



CSC380: Principles of Data Science

Nonlinear Models 1

- Basis Functions
- Support Vector Machine
- Neural Networks

Credit:

- Jason Pacheco,
- Kwang-Sung Jun,
- Chicheng Zhang
- Xincheng yu

1

1

Announcements

2

- Fill out SCS (<https://scsonline.oia.arizona.edu/>) – if 80% responses, will add 5 points to the homework with lowest grade.
- HW7 due next Wednesday, Nov 22 by 11:59pm.
- The final project will be out next Tuesday, Nov 21. The due date is Friday, Dec 8.
- Lecture on next Tuesday will be on [Zoom](#).

2

Review: Logistic Regression

3

Model:

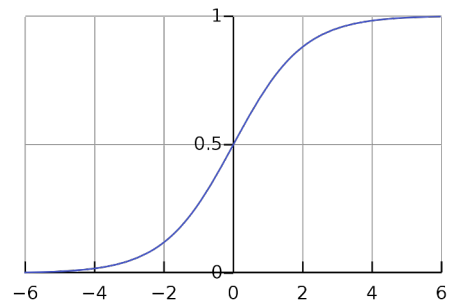
$$y \sim \text{Bernoulli}(p = \sigma(w^\top x))$$

Train: compute the MLE \hat{w}

Test: Given test point x^* compute

$$y^* = \arg \max_{v \in \{-1, 1\}} p(y = v \mid x^*; \hat{w})$$

- Equivalent to $y^* = \mathbf{I}\{\hat{w}^\top x^* \geq 0\}$



$$\sigma(w^\top x) = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}$$

3

Outline

4

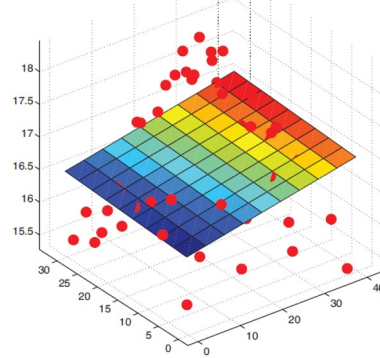
- Basis Functions
- Support Vector Machine
- Neural Networks

4

Linear Models

5

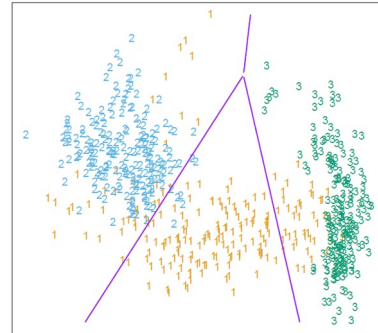
[Image: Murphy, K. (2012)]



Linear Regression Fit a *linear function* to the data,

$$y = w^T x$$

[Image: Hastie et al. (2001)]



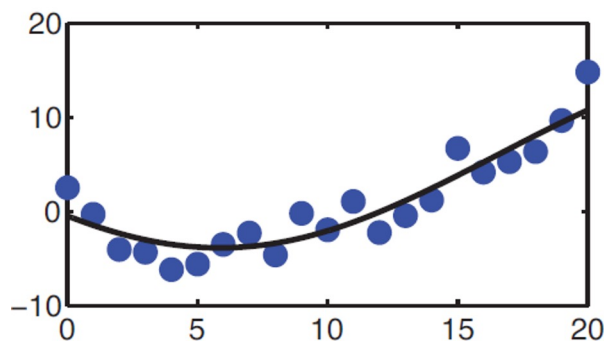
Logistic Regression Learn a decision boundary that is *linear in the data*,

$$y = \mathbf{I}\{w^T x \geq 0\}$$

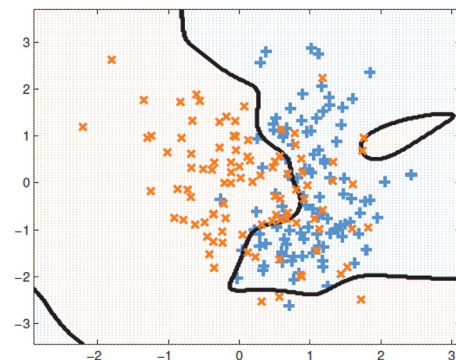
5

Nonlinear Data

6



What if our data are *not* well-described by a linear function?



What if classes cannot be well-distinguished by a linear function?

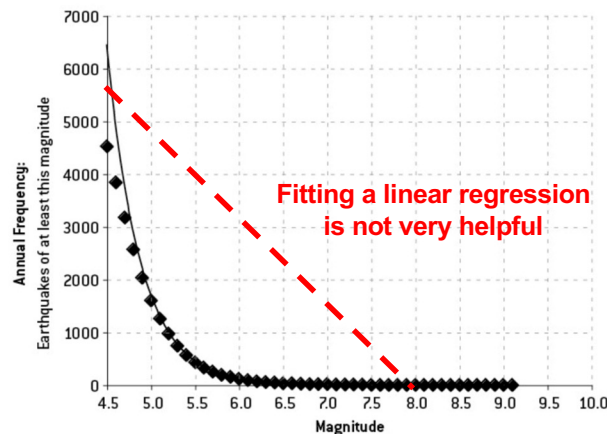
6

Example: Earthquake Prediction

7

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3A: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012



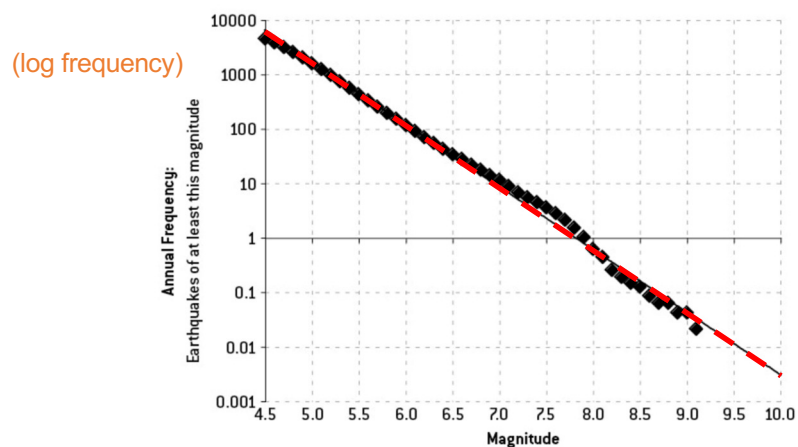
7

Example: Earthquake Prediction

8

Suppose that we want to predict the number of earthquakes that occur of a certain magnitude. Our data are given by,

FIGURE 5-3B: WORLDWIDE EARTHQUAKE FREQUENCIES, JANUARY 1964–MARCH 2012,
LOGARITHMIC SCALE



But plotting outputs on a logarithmic scale reveals a strong linear relationship...

$$\text{it's like } y = e^{-ax+b}$$

8

Beyond linearity: Transformation in x

9

- Recall: for 1d problem, we embedded the feature: $x' = (x, 1) \in \mathbb{R}^2$ so we can encode the intercept term.

$$\phi_0(x) = 1 \quad \phi_1(x) = x \quad y = \mathbf{w}^\top \Phi_{\text{lin}}(x) = \phi_0(x)w_0 + \phi_1(x)w_1 = w_0 + w_1x$$

9

Beyond linearity: Transformation in x

10

- Recall: for 1d problem, we embedded the feature: $x' = (x, 1) \in \mathbb{R}^2$ so we can encode the intercept term.

$$\phi_0(x) = 1 \quad \phi_1(x) = x \quad y = \mathbf{w}^\top \Phi_{\text{lin}}(x) = \phi_0(x)w_0 + \phi_1(x)w_1 = w_0 + w_1x$$

- Actually, the embedding trick is stronger.
 - $(x^2, x, 1)$: 2nd order polynomial with respect to x
 - $(x^d, x^{d-1}, \dots, 1)$: d-th order polynomial (= degree d)

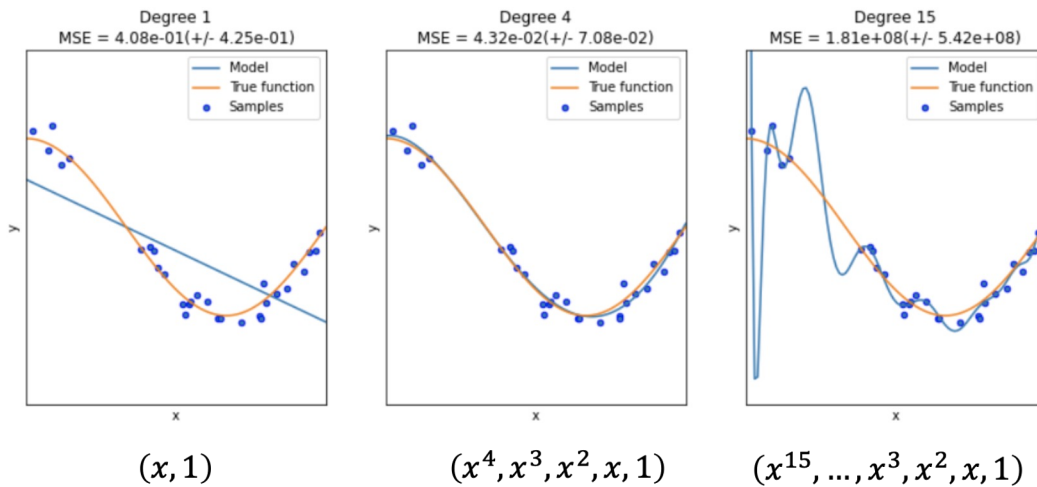
$$\phi_0(x) = 1 \quad \phi_1(x) = x \quad \phi_2(x) = x^2$$

$$y = \mathbf{w}^\top \Phi_{\text{lin}}(x) = \phi_0(x)w_0 + \phi_1(x)w_1 + \phi_2(x)w_2 = w_0 + w_1x + w_2x^2$$

10

Feature embedding trick

11



higher-order polynomial = higher complexity = prone to overfitting!

from <https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning>

11

Basis Functions

12

- A **basis function** can be any function of the input features **X**
- Define a set of B basis functions $\phi_1(x), \dots, \phi_B(x)$
- Fit a linear regression model in terms of basis functions,

$$y = \sum_{b=1}^B w_b \phi_b(x) = w^T \phi(x)$$

notation:
 $\phi(x) := [\phi_1(x), \dots, \phi_B(x)]$

- The model is *linear* in the transformed basis/induced features $\phi(x)$.
- The model is *nonlinear* in the data **X**

12

Linear Regression

13

Recall the ordinary least squares solution is given by,

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1D} \\ 1 & x_{21} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \dots & x_{mD} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \quad w^{\text{OLS}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Design Matrix
(each training input on a column)

Vector of
Training labels

Can similarly solve in terms of basis functions,

$$\Phi = \begin{pmatrix} 1 & \phi_1(x_1) & \dots & \phi_B(x_1) \\ 1 & \phi_1(x_2) & \dots & \phi_B(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_m) & \dots & \phi_B(x_m) \end{pmatrix} \quad w^{\text{OLS}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$$

13

sklearn.preprocessing.PolynomialFeatures

14

degree : int or tuple (min_degree, max_degree), default=2

If a single int is given, it specifies the maximal degree of the polynomial features. If a tuple (min_degree, max_degree) is passed, then min_degree is the minimum and max_degree is the maximum polynomial degree of the generated features. Note that min_degree=0 and min_degree=1 are equivalent as outputting the degree zero term is determined by include_bias.

interaction_only : bool, default=False

If True, only interaction features are produced: features that are products of at most degree distinct input features, i.e. terms with power of 2 or higher of the same input feature are excluded:

- included: $x[0]$, $x[1]$, $x[0] * x[1]$, etc.
- excluded: $x[0] ** 2$, $x[0] ** 2 * x[1]$, etc.

include_bias : bool, default=True

If True (default), then include a bias column, the feature in which all polynomial powers are zero (i.e. a column of ones - acts as an intercept term in a linear model).

order : {'C', 'F'}, default='C'

Order of output array in the dense case. 'F' order is faster to compute, but may slow down subsequent estimators.

14

Example: Polynomial Basis Functions

15

Create three two-dimensional data points [0,1], [2,3], [4,5]:

```
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
```

Compute quadratic features $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$,

```
>>> poly = PolynomialFeatures(degree=2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
```

These are now our new data and ready to fit a model...

15

Example: Polynomial Basis Functions

16

Create a 3-rd order polynomial (cubic) function,

```
f = lambda x: (x-1)*(x-2)*(x-3)
import numpy.random as ra
ra.seed(20)
train_x = np.arange(5)
train_y = f(train_x) + 1*ra.randn(len(train_x))
train_y
```

✓ 0.3s

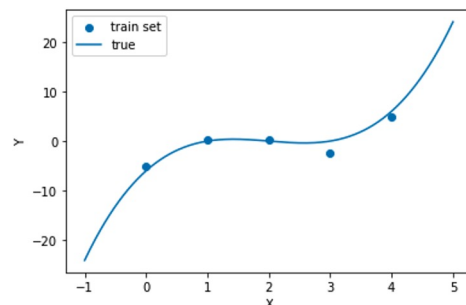
array([-5.11610689, 0.19586502, 0.35753652, -2.34326191, 4.91516741])

Plot train set and the actual function

```
test_x = np.linspace(-1,5,400)
from matplotlib import pyplot as plt
plt.scatter(train_x, train_y)

plt.plot(test_x, f(test_x))
plt.legend(['train set', 'true'])
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

✓ 0.4s



16

Example: Polynomial Basis Functions

17

Create cubic features $(1, x, x^2, x^3)$

```
poly = PolynomialFeatures(degree=3)
train_xx = poly.fit_transform(train_x[:, np.newaxis])
train_xx
```

✓ 0.4s turns train_x (length 5 array) into a matrix (5 by 4 matrix)

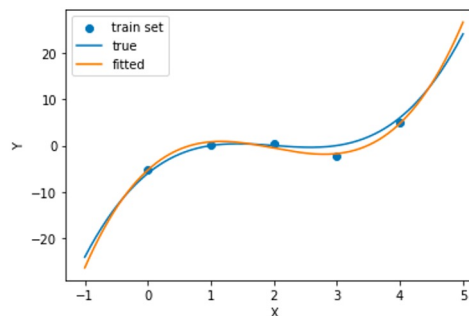
```
array([[ 1.,  0.,  0.,  0.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  2.,  4.,  8.],
       [ 1.,  3.,  9., 27.],
       [ 1.,  4., 16., 64.]])
```

Perform linear regression; plot it

```
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(train_xx, train_y)
test_x = np.linspace(-1, 5, 400)
test_xx = poly.fit_transform(test_x[:, np.newaxis])
pred_y = model.predict(test_xx)

plt.scatter(train_x, train_y)
plt.plot(test_x, f(test_x))
plt.plot(test_x, pred_y)
plt.legend(['train set', 'true', 'fitted'])
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

✓ 0.2s



17

Data Preprocessing

18

- Generally the first step in data science involves *preprocessing* or transforming data in some way
 - Filling in missing values (imputation)
 - Centering / normalizing / standardizing
 - Etc.
- We then fit our models to this preprocessed data
- One way to view preprocessing is simply as computing some basis function $\phi(x)$ nothing more

18

Basis Functions

19

PROs

- More flexible modeling that is nonlinear in the original data
- Increases model expressivity

CONs

- Typically requires **more parameters** to be learned
- More sensitive to **overfitting** training data (due to expressivity)
- Requires more **regularization** to avoid overfitting
- Need to find *good* basis functions (feature engineering)

19

Outline

20

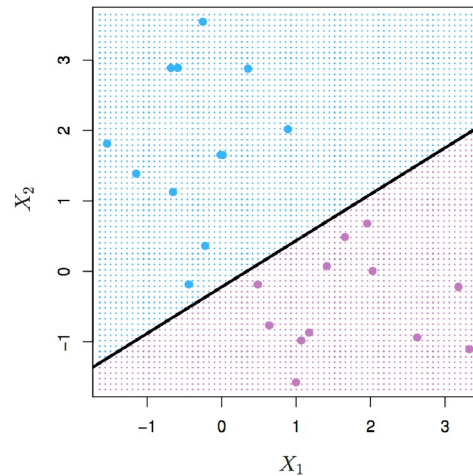
- Basis Functions
- Support Vector Machine
- Neural Networks

20

Linear Decision Boundary

21

*Forget about the 'regression' point of view for now..
At the end of the day, we just want a line that separates the two classes well.*



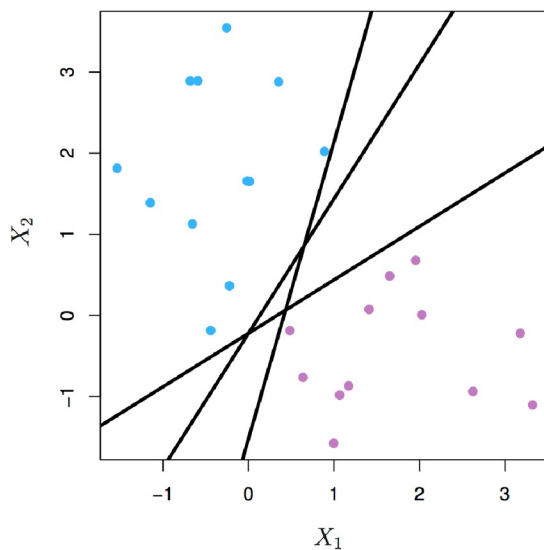
[Source: <http://www-bcf.usc.edu/~gareth/ISL/>]

21

Linear Decision Boundary

22

Note: Any boundary that separates classes is equivalently good on training data

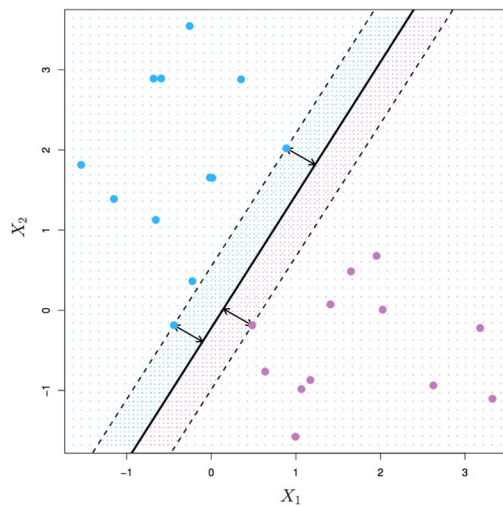


Q: but if you have to choose one, which one will you choose?

22

Classifier Margin

23



The **margin** measures minimum distance between each class and the decision boundary

Observation Decision boundaries with larger margins are more likely to generalize to unseen data

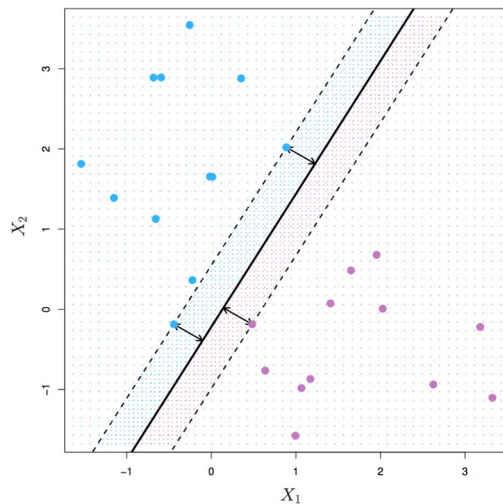
Idea Learn the classifier with the largest margin that still separates the data...

...we call this a **max-margin classifier**

23

Max-Margin Classifier (Linear Separable Case)

24



For now, let's focus on the case where the data is **linearly separable**

(Otherwise, there is no margin to talk about!)

24

Hyperplane

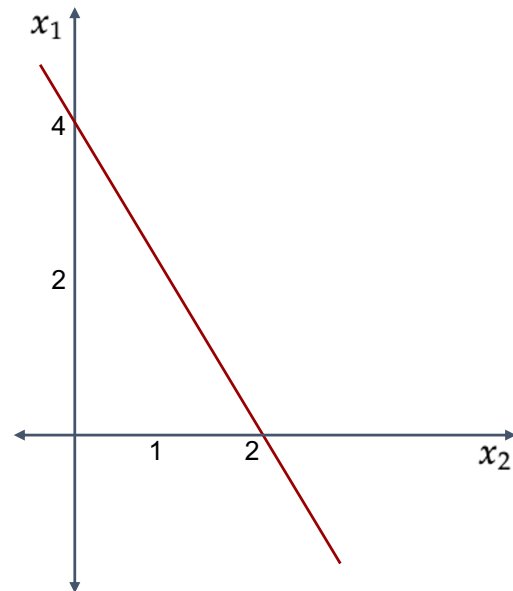
25

A linear discriminant function in D dimensions is given by a hyperplane, defined as follows:

$$\begin{aligned} h(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b \\ &= w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b \end{aligned}$$

For points that lie on the hyperplane, we have:

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0$$



25

Separating Hyperplane

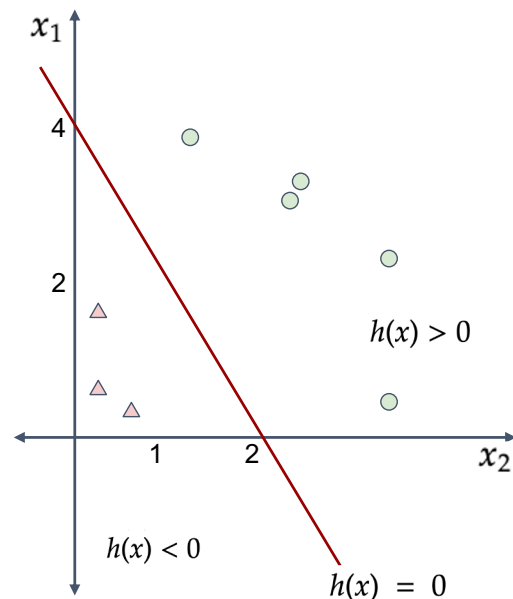
26

A hyperplane $h(\mathbf{x})$ splits the original d-dimensional space into two half-spaces. If the input dataset is linearly separable:

$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0 \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases}$$

Example:

$$h(x) = x_1 + 2x_2 - 4$$



26

Separating Hyperplane: weight vector

27

Let \mathbf{a}_1 and \mathbf{a}_2 be two arbitrary points that lie on the hyperplane, we have:

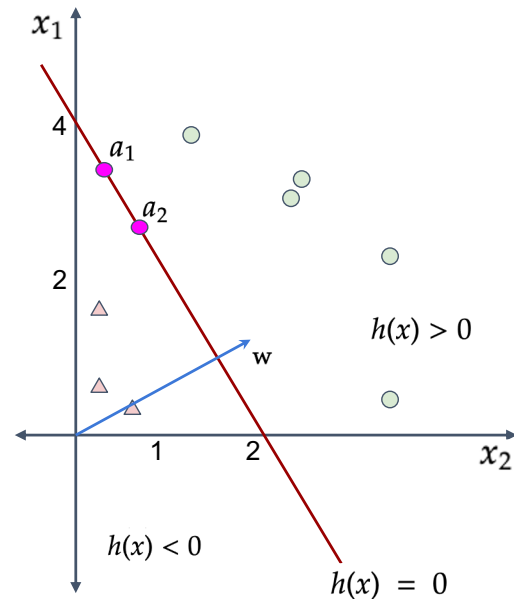
$$h(\mathbf{a}_1) = \mathbf{w}^T \mathbf{a}_1 + b = 0$$

$$h(\mathbf{a}_2) = \mathbf{w}^T \mathbf{a}_2 + b = 0$$

Subtracting one from the other:

$$\mathbf{w}^T (\mathbf{a}_1 - \mathbf{a}_2) = 0$$

The weight vector \mathbf{w} is orthogonal to the hyperplane.



27

Distance of a Point to the Hyperplane

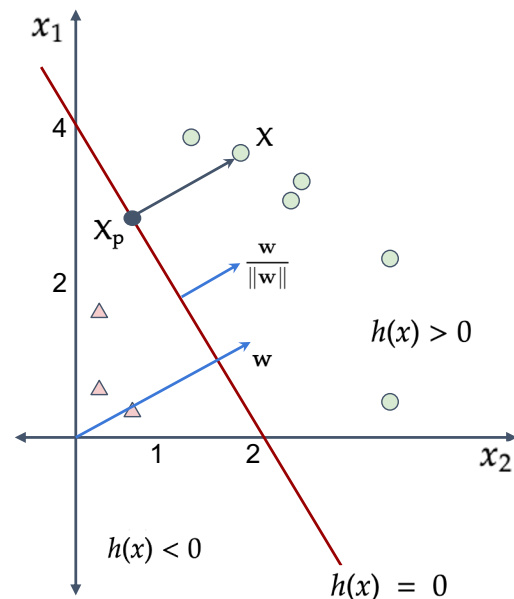
28

Consider a point X not on the hyperplane. Let X_p be the projection of X on the hyperplane.

Let r be the steps need to walk from X_p to X .

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Q: how many steps/direct distance do we need to walk?



28

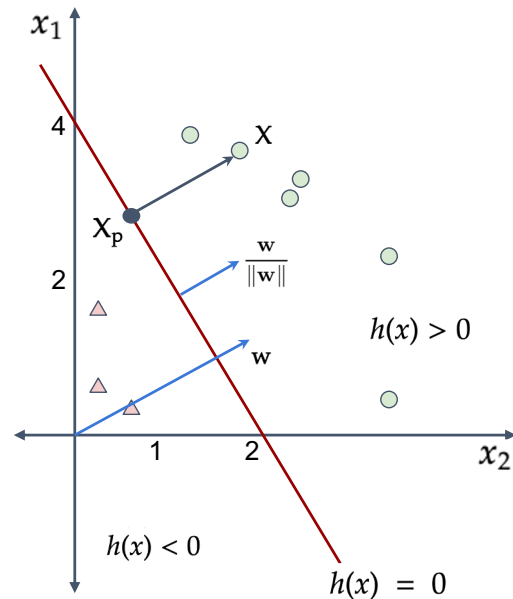
Distance of a Point to the Hyperplane

29

Consider a point X not on the hyperplane. Let X_p be the projection of X on the hyperplane.

Let r be the steps need to walk from X_p to X .

$$\begin{aligned}
 h(\mathbf{x}) &= h\left(\mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}\right) \\
 &= \mathbf{w}^T \left(\mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b \\
 &= \underbrace{\mathbf{w}^T \mathbf{x}_p + b}_{h(\mathbf{x}_p)} + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \\
 &= \underbrace{h(\mathbf{x}_p)}_0 + r \|\mathbf{w}\| \\
 &= r \|\mathbf{w}\| \qquad r = \frac{h(\mathbf{x})}{\|\mathbf{w}\|}
 \end{aligned}$$



29

Distance of a Point to the Hyperplane

30

Q: What is the direct distance from origin ($\mathbf{x}=0$) to the hyperplane?

$$r = \frac{h(\mathbf{x})}{\|\mathbf{w}\|} \quad r = \frac{h(\mathbf{0})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T \mathbf{0} + b}{\|\mathbf{w}\|} = \frac{b}{\|\mathbf{w}\|}$$

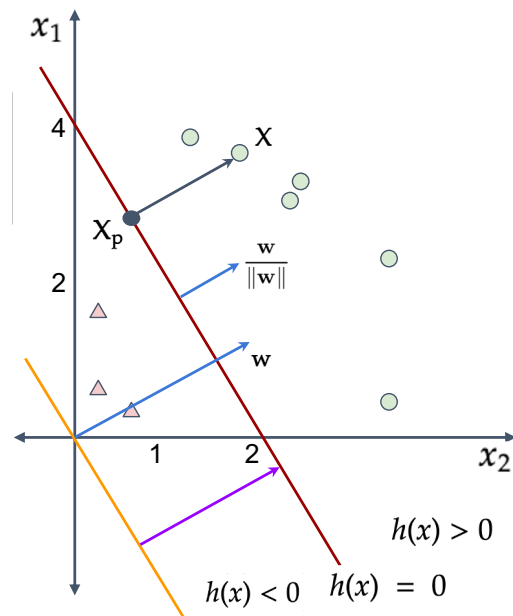
Example:

$$h(\mathbf{x}) = x_1 + 2x_2 - 4$$

$$\mathbf{w}^T \mathbf{x} + b = (1 \ 2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 4$$

$$\frac{b}{\|\mathbf{w}\|} = -\frac{4}{\sqrt{5}}$$

Q: how to deal with negative distance?



30

Distance of a Point to the Hyperplane

31

Q: How to deal with negative distance?

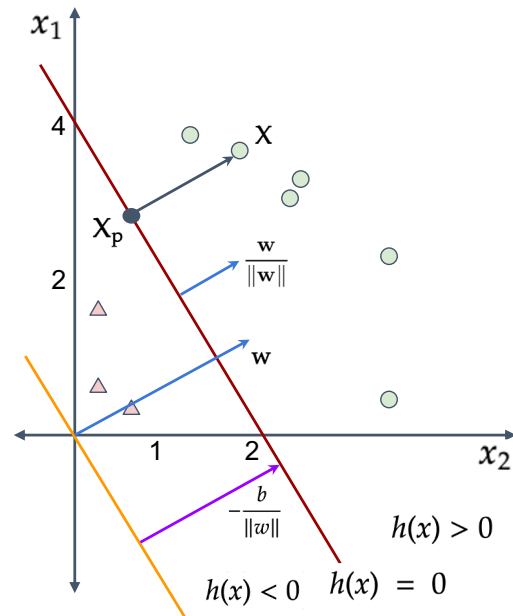
$$r = \frac{h(\mathbf{x})}{\|\mathbf{w}\|}$$

$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0 \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases}$$

$$\delta = y r = \frac{y h(\mathbf{x})}{\|\mathbf{w}\|}$$

Example (when point is the origin):

$$(-1) \cdot \frac{b}{\|w\|} = \frac{4}{\sqrt{5}}$$



31

Margin and Support Vectors

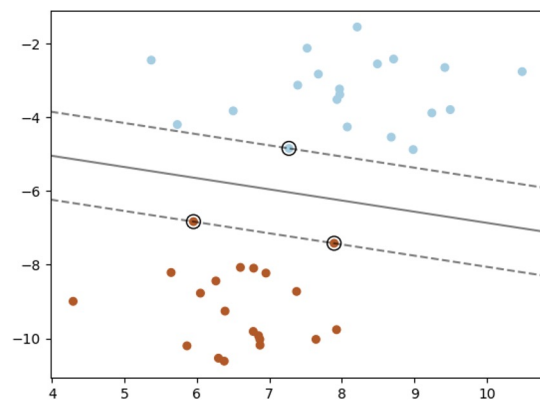
32

Over all the n points, the **margin** of the linear classifier is the minimum distance of a point from the separating hyperplane:

$$\delta^* = \min_{\mathbf{x}_i} \left\{ \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|} \right\}$$

All the points that achieve this minimum distance are called **support vectors**.

$$\delta^* = \frac{y^*(\mathbf{w}^T \mathbf{x}^* + b)}{\|\mathbf{w}\|}$$



32

Max-Margin Classifier (Linear Separable Case) 33

For training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, a classifier $f(x) = w^T x + b$ with 0 train error will satisfy

$$y^{(i)} f(x^{(i)}) = y^{(i)} (w^T x^{(i)} + b) > 0$$

↓ negative margin when misclassifying it!

The distance for $(x^{(i)}, y^{(i)})$ to separating hyperplane

$$\frac{y^{(i)} (w^T x^{(i)} + b)}{\|w\|}$$

The margin of a classifier $f(x)$ is

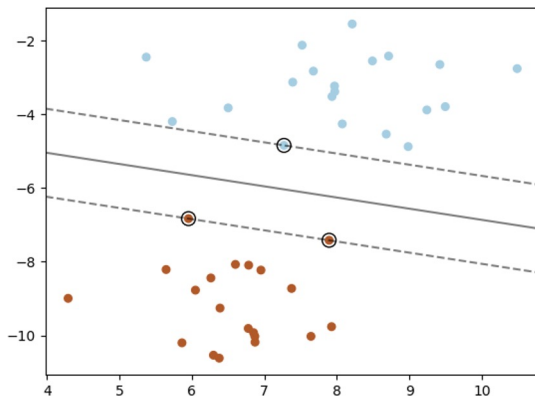
$$\min_i \frac{y^{(i)} (w^T x^{(i)} + b)}{\|w\|}$$

Find f that maximize margin

$$\arg \max_{w,b} \min_i \frac{y^{(i)} (w^T x^{(i)} + b)}{\|w\|}$$

33

Max-Margin Classifier (Linear Separable Case) 34



Find the parameters (w,b) that **maximize** the **smallest margin** over all the training data

$$\arg \max_{w,b} \underbrace{\min_i \frac{y^{(i)} (w^T x^{(i)} + b)}{\|w\|}}_{\text{Minimum margin over all training data}}$$

Maximize the minimum margin

34

Canonical Hyperplane

35

Issue: infinite equivalent hyperplanes result in infinite solutions:

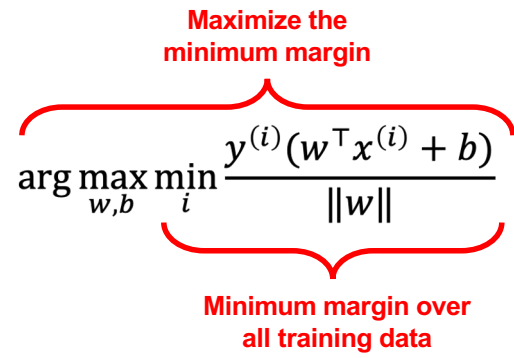
- Multiplying on both sides by some scalars yields an equivalent hyperplane

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

Example of equivalent hyperplanes:

$$h(x) = x_1 + 2x_2 - 4$$

$$h(x) = 2x_1 + 4x_2 - 8$$



The diagram shows the optimization objective function for finding the canonical hyperplane. The function is $\arg \max_{w,b} \min_i \frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|}$. A red bracket above the fraction is labeled "Maximize the minimum margin". A red bracket below the fraction is labeled "Minimum margin over all training data".

$$\arg \max_{w,b} \min_i \frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|}$$