quadratic programming problem. (1998 by John Platt)

After finding optimal α_i , we can plug back to find optimal \vec{w} .

From the distance formula, the intersection term can be calculated by **one**

support vector $(\vec{x}^{(s)}, y^{(s)} = 1)$

$$b = 1 - \vec{w}^T \vec{x}^{(s)} = 1 - \sum_{i=1}^n \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x}^{(s)} \rangle$$

$$\vec{w} = \sum_{i=1}^n \alpha_i y^{(i)} \vec{x}^{(i)}$$

Or we want to use **all support vectors** $\{(\vec{x}^{(s)}, y^{(s)}) \mid s \in S\}$ and take average for numerically stable solution:

$$b = \frac{1}{|S|} \sum_{s \in S} (y^{(s)} - \vec{w}^T \vec{x}^{(s)}) = \frac{1}{|S|} \sum_{s \in S} \left(y^{(s)} - \sum_{i=1}^n \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x}^{(s)} \rangle \right)$$

Or we want to start with **original data** in computation formula:

$$b = - \frac{\min_{i: y^{(i)}=1} \vec{w}^T \vec{x}^{(i)} + \max_{i: y^{(i)}=-1} \vec{w}^T \vec{x}^{(i)}}{2}$$

➤ **Prediction:** \hat{y}_i

$$\vec{w} \quad b$$

$$\vec{w}^T \vec{x} + b = 0$$

After we have the optimal model (parameters) \vec{w}^T, b , we can make predictions for a test data point \vec{x} :

$$= w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

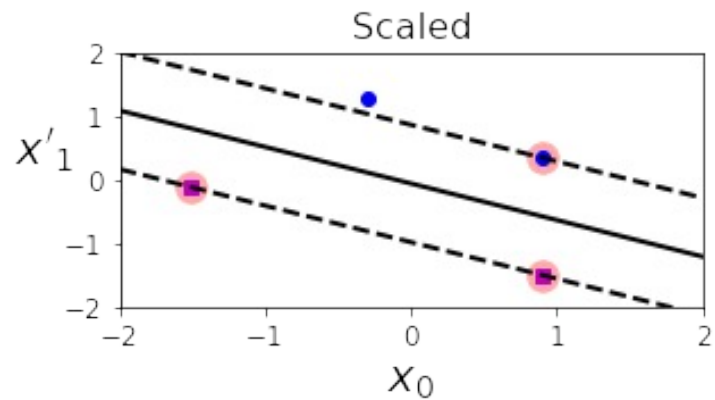
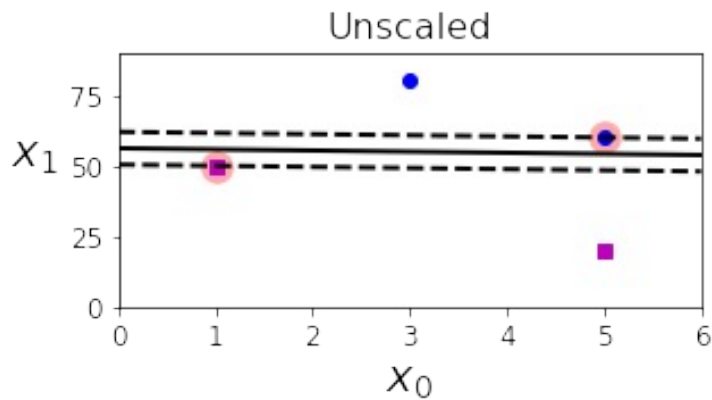
$$f(\vec{x}) = \vec{w}^T \vec{x} + b = \sum_{i=1}^n \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b$$

- Only involves **inner product** of the input data $\{(\vec{x}^{(i)}, y^{(i)}), i = 1, \dots, n\}$!
- $\alpha_i = 0$ except for support vectors. So the formula can also be written as

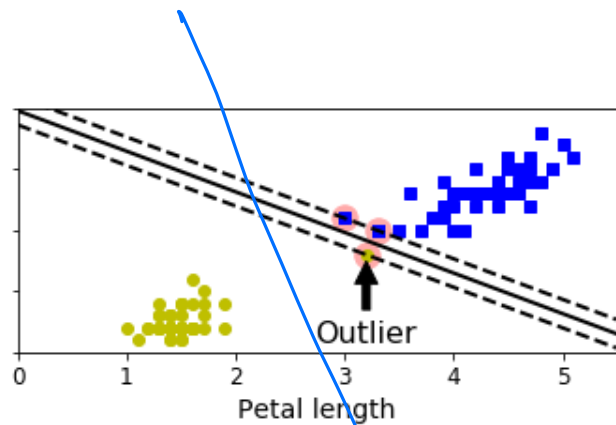
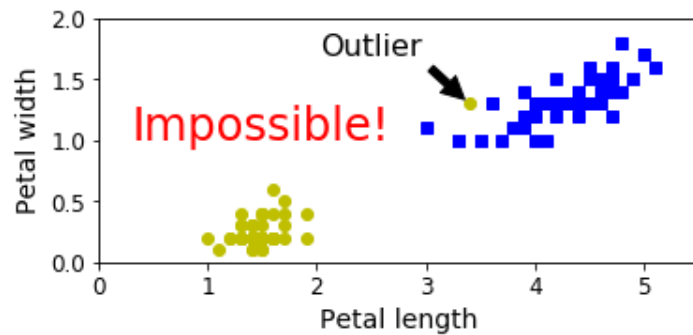
$$f(\vec{x}) = \sum_{i \in S} \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b$$

where S is the set of indices of support vectors.

Sensitivity to feature scales



Outlier:

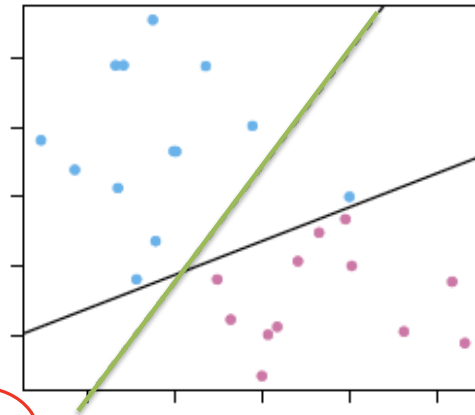
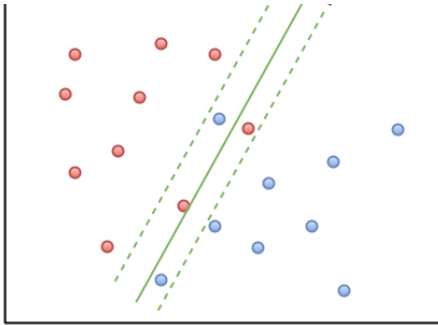


➤ Non-separable cases:

- In general, the two classes are usually **not separable** by any hyperplane.
- Even if they are, the max margin may not be desirable because of its high variance, and thus possible over-fit.
- The generalization of the maximal margin classifier to the non-separable case is known as the **support vector classifier**.
- Use a **soft-margin** in place of the max margin.
- **Soft-margin classifier** (support vector classifier) allow some violation of the margin: some can be on the wrong side of the margin (in the river) or even wrong side of the hyperplane.

➤ **Non-separable cases:** (Soft-margin SVM classifier)

If the datasets are **not linearly separable**, or we want SVM less sensitive to **outliers**.



Soft-margin SVM optimization: find \vec{w} and b

$$\min_{\vec{w}, b} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^n \xi_i$$

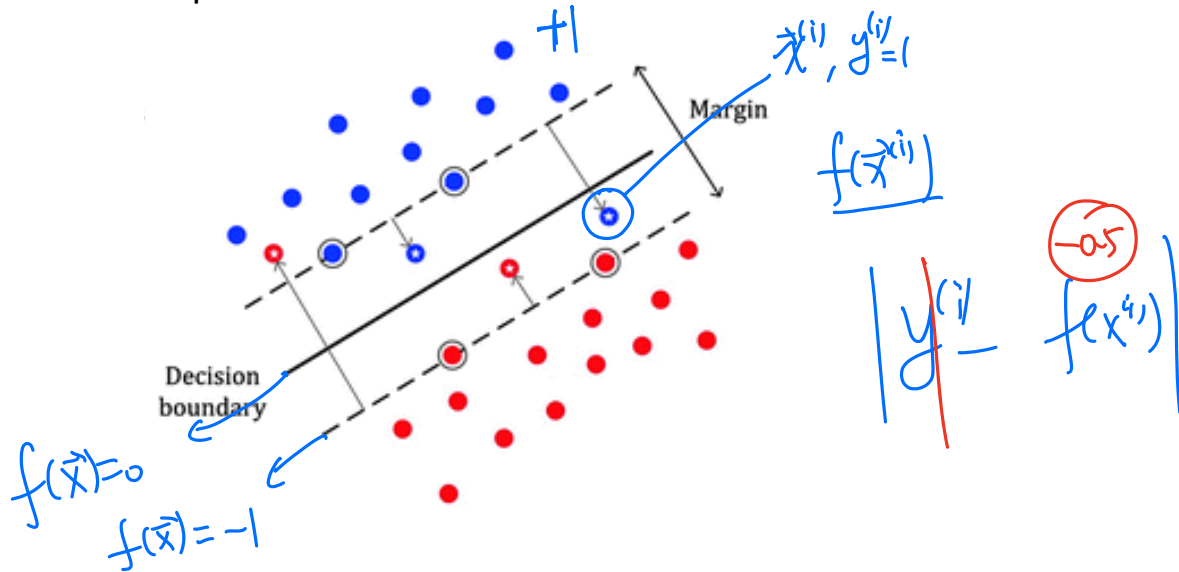
$$\text{such that } 1 - \xi_i - y^{(i)} (\vec{w}^T \vec{x}^{(i)} + b) \leq 0$$

$$\text{for } 1 \leq i \leq n$$

Here, for each training point, we introduce $\xi_i \geq 0$, which is called a slack variable.

$$\xi_i := \begin{cases} 0 & \text{for data points on or inside the correct margin boundary} \\ |y^{(i)} - f(x^{(i)})| & \text{for other points, where } f(\vec{x}^{(i)}) = \vec{w} \cdot \vec{x}^{(i)} + b \end{cases}$$

- $0 < \xi_i < 1$ for data points inside the margin, but on the correct side of the decision boundary.
- $\xi_i = 1$ for data points on the decision boundary.
- $\xi_i > 1$ for data points will be misclassified.



The classification constraints $y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1$ will be replaced by

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1 - \xi_i$$

Now we maximize the margin while **softly** penalizing points that lie on the wrong side of the margin boundary (**Soft-margin SVM optimization**)

$$\min_{\vec{w}, b} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^n \xi_i$$

$$\text{such that } 1 - \xi_i - y^{(i)}(\vec{w}^T \vec{x}^{(i)} + b) \leq 0 \quad \text{for } 1 \leq i \leq n$$

Similarly as hard-margin SVM, we use Lagrangian and KKT to simplify the optimization question to be

$$\max_{\vec{\alpha}} F(\vec{\alpha}) := \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

subject to

$$\sum_{i=1}^n \alpha_i y^{(i)} = 0$$

$$\text{and } 0 \leq \alpha_i \leq C \quad \text{for } 1 \leq i \leq n$$

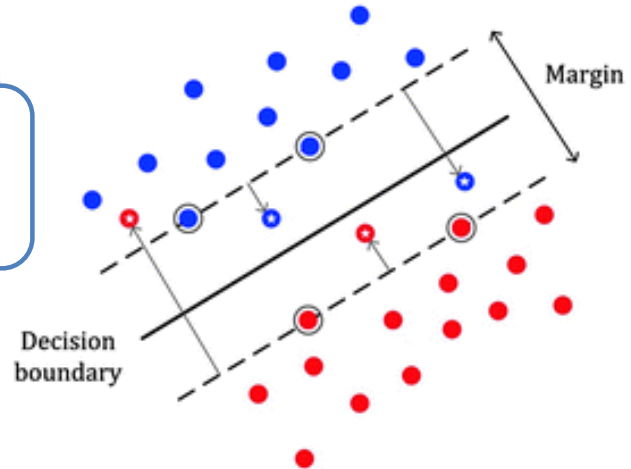
- The hyperparameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.
- If $C \rightarrow \infty$, it recover the hard-margin SVM.

The **intersection term** can be calculated by **one support vector** $(\vec{x}^{(s)}, y^{(s)} = 1)$ with $0 \leq \alpha_i \leq C$ and $\xi_i = 0$

$$b = 1 - \sum_{i=1}^n \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x}^{(s)} \rangle$$

Or we want to use set of **support vectors** $\{(\vec{x}^{(s)}, y^{(s)}) \mid s \in M\}$ with $0 \leq \alpha_i \leq C$ and $\xi_i = 0$, and take average for numerically stable solution:

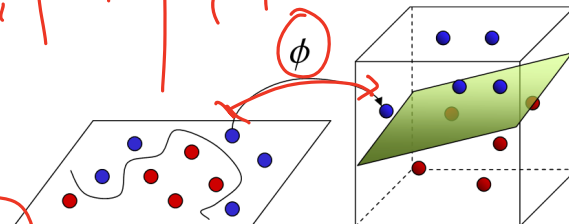
$$b = \frac{1}{|M|} \sum_{s \in M} \left(y^{(s)} - \sum_{i=1}^n \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x}^{(s)} \rangle \right)$$



➤ The kernel method

For any linear method (e.g., linear regression, logistics regression, LDA), we can easily generalize it to non-linear method by introducing new variables (features).

x_1	x_2	x_1^2
\vdots	\vdots	\vdots



Input Space

Feature Space

For example,

$$z_1 = x_1, z_2 = x_2,$$

$$z_3 = x_1^2, z_4 = x_2^2, z_5 = x_1 x_2,$$

$$z_6 = x_1^3, z_7 = x_2^3, z_8 = x_1^2 x_2, z_9 = x_1 x_2^2,$$

Formally, we can consider this procedure as defining a **feature map**:

$$\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$$

$$\vec{x} \rightarrow \phi(\vec{x})$$

$$2z_1 + 3z_2 - z_3 = 0$$

$$2x_1 + 3x_2 - x_1^2 = 0$$

$$d=100$$

$$\text{degree } 4 = m$$

The **difficulty** is that dimension D is very large or even infinite.

For example, using polynomial of degree m , there are $D \sim O(d^m)$ parameters.

For a relatively easy question, if $d = 100$ and $m = 4$, there are about $d^4 \approx 4$ million parameters!

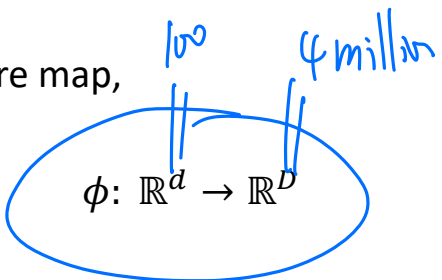
Question: How to solve the difficulty?

Answer: The kernel method (trick).

$$O(d)$$

Suppose there is a machine learning model, in the optimization of the cost and the prediction formula, only **inner products** of the data points are involved: $\langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$, or $\langle \vec{x}^{(i)}, \vec{x} \rangle$ for prediction for \vec{x} .

After we applied the feature map,



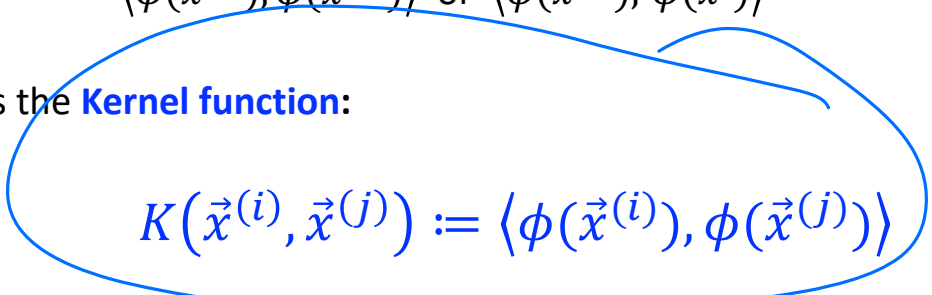
$\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$

all calculations will be replaced by $\phi(\vec{x}) \in \mathbb{R}^D$. (Very large dimension)

We assume that all calculations **only involve inner products**

$$\langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \rangle \text{ or } \langle \phi(\vec{x}^{(i)}), \phi(\vec{x}) \rangle$$

Define it as the **Kernel function**:


$$K(\vec{x}^{(i)}, \vec{x}^{(j)}) := \langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \rangle$$

Example: (quadratic)

$$\phi: \mathbb{R}^3 \rightarrow \mathbb{R}^9$$

For \vec{x} and $\vec{z} \in \mathbb{R}^3$, consider the quadratic feature map:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\phi(\vec{x}) := \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix} \in \mathbb{R}^{3^2}$$

$$\phi(\vec{z}) = \begin{bmatrix} z_1 z_1 \\ z_1 z_2 \\ \vdots \\ z_3 z_3 \end{bmatrix}$$

The kernel function:

$$K(\vec{x}, \vec{z}) := \langle \phi(\vec{x}), \phi(\vec{z}) \rangle = \sum_{i=1}^d \sum_{j=1}^d x_i x_j z_i z_j$$

$$\sim O(d^2)$$

$$= \left(\sum_{i=1}^d x_i z_i \right) \left(\sum_{j=1}^d x_j z_j \right) = \left(\sum_{i=1}^d x_i z_i \right)^2 = (\vec{x}^T \vec{z})^2 \sim O(d)$$

$$\vec{w}^T \vec{x} + b = 0$$

Recall that in hard-margin SVM,

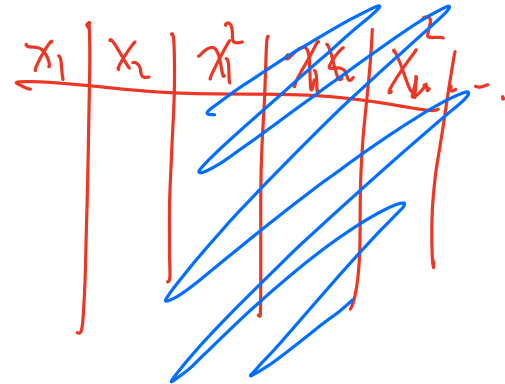
$$\max_{\vec{\alpha}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y^{(i)} y^{(j)} \alpha_i \alpha_j \underbrace{\phi(\vec{x}^{(i)}, \vec{x}^{(j)})}_{=: K(\vec{x}^{(i)}, \vec{x}^{(j)})}$$

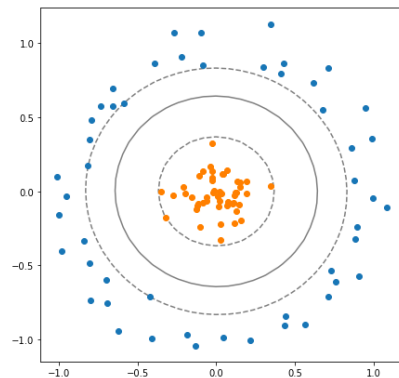
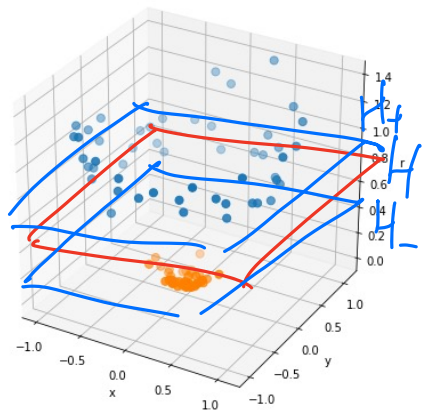
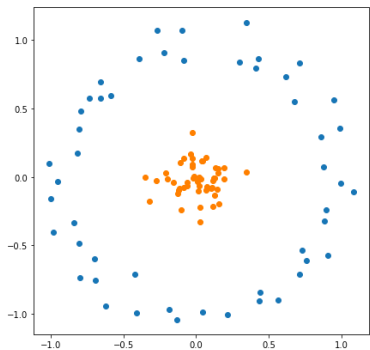
$$\text{subject to } \sum_{i=1}^n \alpha_i y^{(i)} = 0 \quad \text{and } \alpha_i \geq 0 \text{ for } 1 \leq i \leq n$$

$$b = 1 - \sum_{i=1}^n \alpha_i y^{(i)} \underbrace{\phi(\vec{x}^{(i)}, \vec{x}^{(s)})}_{=}$$

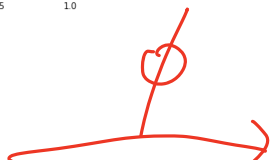
Prediction:

$$f(\vec{x}) = \sum_{i=1}^n \alpha_i y^{(i)} \underbrace{\phi(\vec{x}, \vec{x}^{(i)})}_{=}$$





Data in \mathbb{R}^2



Feature space \mathbb{R}^3

Boundary:

$$\sum_{i=1}^n \alpha_i y^{(i)} K(\vec{x}^{(i)}, \vec{x}) + b = 0$$

➤ Kernel Functions

1. Quadratic Kernel

For \vec{x} and $\vec{z} \in \mathbb{R}^d$, define kernel function:

$$K(\vec{x}, \vec{z}) := (\vec{x}^T \vec{z} + c)^2$$

What is the feature map $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$?

$$\phi(\vec{x}) := \begin{bmatrix} x_1 x_1 \\ \vdots \\ x_1 x_d \\ \vdots \\ x_d x_d \\ \sqrt{2c} x_1 \\ \vdots \\ \sqrt{2c} x_d \\ c \end{bmatrix} \in \mathbb{R}^{d^2+d+1}$$

Do we need the feature map ϕ ?

$$\langle \underline{\phi(\vec{x})}, \underline{\phi(\vec{z})} \rangle$$

$$\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$$

$$D = d^2 + d + 1$$

$$\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$$

2. Polynomial Kernel

thm:

For \vec{x} and $\vec{z} \in \mathbb{R}^d$, define degree n polynomial kernel function:

$$K(\vec{x}, \vec{z}) := (\vec{x}^T \vec{z} + c)^n$$

$O(d)$

3. Sigmoid Kernel

For \vec{x} and $\vec{z} \in \mathbb{R}^d$, define Sigmoid kernel function:

$$K(\vec{x}, \vec{z}) := \tanh(\eta \vec{x}^T \vec{z} + c)$$

$$\text{where } \tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$$

4. Gaussian Kernel

For \vec{x} and $\vec{z} \in \mathbb{R}^d$, define Gaussian kernel function:

Thm:

$$K(\vec{x}, \vec{z}) := \exp\left(-\frac{\|\vec{x} - \vec{z}\|^2}{2\sigma^2}\right)$$

$$\langle \phi(\vec{x}), \phi(\vec{z}) \rangle$$



Remark:

- If σ is very small, then overfitting. If σ is very large, then underfitting

- What is the feature map $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$?

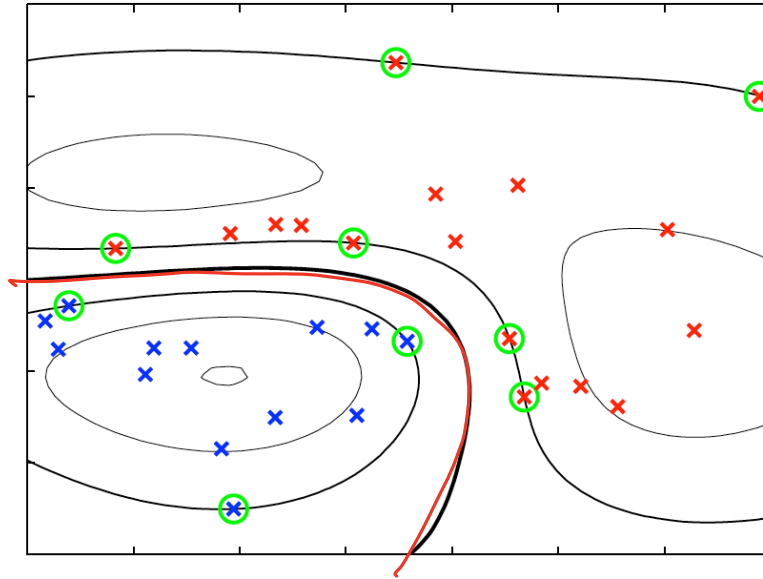
$$\textcircled{d=1}$$

$$x \rightarrow \phi(x) =$$

$$\begin{bmatrix} 1 \\ x/\sqrt{2}\sigma \\ x^2/\sigma^2\sqrt{2} \\ x^3/\sigma^3\sqrt{2} \\ \vdots \\ 1 \end{bmatrix}$$

$$\in \mathbb{R}^\infty$$

Example: SVM with kernel trick



Example of two classes in two dimensions showing contours of constant $f(\vec{x})$ obtained from a support vector machine having a **Gaussian kernel** function. Also shown are the decision boundary, the margin boundaries, and the support vectors.

scikit-learn

- SVM:

<https://scikit-learn.org/stable/modules/svm.html#svm>

- Kernel Functions:

<https://scikit-learn.org/stable/modules/svm.html#kernel-functions>

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$, where r is specified by `coef0`.

$$K: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$$

$$K(\vec{x}, \vec{z}) = \langle \phi(\vec{x}), \phi(\vec{z}) \rangle$$

How to show a map is a feature maps?

Theorem: (Mercer 1909)

$$K(\vec{x}, \vec{z}) = \langle \phi(\vec{x}), \phi(\vec{z}) \rangle$$

Let $K: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be a binary map.

The map K is a **kernel function** if and only if for any finite sequence $\{\vec{x}^{(1)}, \dots, \vec{x}^{(m)}\}$, the matrix

$\in \mathbb{R}^{m \times m}$

$$M = \begin{bmatrix} \vdots & & \\ \dots & K(\vec{x}^{(i)}, \vec{x}^{(j)}) & \dots \\ \vdots & & \end{bmatrix}$$

$$= \begin{bmatrix} K(\vec{x}^{(1)}, \vec{x}^{(1)}) & \dots & K(\vec{x}^{(1)}, \vec{x}^{(m)}) \\ \vdots & \ddots & \vdots \\ K(\vec{x}^{(m)}, \vec{x}^{(1)}) & \dots & K(\vec{x}^{(m)}, \vec{x}^{(m)}) \end{bmatrix}$$

is **symmetric** and **positive semi-definite**.

$$M^T = M$$

\Downarrow
all eigenvalues ≥ 0

Proof:

" \Rightarrow "

If K is a kernel function, then there exists a map $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$ such that

$$K(\vec{x}^{(i)}, \vec{x}^{(j)}) = \langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \rangle$$

First, $K(\vec{x}^{(i)}, \vec{x}^{(j)}) = K(\vec{x}^{(j)}, \vec{x}^{(i)})$ by the property of inner product.

Second, the quadratic form

$$\begin{aligned} \vec{z}^T M \vec{z} &= \sum_{i,j}^d z_i \langle \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) \rangle z_j = \sum_{i,j}^d \langle z_i \phi(\vec{x}^{(i)}), \phi(\vec{x}^{(j)}) z_j \rangle \\ &= \left\langle \sum_{i=1}^d z_i \phi(\vec{x}^{(i)}), \sum_{j=1}^d z_j \phi(\vec{x}^{(j)}) \right\rangle = \left\| \sum_{i=1}^d z_i \phi(\vec{x}^{(i)}) \right\|^2 \geq 0 \end{aligned}$$

M defined by inner product this way is called the **Gram matrix**.

$$f(\vec{x}, \vec{y}) = x_1 y_1 + x_2 y_2$$

$$f\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \vec{y}\right) = y_1 + 2y_2$$

" \Leftarrow "

Suppose K is a binary map such that $M = [K(\vec{x}^{(i)}, \vec{x}^{(j)})]$ satisfies the properties.

Consider $\phi_{(\vec{x})}(-) := K(-, \vec{x})$, which is map from \mathbb{R}^n to \mathbb{R} .

Let $\mathcal{F} := \text{Span}\{\phi_{(\vec{x})} \mid \vec{x} \in \mathbb{R}^n\}$ be a subspace of the function space $C(\mathbb{R}^n, \mathbb{R})$

Claim 1. $\phi_{(\vec{x})}$ defines a map from \mathbb{R}^n to \mathcal{F} .

Claim 2. \mathcal{F} is an **inner product space** with

$$\langle \phi_{(\vec{x})}, \phi_{(\vec{z})} \rangle_{\mathcal{F}} := K(\vec{x}, \vec{z})$$

How to construct feature maps?

kernel

Theorem:

$$K_1(\vec{x}, \vec{z}) = \vec{x}^T \vec{z} + c$$

If K_1 and K_2 are kernel functions, then the following are also kernel functions.

- $K(\vec{x}, \vec{z}) := \underline{aK_1(\vec{x}, \vec{z}) + bK_2(\vec{x}, \vec{z})}$, where $a, b \geq 0$
- $K(\vec{x}, \vec{z}) := \underline{K_1(\vec{x}, \vec{z})K_2(\vec{x}, \vec{z})}$
- $K(\vec{x}, \vec{z}) := K_1(f(\vec{x}), f(\vec{z}))$, where f is a function from $\mathbb{R}^d \rightarrow \mathbb{R}^M$
- $K(\vec{x}, \vec{z}) := P(K_1(\vec{x}, \vec{z}))$, where $P(t)$ is a polynomial with non-negative coefficients.
- $K(\vec{x}, \vec{z}) := \exp(K_1(\vec{x}, \vec{z}))$
- $K(\vec{x}, \vec{z}) := \vec{x}^T S \vec{z}$, where S is a symmetric positive semidefinite matrix.
- $K(\vec{x}, \vec{z}) := f(\vec{x})K_1(\vec{x}, \vec{z})f(\vec{z})$, where $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is any function.

➤ Support Vector Machine - Regression (SVR)

Support Vector Machine can also be used as a **regression** method, maintaining all the main features that characterize the algorithm (maximal margin).

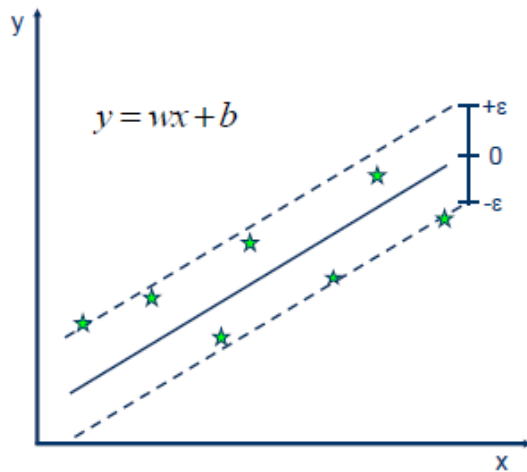
First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities.

In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem.

But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration.

However, the **main idea** is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

Support Vector Machine - Regression (SVR)



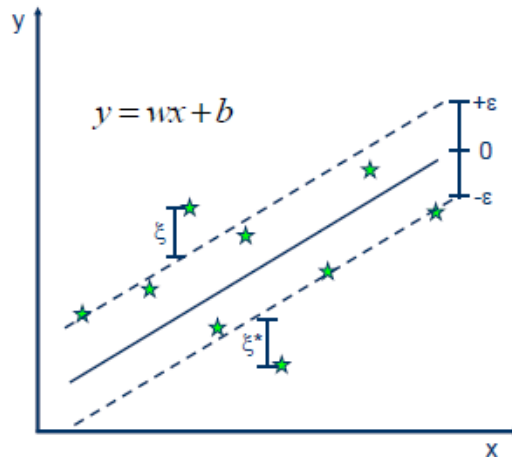
- Solution:

$$\min \frac{1}{2} \|w\|^2$$

- Constraints:

$$y_i - wx_i - b \leq \varepsilon$$

$$wx_i + b - y_i \leq \varepsilon$$



- Minimize:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*)$$

- Constraints:

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$

$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$

Linear SVR

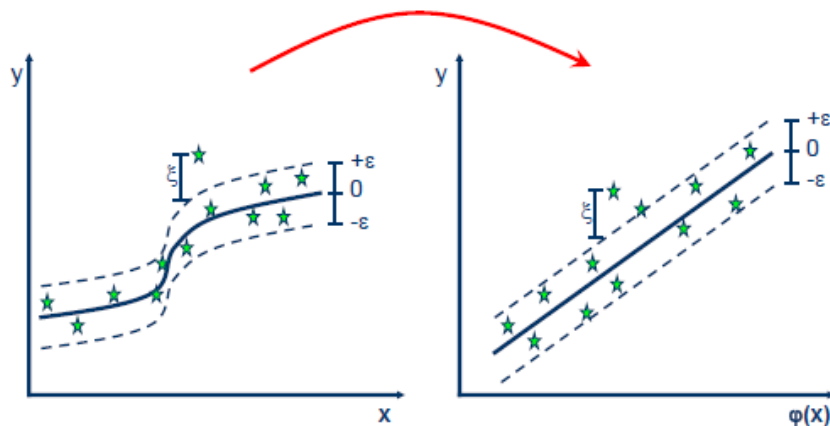
$$y = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$

Non-linear SVR

The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

$$y = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \cdot \langle \phi(x_i), \phi(x) \rangle + b$$

$$y = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b$$



➤ Apply Kernel Methods to **Linear Regressions**:

$$\phi: \mathbb{R}^d \rightarrow \mathbb{R}^D$$

Data: $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) \mid i = 1, \dots, n\}$

★ **Model:** $h(\vec{x}) = \vec{\theta}^T \vec{x}$ — NOT inner prod. $X = \begin{bmatrix} \vec{x}^{(1,T)} \\ \vec{x}^{(2,T)} \\ \vdots \\ \vec{x}^{(n,T)} \end{bmatrix} X^T = [\vec{x}^{(1)} \ \vec{x}^{(2)} \ \dots \ \vec{x}^{(n)}]$

If the **mean** of the data matrix X is **zero**, **Ridge regression** cost function:

$$J^{Ridge}(\vec{\theta}) := (X\vec{\theta} - \vec{y})^T (X\vec{\theta} - \vec{y}) + \lambda \vec{\theta}^T \vec{\theta}$$

★ The optimal solution is

$$\vec{\theta} = (X^T X + \lambda I)^{-1} X^T \vec{y}$$

$$\begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$$

Define $\vec{\theta} = X^T \vec{\beta}$ for some new parameter vector $\vec{\beta} \in \mathbb{R}^n$, called dual parameters

$$\vec{\theta} = X^T \vec{\beta} = [\vec{x}^{(1)} \ \dots \ \vec{x}^{(n)}] \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = \sum_{i=1}^n \beta_i \vec{x}^{(i)}$$



The **dual model for linear regression** is

$$h(\vec{x}) = \vec{\theta}^T \vec{x} = \langle \vec{x}, \vec{\theta} \rangle = \sum_{i=1}^n \beta_i \langle \vec{x}, \vec{x}^{(i)} \rangle$$

$$K(\vec{x}, \vec{x}^{(i)})$$

The **cost function**

$$J(\vec{\beta}) = 0 \quad J^{Ridge}(\vec{\beta}) := \underbrace{(XX^T \vec{\beta} - \vec{y})^T (XX^T \vec{\beta} - \vec{y}) + \lambda \vec{\beta}^T XX^T \vec{\beta}}$$

Solutions of $\vec{\beta}$ for optimizing the cost function:

$$\vec{\beta} = (XX^T + \lambda I)^{-1} \vec{y}$$

$$K(\vec{x}^{(i)}, \vec{x}^{(j)})$$

$$\text{Here, } XX^T = \begin{bmatrix} \vdots & \vdots & \vdots \\ \dots \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle \dots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Now you can apply the **kernel tricks** to the dual linear model.

References:

- Chapter 7 in Pattern Recognition and Machine Learning by Chris Bishop.
- Chapter 9 An introduction to Statistical Learning by James, Witten, Hastie, Tibshirani
- Chapter 12 The elements of Statistical Learning, by Hastie, Tibshirani, Friedman
- <https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf>