Support Vector Machine

Huy V. Vo

MASSP

2021

Learning problem

- Input data: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}^k$.
- Approximator: $f_w : \mathbb{R}^d \to \mathbb{R}^k$ where w is the parameters of f. Find w such that f predicts exactly the labels of unseen data, i.e., $f_w(x_t) = y_t$ for unseen data point x_t.
- Loss function $\ell : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}$. $\ell(f_w(x_t), y_t)$ is small if $f_w(x_t)$ is close to y_t.
- Test data and train data are supposed to be drawn from the same distribution $\to$ find w such that $f_w(x)$ and y are close on the training data.

Learning problem (cont)

- We solve the optimization problem:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \lambda \Omega(w),$$

  where $\Omega(w)$ is a regularization term on w and its weight λ. Functions ℓ, f_w and Ω are usually chosen convex to ease the optimization.
- Different models have different loss functions ℓ and/or approximators (f).

Linear models

Linear Regression (LinReg)

- $w = (w_0, w_1, ..., w_d)^T$ and $f_w(x) = \langle w, [1; x] \rangle$[a].
- $l(y, y') = \|y - y'\|^2$.
- $\Omega(w) = \|w\|_{L_2}^2, \|w\|_{L_1}, \ldots$

---

[a] $[1; x]$ is the Matlab notation for $\begin{pmatrix} 1 \\ x \end{pmatrix}$

Logistic Regression (LogReg)

- $w = (w_0, w_1, ..., w_d)^T$ and $f_w(x) = \sigma(\langle w, [1; x] \rangle)$ where $\sigma(x) = \frac{1}{1+\exp(-x)}$.
- $l(y, y') = -y' \log(y) - (1 - y') \log(1 - y)$.
- .. or in another way: $f_w(x) = \langle w, [1; x] \rangle$ and $l(y, y') = -y' \log(\sigma(y)) - (1 - y') \log(1 - \sigma(y))$.
- $\Omega(w) = \|w\|_{L_2}^2, \|w\|_{L_1}, \ldots$

Linear models

- The approximators of LinReg and LogReg are linear functions. They are linear models.

No Free Lunch (NFL) Theorem

Without having substantive information about the modeling problem, there is no single model that will always do better than any other model.

NFL consequences

- One should consider multiple models for a machine learning problems and choose the algorithm to focus on (validation) $\Rightarrow$ one should know multiple models.
- Additional knowledge about the data can be helpful in choosing models $\Rightarrow$ feature engineering, data visualization, ...

Linear models

- LogReg is often better than LinReg in practice. Why?
- This theorem is only true when considering all datasets.

Limitations of linear models

Linear models suppose a linear relation between the input and the output which is not true for many types of data.

- Today's lecture: Support Vector Machine (SVM), another linear model which enables adding non-linearity to data.

Support Vector Machine

- A linear model: $f_w(x) = \langle w, [1; x] \rangle$.
- Hinge loss: $\ell(y, y') = \max(0, 1 - yy')$.
- Optimization problem:

$$\min_w \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i)) + \lambda \|w\|^2. \tag{1}$$

- This is a convex optimization problem. It can be solved with appropriate solvers or gradient descent.

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} -\mathbb{1}_{1-y_i f_w(x_i)>0} y_i \begin{pmatrix} 1 \\ x_i \end{pmatrix} + 2\lambda w. \tag{2}$$

Hinge loss

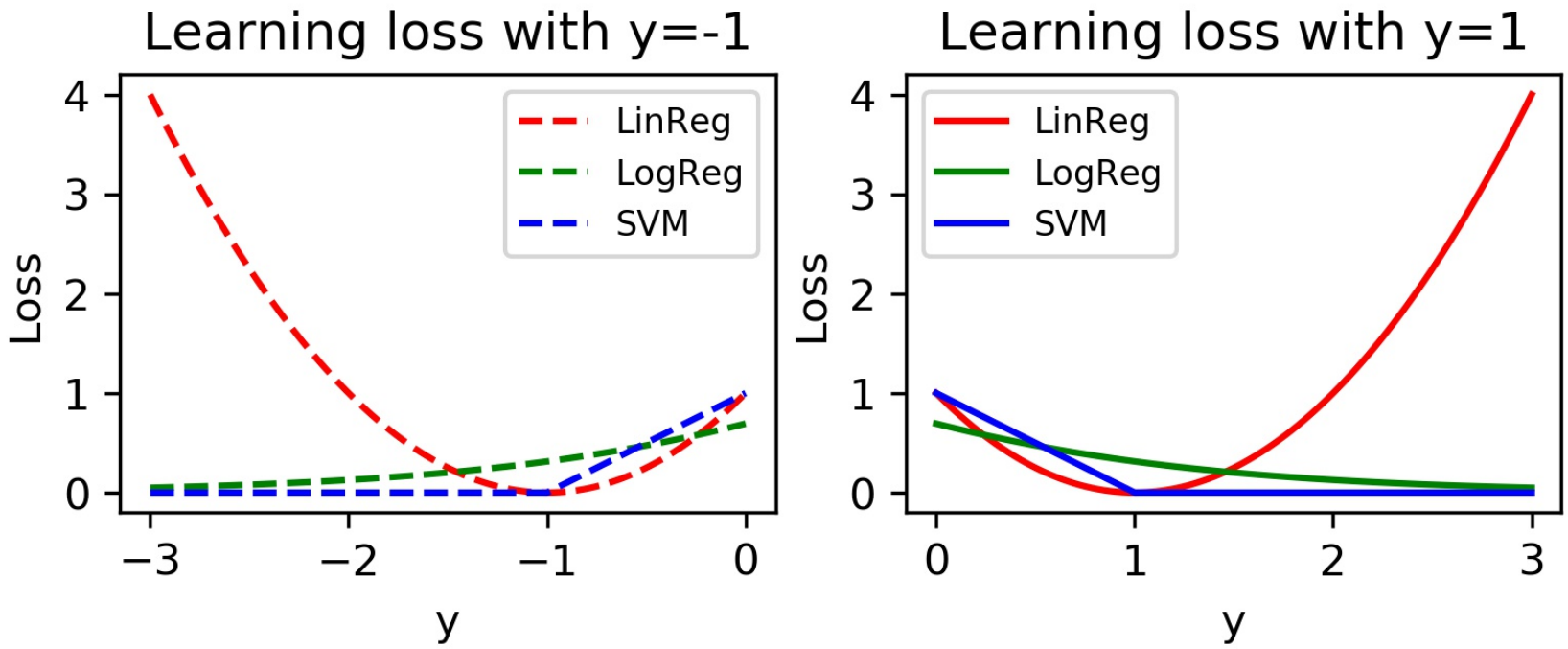


Figure: Visualizations of the loss function of linear regression, logistic regression and support vector machine.

Support Vector Machine

Possible questions

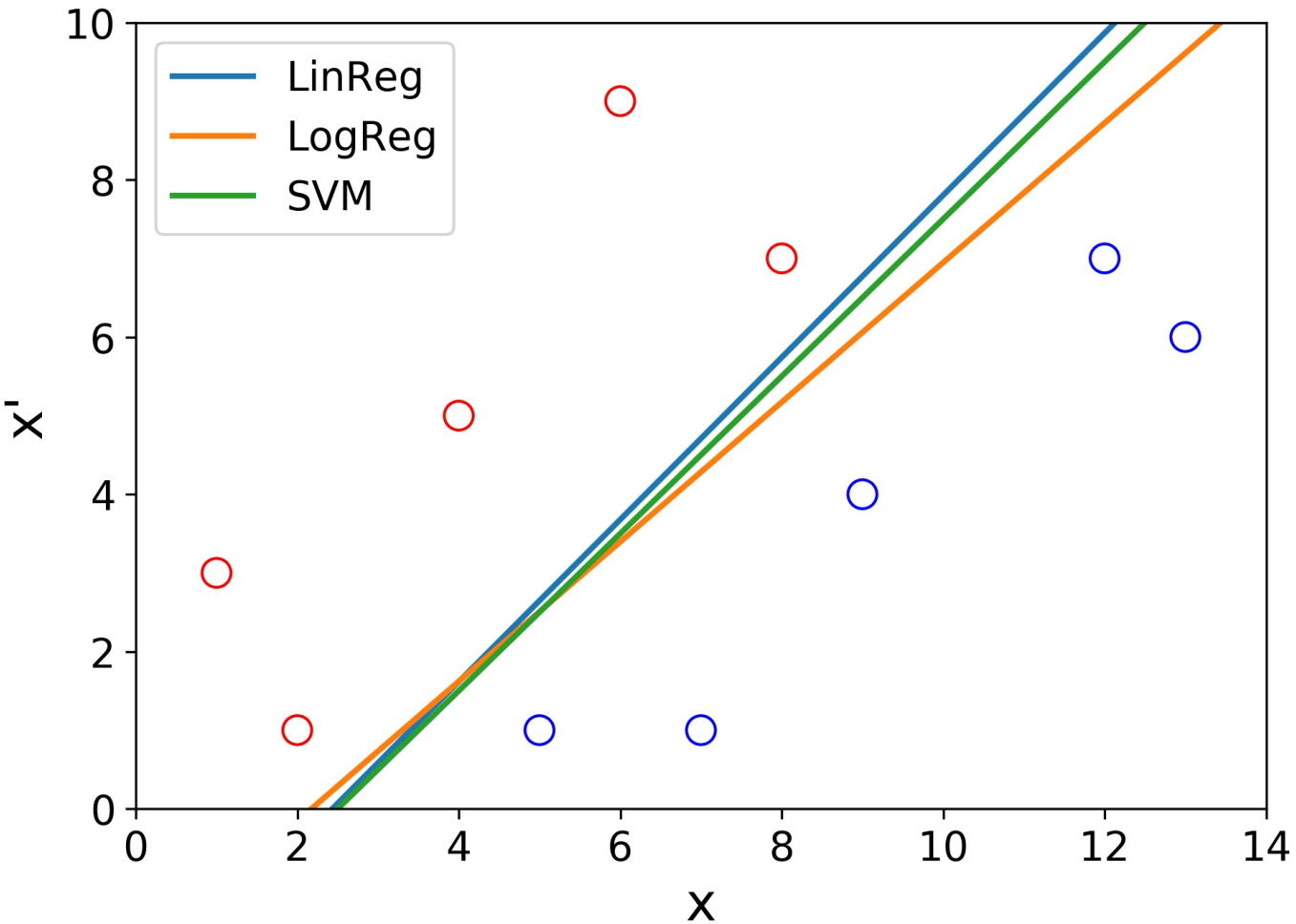
- What is special about SVM?
- What are support vectors?
- Why bothers with yet another loss function (Hinge loss)?
- How does SVM enable adding non-linearity to data?

Next

- SVM is a result from the Maximum Margin Problem.
- Non-linear SVM with kernels.

Toy dataset

- Data: $x_i \in \mathbb{R}^2, y_i \in \{\pm 1\}$.



x	x'	y
1	3	-1
2	1	-1
4	5	-1
6	9	-1
8	7	-1
5	1	1
7	1	1
9	4	1
12	7	1
13	6	1

Maximum margin problem: Linearly separable case

Problem statement

Given the dataset, find the linear boundary that is the furthest from both the negative ($y = -1$) and the positive ($y = 1$) set.

Properties

- Existence: such a boundary exists since the dataset is linearly separable.
- Uniqueness: exercise.
- The boundary is equidistant from the negative and the positive sets: Otherwise, one can move the boundary toward the further set to reduce the maximal distance to both sets. Formal proof: exercise.

Maximum margin problem: Linearly separable case

Problem formulation

- Boundary equation: $f_w(x) = \langle w, x \rangle + w_0 = 0$. We have $f(x) \leq 0$ for all x on the negative sets and $f(x) \geq 0$ for all x on the positive sets.
- Distance from x_i to the boundary: $d(x_i) = \frac{|f(x_i)|}{\|w\|} = \frac{|\langle x_i, w \rangle + w_0|}{\|w\|}$.
  item Support vectors: data points closest to the boundary.
- Distance from the support vectors: $d^* = \frac{1}{\|w\|}$.
- If $y_i = -1$ then $\frac{|\langle x_i, w \rangle + w_0|}{\|w\|} \geq \frac{1}{\|w\|} \Leftrightarrow f(x_i) \leq -1$. Similarly, if $y = 1$ then $f(x_i) \geq 1$.
- Optimization problem:

$$\min_w \frac{1}{2}\|w\|^2 \text{ such that } y_i(\langle x_i, w \rangle + w_0) \geq 1 \, \forall \, i. \tag{3}$$

Maximum margin problem: Linearly separable case

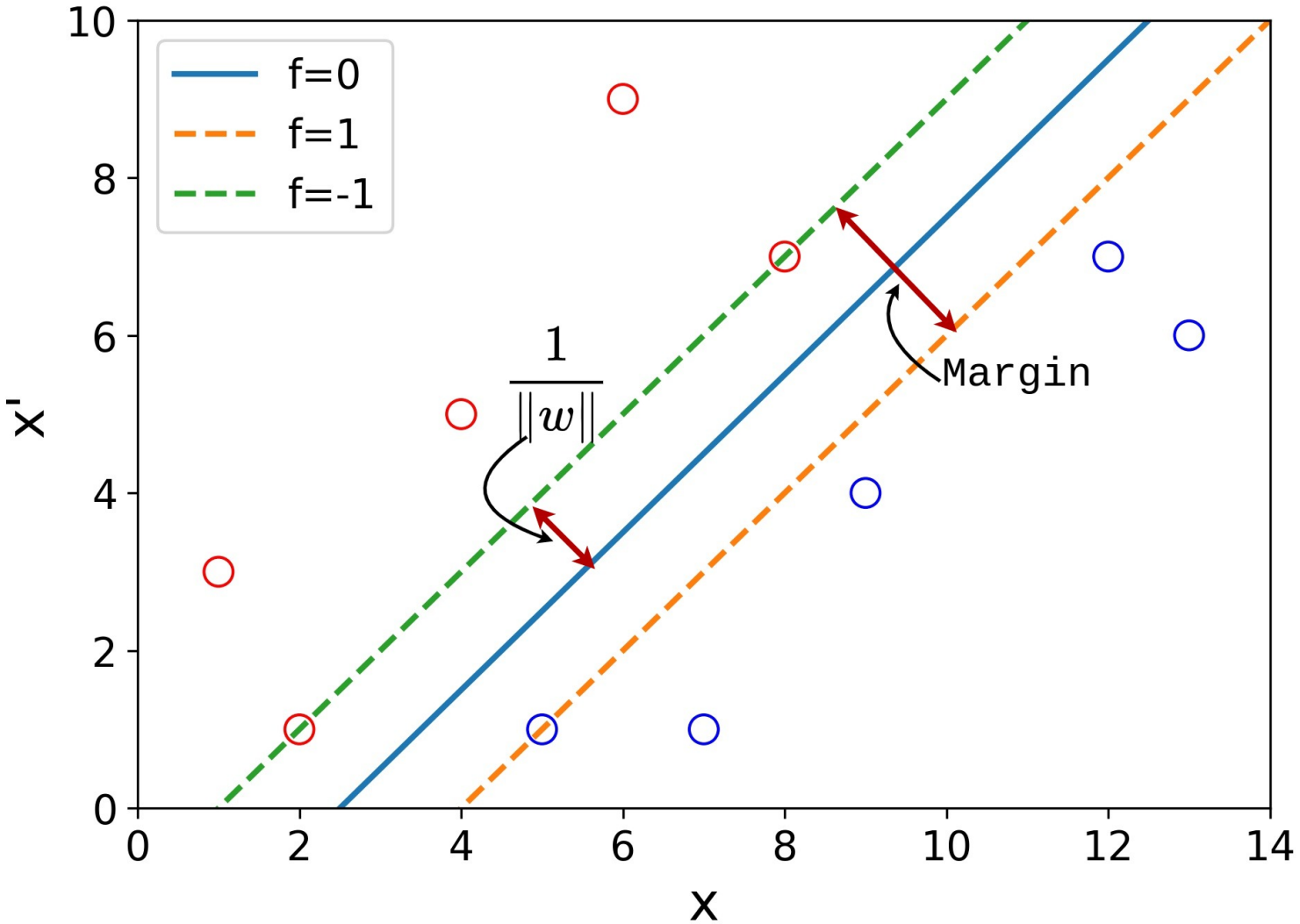


Figure: Maximum margin problem in linear separable case.

Maximum margin problem: Linearly separable case

Property

- Problem 3 is convex. It has a unique solution.
- There is no closed-form solution but it can be solved numerically.

Numerical solution

```
from qpsolvers import solve_qp
import numpy as np
n = 10
X = np.vstack(([1,2,4,6,8,5,7,9,12,13], [3,1,5,9,7,1,1,4,7,6], np.ones(10))).T
y = np.array([-1,-1,-1,-1,-1,1,1,1,1,1]).reshape(-1,1);
G = -X*y
h = -np.ones((n,))
P = np.array([[1,0,0],[0,1,0],[0,0,0]],dtype=np.float64)
q = np.zeros((3,))
solve_qp(P,q,G,h,solver='cvxopt')
```

```
array([ 0.66667, -0.66667, -1.66667])
```

$$w = [2/3, -2/3], w_0 = -5/3$$

Maximum margin problem: Linearly separable case

Limitations of formulation 3

- Algorithms for numerical approximation is slow if p is large.
- The role of support vectors is not clear.
- Cannot use kernel trick.

Dual problem

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \text{ s.t. } \alpha_i \geq 0 \, \forall \, i \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0.$$

- Karush-Kuhn-Tucker (KKT) condition: $\alpha_i [y_i (\langle x, w \rangle + w_0) - 1] = 0 \, \forall \, i$.
- Retrieve w: $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.
- Inferencce: $f(x) = \langle x, w \rangle + w_0 = \sum_{i=1}^{n} \alpha_i y_i \langle x, x_i \rangle + w_0$.

Maximum margin problem: Linearly separable case

Numerical solution

```
from qpsolvers import solve_qp
import numpy as np
n = 10
X = np.vstack(([1,2,4,6,8,5,7,9,12,13], [3,1,5,9,7,1,1,4,7,6])).T.astype(float)
y = np.array([-1,-1,-1,-1,-1,1,1,1,1,1]).reshape(-1,1).astype(float)
X = X*y
P = np.matmul(X,X.T);
q = -np.ones((n,))
G = -np.eye(n)
h = np.zeros((n,))
A = y.reshape((1,n))
b = np.array([0.0])
alpha = solve_qp(P,q,G,h,A,b,solver='cvxopt')
for el in alpha: print(f'{el:.3f}', end=' ')
```

```
0.000 0.333 0.000 0.000 0.111 0.444 0.000 0.000 0.000 0.000
```

Tmp

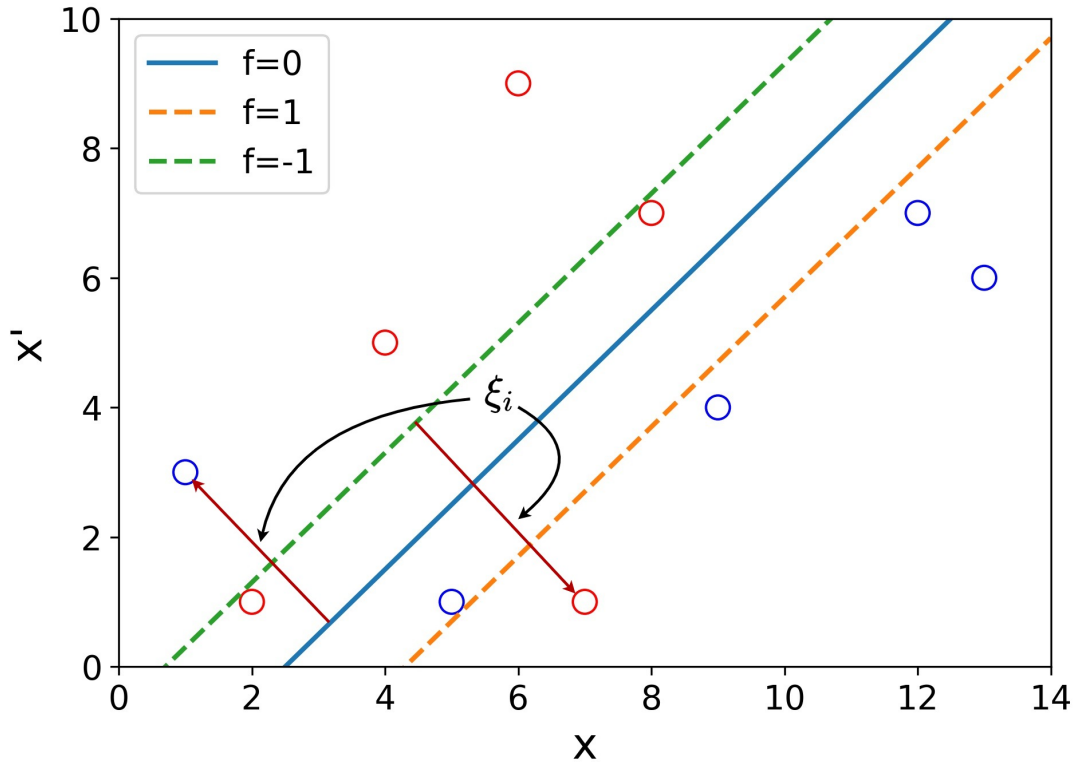
- $\alpha = [0, \frac{1}{3}, 0, 0, \frac{1}{9}, \frac{4}{9}, 0, 0, 0, 0]$,
  $w = \alpha_2 y_2 x_2 + \alpha_5 y_5 x_5 + \alpha_6 y_6 x_6 = [2/3, -2/3]$
- The points are weighted by α. Only support vectors have a positive α.

Maximum margin problem: Linearly separable case

Dual problem

- The role of support vectors is clear (KKT).
- The optimization does not depend on data dimension $\rightarrow$ can deal with high-dimension data (text, images, ...).
- Dual vectors are sparse $\rightarrow$ efficient sparse optimization methods can be used.
- Both the optimization and the inference do not depend on the vectors, only their inner products $\rightarrow$ kernel trick.

Maximum margin problem: Real data



- Misclassified error ξ_i: The amount needed to move x_i to the right side.
- If $y_i = 1$, $\langle x_i, w \rangle + w_0 + \xi_i \geq 1 \Leftrightarrow y_i(\langle x_i, w \rangle + w_0) + \xi_i \geq 1$.
- If $y_i = -1$,
  $\langle x, w_i \rangle + w_0 - \xi_i \leq -1 \Leftrightarrow y_i(\langle x_i, w \rangle + w_0) + \xi_i \geq 1$.
- Maximize the margin while also minimizing the total misclassified error.

Maximum margin problem: Real data

Primal problem

$$\min_{\beta,\beta_0,\xi_i} \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{n} \xi_i \quad \text{s.t. } y_i(\langle x, \beta\rangle + \beta_0) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \,\forall\, i.$$

This formulation is equivalent to the formulation with Hinge loss.

Dual problem

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C \,\forall\, i,$$

where C is the tolerance for errors.

Nonlinear case

Observations

- Some data is not linearly separable in its original space but is in another projected space (Fig. 3) $\rightarrow$ we can use SVM in the latter.
- Finding a suitable projected space is difficult. Sometimes, the projected space is of infinite dimension, e.g.,

$$x \mapsto g_x(u) = \begin{cases} u \text{ if } u \leq x \\ x \text{ otherwise} \end{cases}$$

with $x \in [0, 1]$ and $\langle g_x, g_y \rangle = \int_0^1 g'_x(u) g'_y(u) du$.

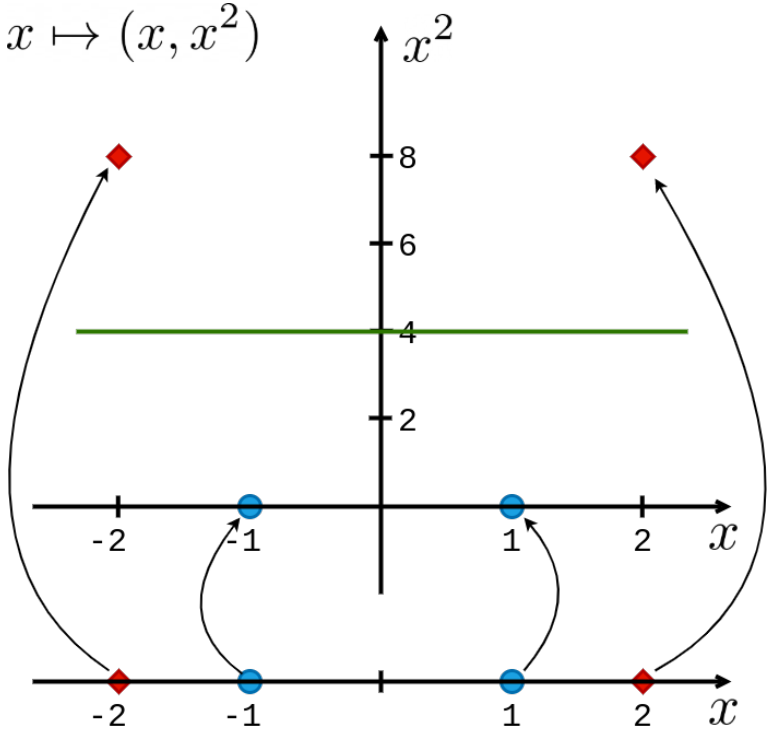


Figure: Appropriate projections can transform non-linearly separable data into linearly separable data.

Kernels

Kernel trick

- Knowing the inner product between projections $K(x, y) = \langle g(x), g(y) \rangle$ is enough for SVM.
- Functions $K : X \times X \to \mathbb{R}$ are called kernels.
- Kernel trick: Instead of finding suitable g, we find suitable K.
- Dual problem becomes

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C \, \forall \, i,$$

- Not all kernels can be used for SVM, only **positive definite kernels**.

Kernels

Positive definite kernels

A kernel $K : X \times X \to \mathbb{R}$ is positive definite if

$$\sum_{i,j}^{n} \alpha_i \alpha_j K(x_i, x_j) \geq 0 \tag{4}$$

for any choice of $n \in \mathbb{N}$, $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{R}$ and $x_1, x_2, \ldots, x_n \in X$.

Popular kernels

- Polynomial kernels: $K(x, y) = (x^T y)^n$.
- Exponential kernels: $K(x, y) = \exp x^T y$.
- Composite kernels: $K_1 + K_2$, $K_1 K_2$, cK_1, $\exp K_1$, ...
- RBF: $K(x, y) = \exp -\frac{\|x - y\|^2}{2\sigma^2}$.

Kernels

Kernel advantages

- Adding non-linearity to SVM.
- Easy to deal with complex data: ADN sequences, text, images, proteins, ...

SVM in Python

sklearn.svm.SVC

class `sklearn.svm.` **SVC**(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=- 1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*) [source]

sklearn.svm.LinearSVC

class `sklearn.svm.` **LinearSVC**(*penalty='l2'*, *loss='squared_hinge'*, *, *dual=True*, *tol=0.0001*, *C=1.0*, *multi_class='ovr'*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *verbose=0*, *random_state=None*, *max_iter=1000*) [source]

sklearn.svm.LinearSVR

class `sklearn.svm.` **LinearSVR**(*, *epsilon=0.0*, *tol=0.0001*, *C=1.0*, *loss='epsilon_insensitive'*, *fit_intercept=True*, *intercept_scaling=1.0*, *dual=True*, *verbose=0*, *random_state=None*, *max_iter=1000*) [source]

What to remember

- SVM is a linear model with Hinge loss.
- SVM is a maximum margin classifier.
- Primal vs. dual.
- Kernel tricks.

Exercises

- Derive dual problems from primal problems.
- Find the SVM coefficients (w and w_0) for the linearly separable toy dataset with LinearSVC. Choose the parameters carefully. Is your result identical to the numbers in the lecture?