



Computer
Science

CSC380: Principles of Data Science

Nonlinear Models 2

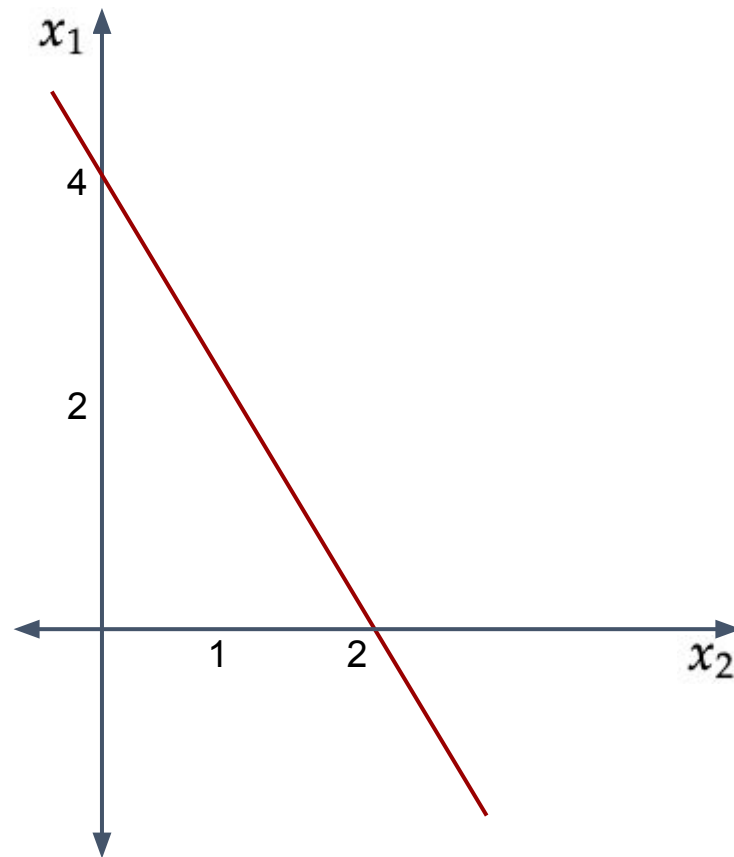
Xinchen Yu

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:

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



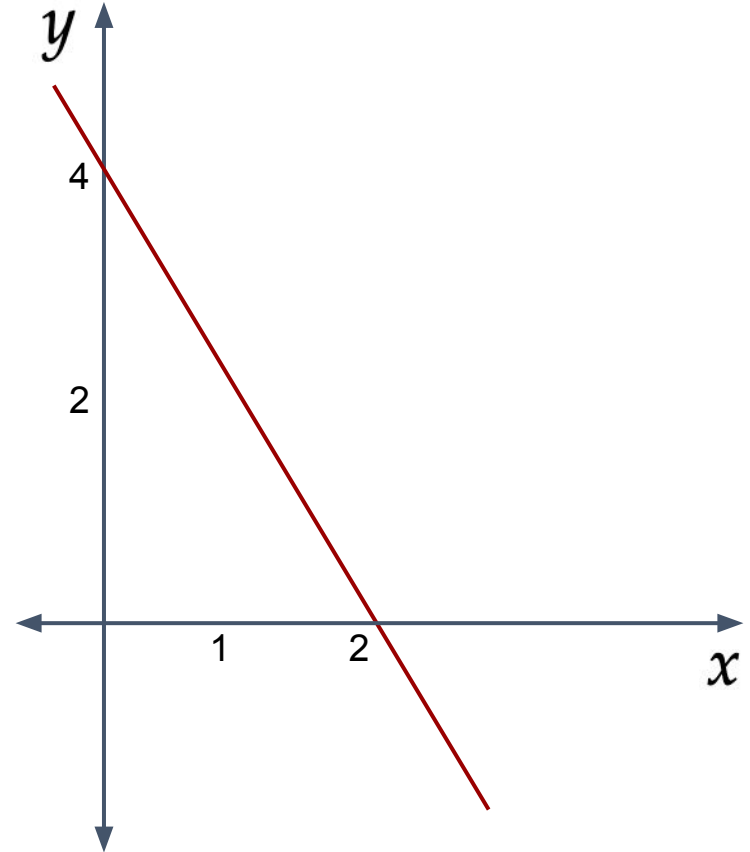
$$y = -2x + 4$$

$$y + 2x - 4 = 0$$

$$y \rightarrow x_1$$

$$x \rightarrow x_2$$

$$x_1 + 2x_2 - 4 = 0$$

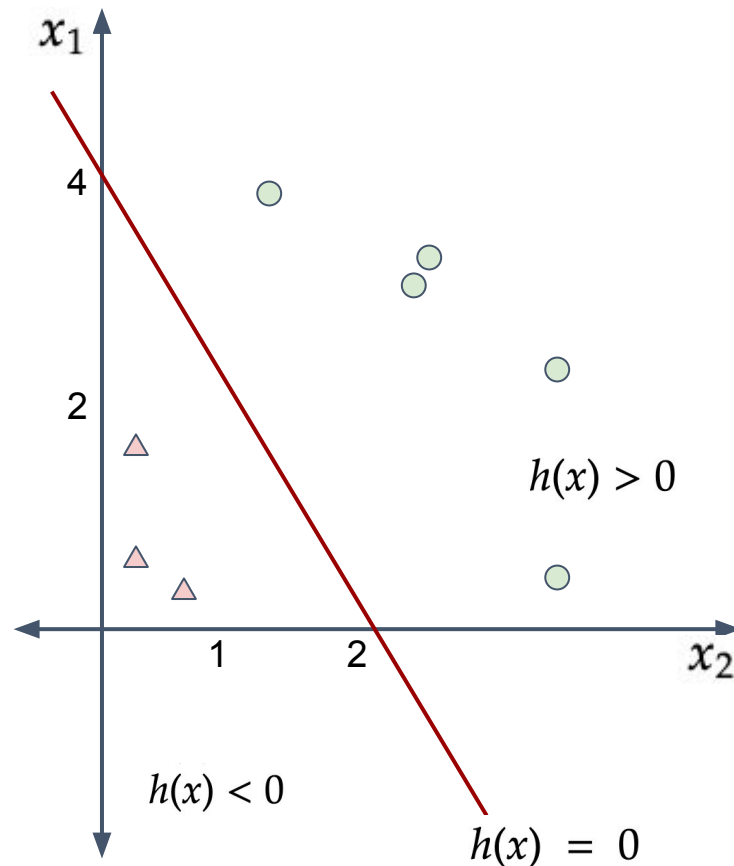


A hyperplane $h(x)$ splits the original d -dimensional space into two half-spaces. If the input dataset is linearly separable:

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

Example:

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



$$y = -2x + 4$$

$$\circ y > -2x + 4 \quad y + 2x - 4 > 0$$

$$\triangle y < -2x + 4 \quad y + 2x - 4 < 0$$

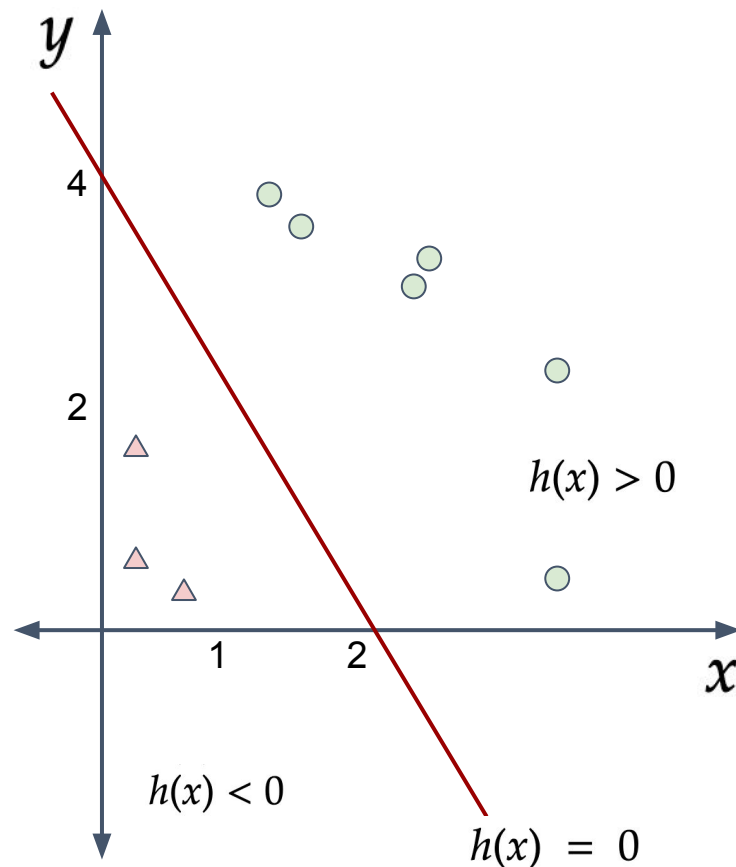
$$y \rightarrow x_1$$

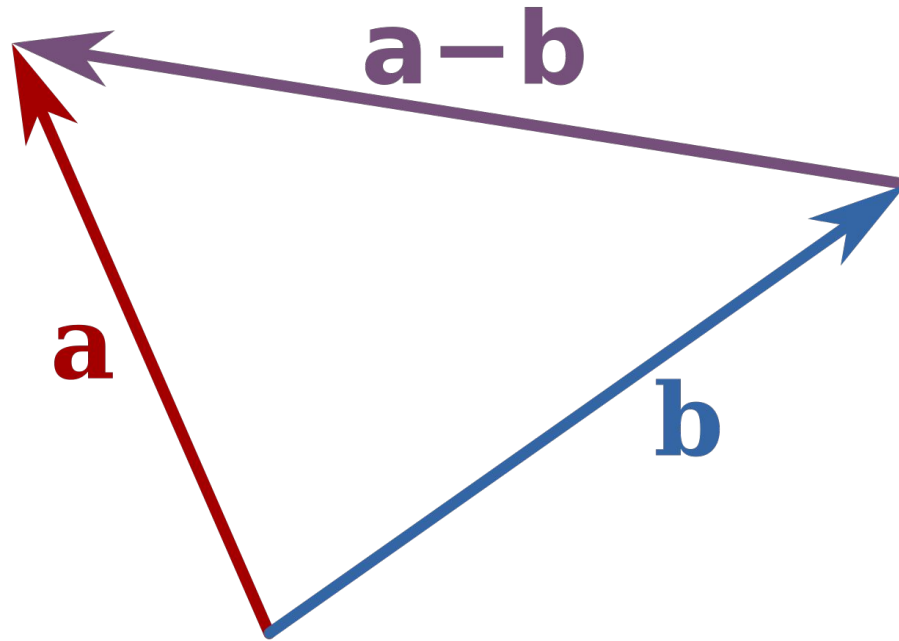
$$x \rightarrow x_2$$

$$\circ x_1 + 2x_2 - 4 > 0$$

$$\triangle x_1 + 2x_2 - 4 < 0$$

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





Separating Hyperplane: weight vector

7

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

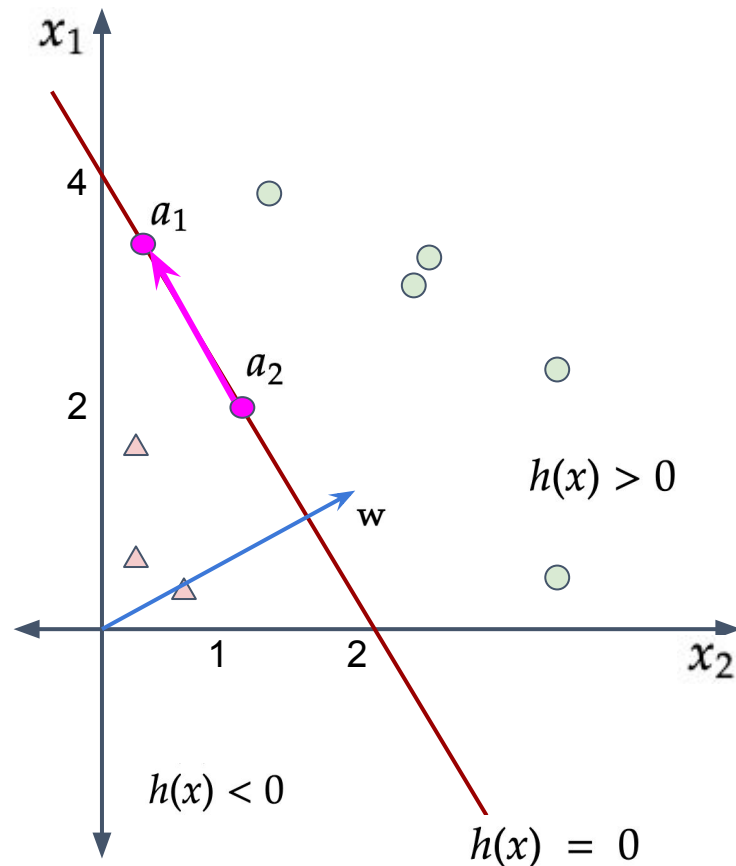
$$h(\mathbf{a}_1) = \mathbf{w}^T \mathbf{a}_1 + b = 0$$

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

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.



Distance of a Point to the Hyperplane

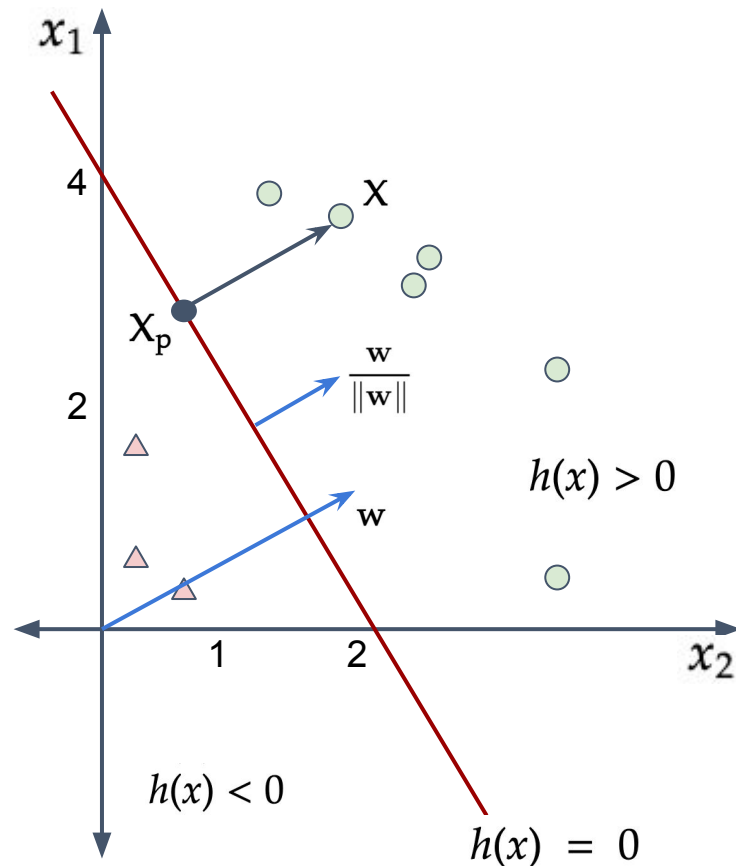
8

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?



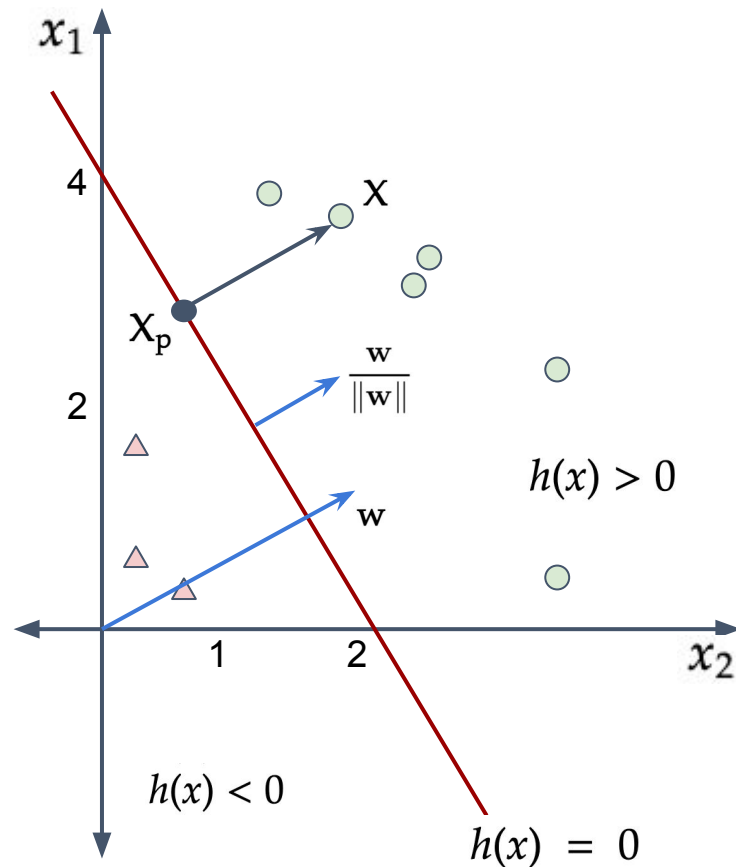
Distance of a Point to the Hyperplane

9

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 .

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



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

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

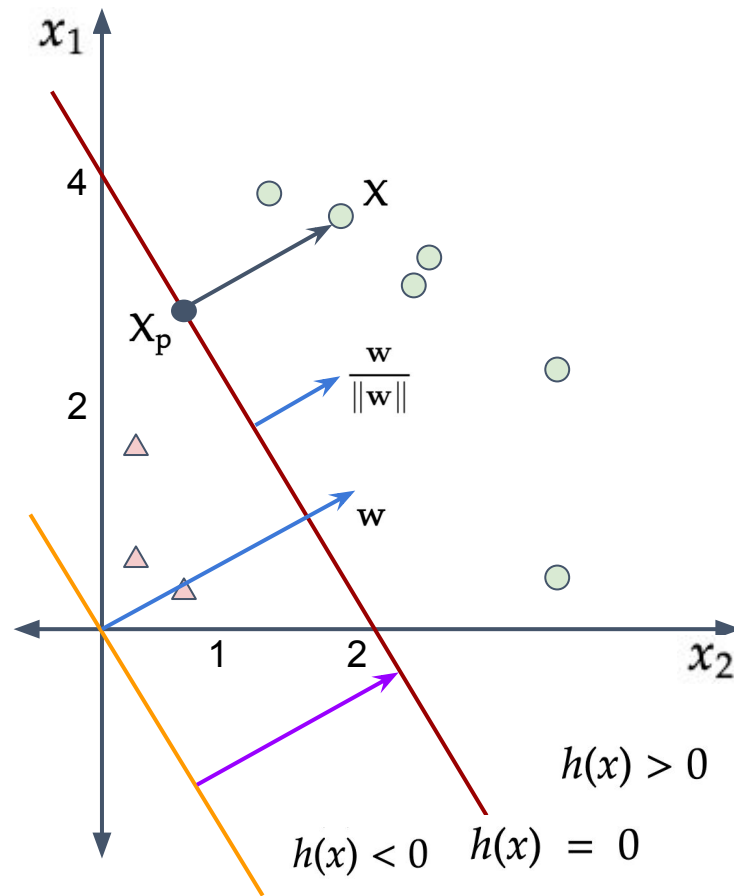
Example:

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

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

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

Q: how to deal with negative distance?



Q: How to deal with negative distance?

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

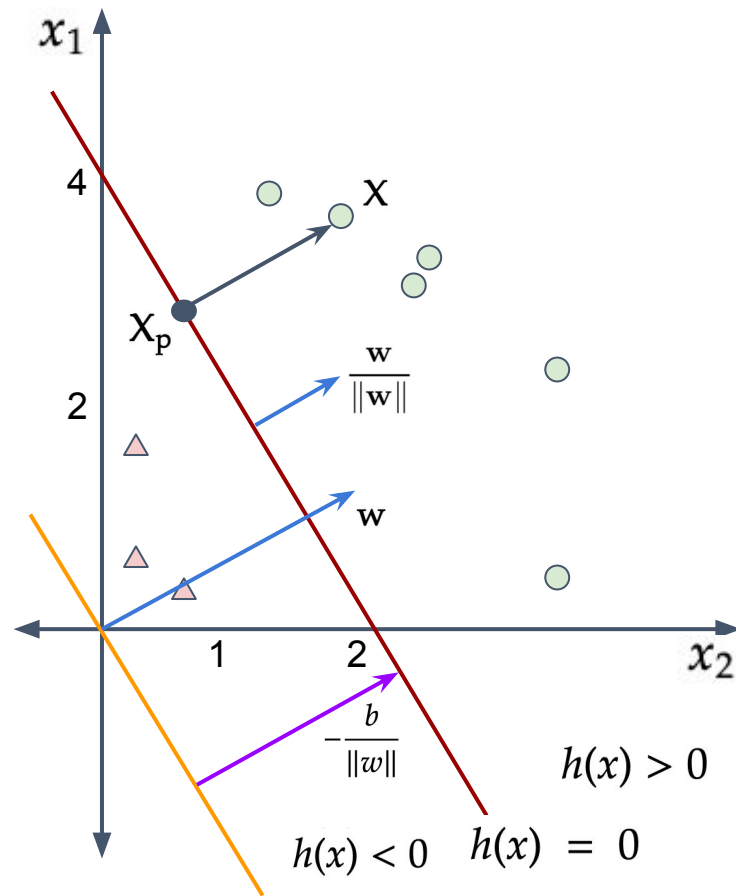
$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0 \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases}$$

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

Q: why not using absolute value?

Example (when point is the origin):

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



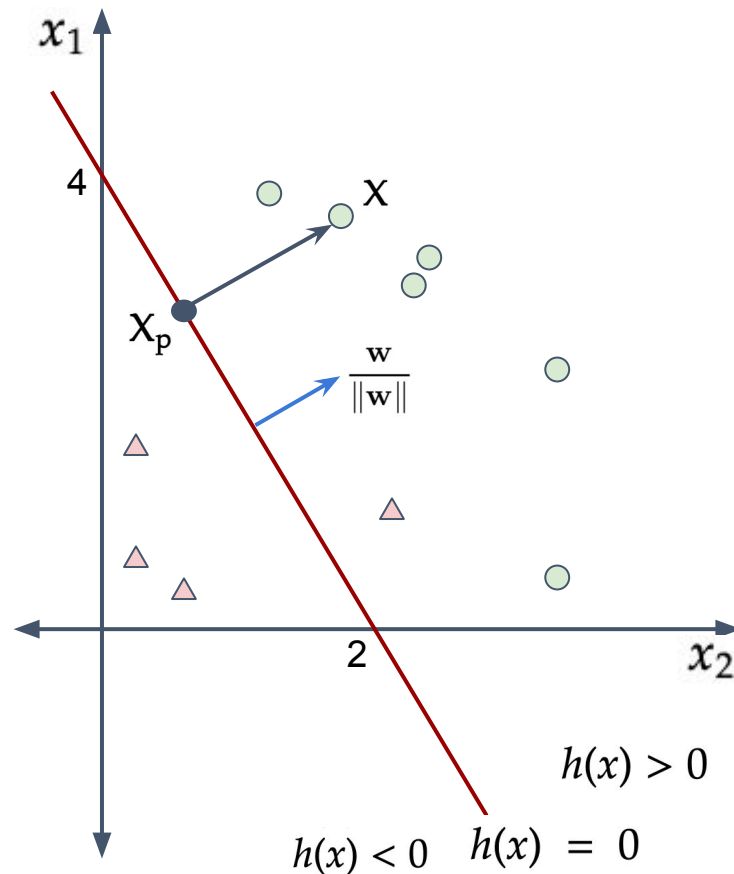
Q: How to deal with negative distance?

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

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

Q: why not using absolute value of $h(x)$?

We not only care about the distance of a point to the hyperplane, but also care if the point is correctly labeled: using absolute value only gets the distance (because always positive).

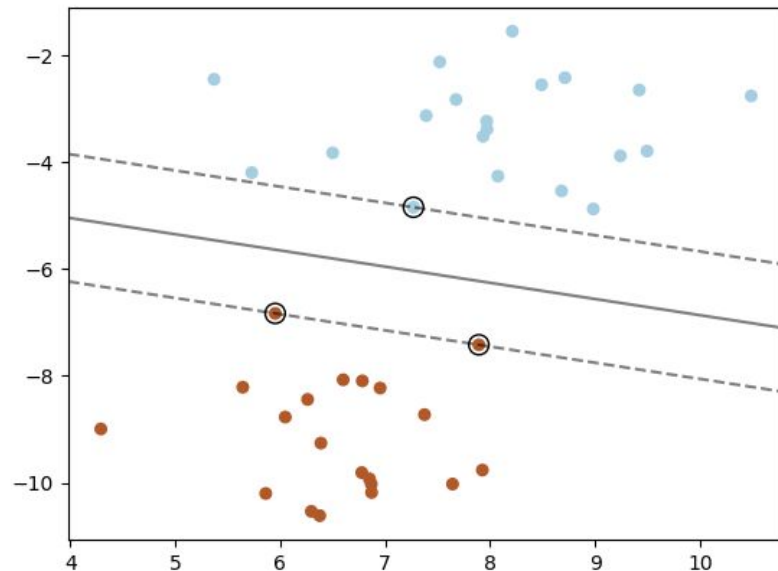


Over all the n points, the **margin** of the linear classifier is the minimum distance of a point from the separating hyperplane:

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

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

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



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

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

↓ negative margin when misclassifying it!

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

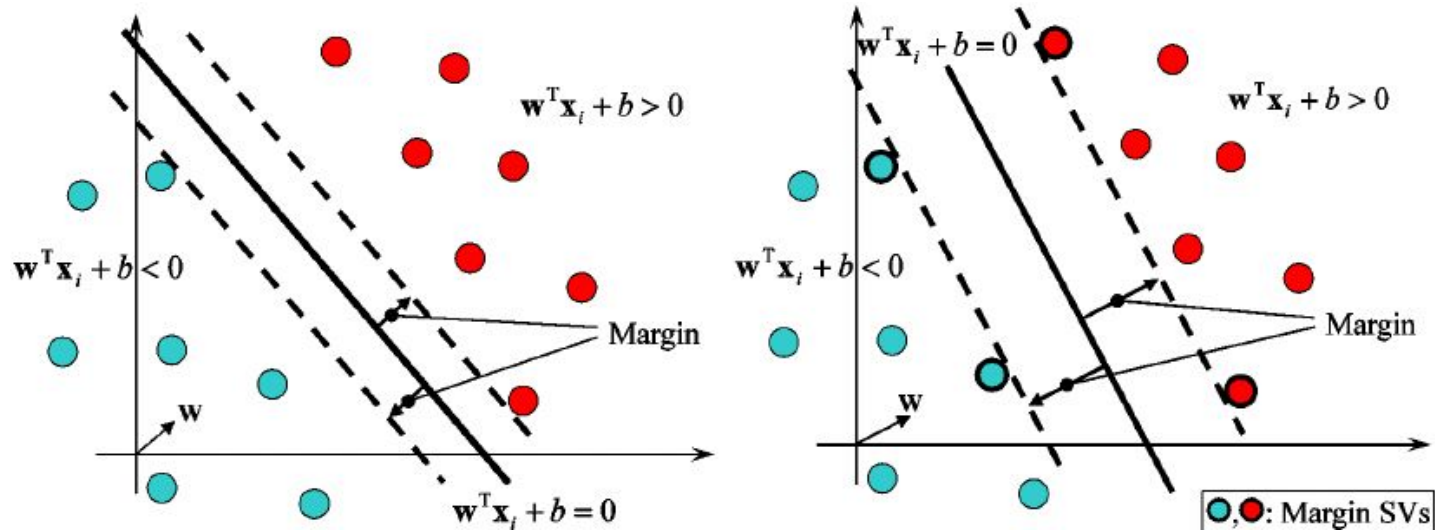
$$\frac{y^{(i)} (w^\top x^{(i)} + b)}{\|w\|}$$

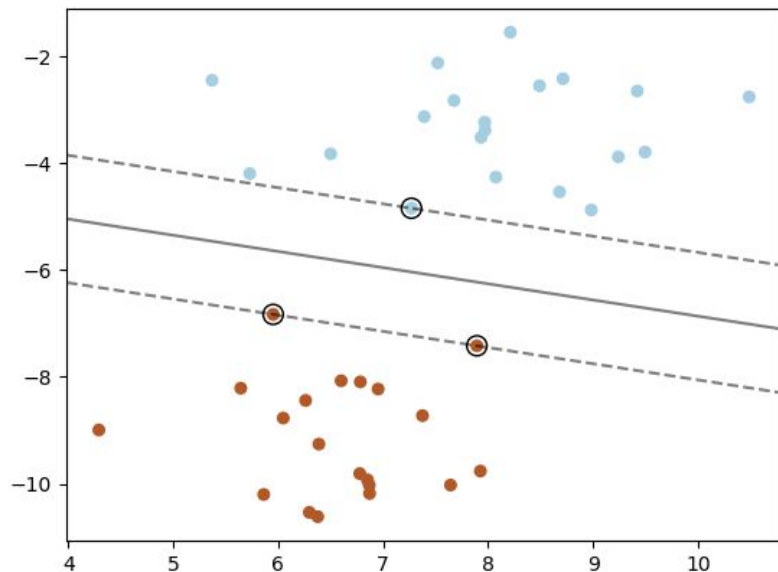
The margin of a classifier $f(x)$ is

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

Find f that maximize margin

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





Maximize the minimum margin

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

Minimum margin over all training data

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

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

Issue: infinite equivalent hyperplanes result in infinite solutions:

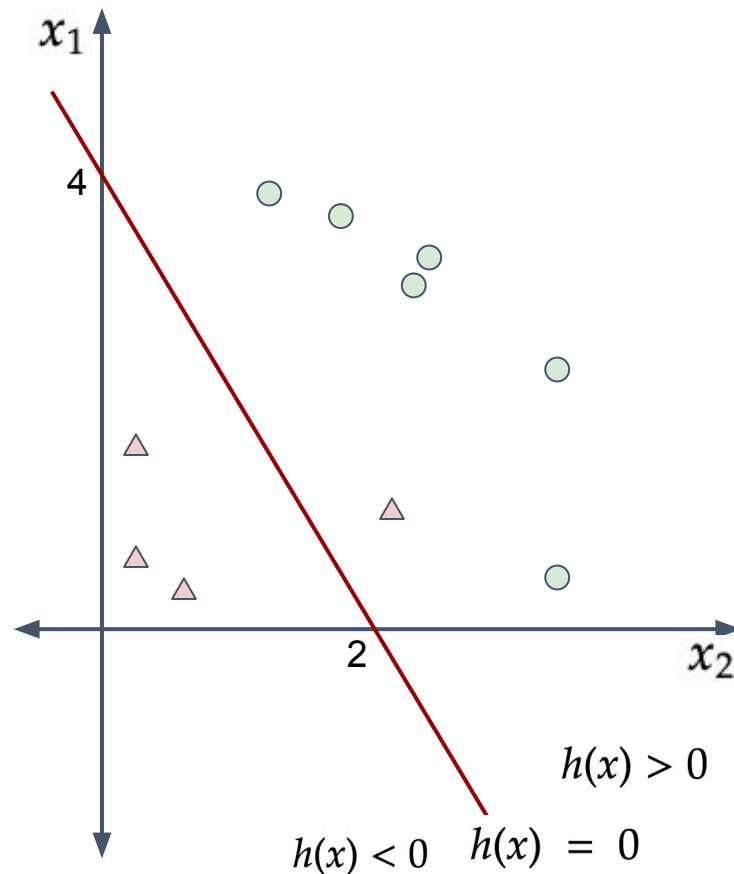
- Multiplying on both sides by some scalars yields an equivalent hyperplane

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

Example of equivalent hyperplanes:

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

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




Way to solve this issue:

- Choose the scalar s such that the absolute distance of a **support vector** from the hyperplane is 1.

$$sy^*(\mathbf{w}^T \mathbf{x}^* + b) = 1$$

$$s = \frac{1}{y^*(\mathbf{w}^T \mathbf{x}^* + b)} = \frac{1}{y^* h(\mathbf{x}^*)}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \text{ for all points } \mathbf{x}_i \in \mathbf{D}$$

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


Margin: $\delta^* = \frac{1}{\|\mathbf{w}\|}$

Max margin: $h^* = \arg \max_h \{\delta_h^*\} = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\}$

$$h'(\mathbf{x}) = \begin{pmatrix} 5 \\ 2 \end{pmatrix}^T \mathbf{x} - 20 = 0$$

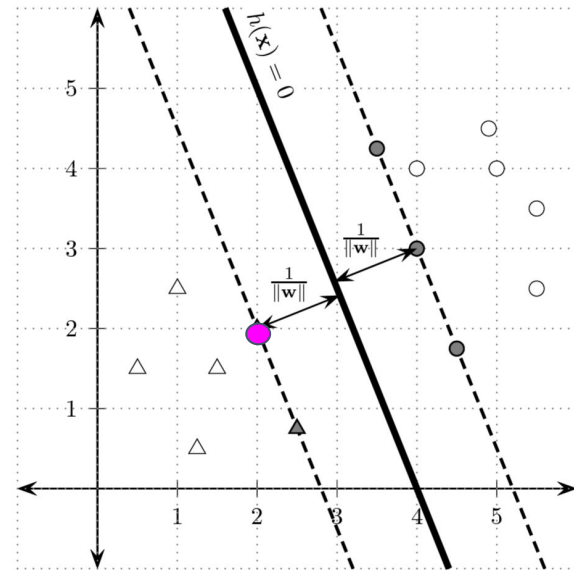
support vector $\mathbf{x}^* = (2, 2)$, with class $y^* = -1$

$$s = \frac{1}{y^* h'(\mathbf{x}^*)} = \frac{1}{-1 \left(\begin{pmatrix} 5 \\ 2 \end{pmatrix}^T \begin{pmatrix} 2 \\ 2 \end{pmatrix} - 20 \right)} = \frac{1}{6}$$

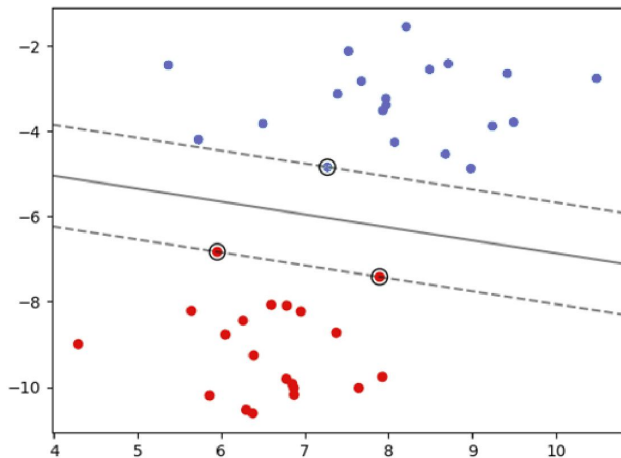
$$\mathbf{w} = \frac{1}{6} \begin{pmatrix} 5 \\ 2 \end{pmatrix} = \begin{pmatrix} 5/6 \\ 2/6 \end{pmatrix} \quad b = \frac{-20}{6}$$

$$h(\mathbf{x}) = \begin{pmatrix} 5/6 \\ 2/6 \end{pmatrix}^T \mathbf{x} - 20/6 = \begin{pmatrix} 0.833 \\ 0.333 \end{pmatrix}^T \mathbf{x} - 3.33$$

$$\delta^* = \frac{y^* h(\mathbf{x}^*)}{\|\mathbf{w}\|} = \frac{-1 \left(\begin{pmatrix} 5/6 \\ 2/6 \end{pmatrix}^T \begin{pmatrix} 2 \\ 2 \end{pmatrix} - 20/6 \right)}{\sqrt{\left(\frac{5}{6}\right)^2 + \left(\frac{2}{6}\right)^2}} = \frac{1}{\frac{\sqrt{29}}{6}} = 1.114$$



... it leads to



$$\min_{w,b} \frac{1}{2} \|w\|^2$$

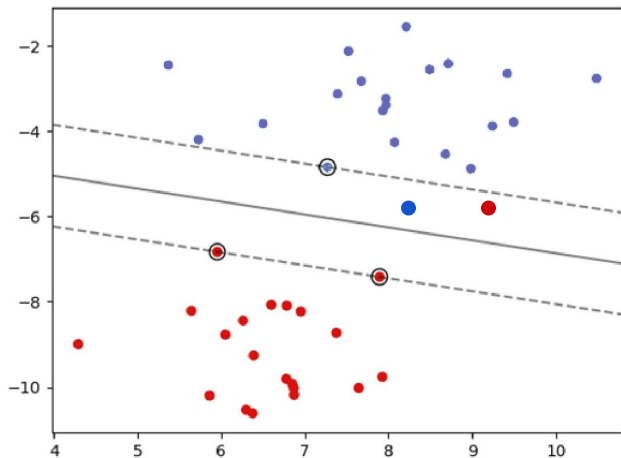
subject to

$$y^{(i)}(w^\top x^{(i)} + b) \geq 1 \quad \text{for } i = 1, \dots, m$$

This is a convex (quadratic) optimization problem
that can be solved efficiently

- Data are D-dimensional *vectors*
- Margins determined by nearest data points called ***support vectors***
- We call this a *support vector machine* (SVM)

If the data is linearly not separable,



$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^m \xi_i \\ \text{subject to} \quad & y^{(i)}(w^\top x^{(i)} + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \text{for } i = 1, \dots, m \end{aligned}$$

Tradeoff between margin and the error!

Support Vector Machine

22

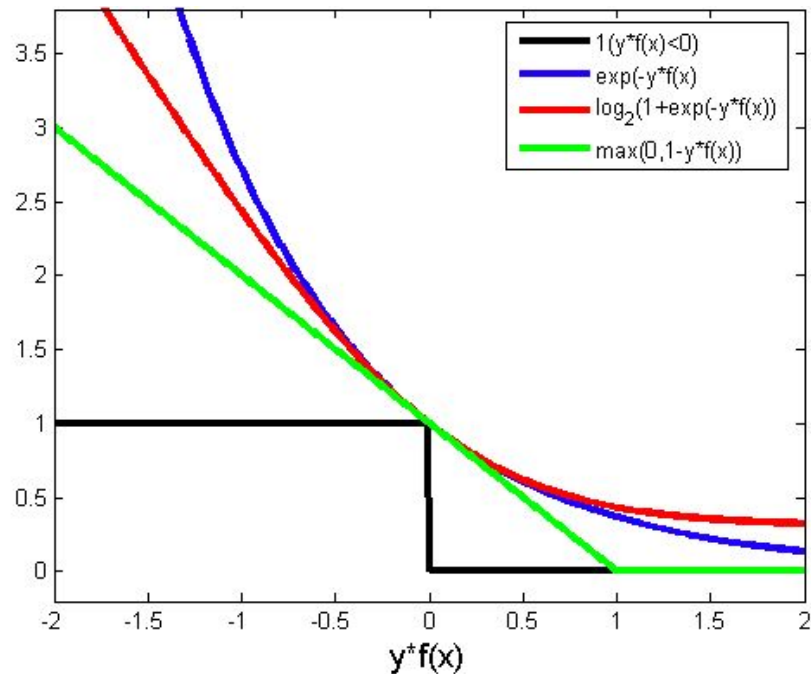
$$\begin{aligned} & \text{minimize } \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^m \xi_i \\ & \text{subject to} \\ & y^{(i)}(w^\top x^{(i)} + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \text{for } i = 1, \dots, m \end{aligned}$$

Equivalent formulation

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m (1 - y^{(i)}(w^\top x^{(i)} + b))_+$$

$$\ell(f; x^{(i)}, y^{(i)}) = (1 - y^{(i)} f(x^{(i)}))_+$$

$$(X)_+ := \max(X, 0)$$



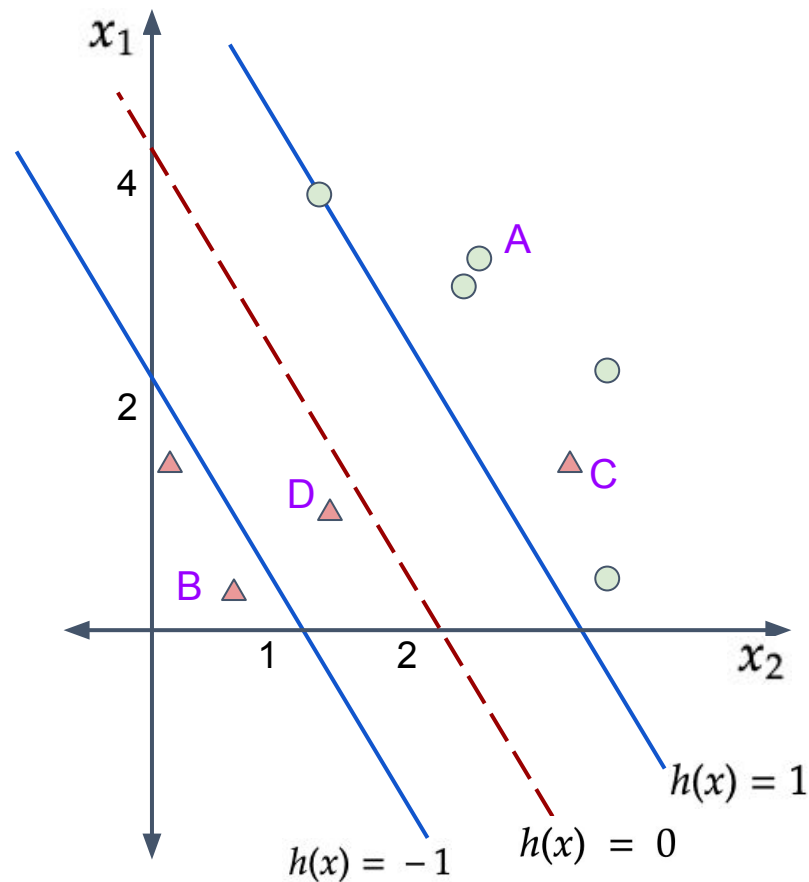
$$\text{Hinge loss} = \max(0, 1 - y_i(w^T x_i + b))$$

$$A: \max(0, 1 - 1 \cdot (> 1)) \rightarrow 0$$

$$B: \max(0, 1 - (-1) \cdot (< -1)) \rightarrow 0$$

$$C: \max(0, 1 - (-1) \cdot (> 1)) \rightarrow > 1$$

$$D: \max(0, 1 - (-1) \cdot (\text{between } [-1, 0])) \\ \rightarrow \text{between } [0, 1]$$



General Principle

2
4

$$\arg \min_{w,b} \frac{1}{2} \|w\|^2 + \textcolor{red}{C} \sum_{i=1}^m (1 - y^{(i)}(w^\top x^{(i)} + b))_+$$

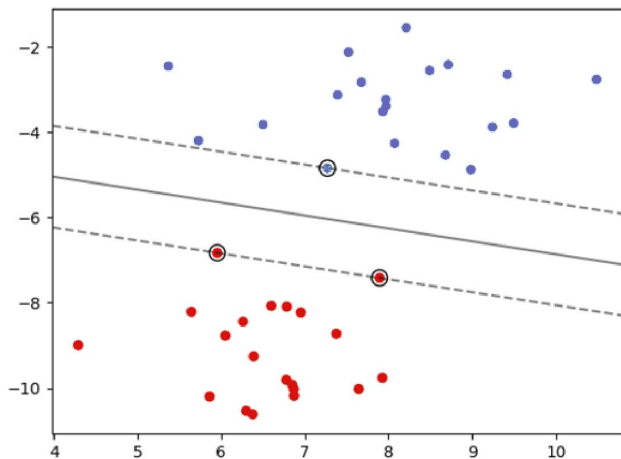
=> by setting $C = 1/\lambda$, it's
equivalent to solve

$$\arg \min_{w,b} \textcolor{red}{\lambda} \frac{1}{2} \|w\|^2 + \sum_{i=1}^m (1 - y^{(i)}(w^\top x^{(i)} + b))_+$$

SVM belongs to the general loss-oriented formulation!

$$\text{Model} = \arg \min_{\text{model}} \text{Loss}(\text{Model}, \text{Data}) + \lambda \cdot \text{Regularizer}(\text{Model})$$

Support Vectors

2
5

$$\min_{w, b} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^m \xi_i$$

subject to

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

$$\xi_i \geq 0 \quad \text{for } i = 1, \dots, m$$

Those data points achieving equality $y^{(i)}(w^\top x^{(i)} + b) = 1 - \xi_i$ are called **support vectors**.

Turns out, if you knew support vectors already, solving the optimization problem above with **just the support vectors as train set** leads to the same solution.

⇒ Leave-one-out cross validation can be done fast!

- Recall: logistic regression had a very natural extension to multi-class.
- What about SVM?

[One-vs-the-rest trick]

- Given: dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$
- For each class $c \in \{1, \dots, C\}$
 - Define label $z^{(i)} \in \{-1, 1\}$ where 1 for class c and -1 for other classes, for all $i=1, \dots, m$.
 - Train a classifier f_c with $\{(x^{(i)}, z^{(i)})\}_{i=1}^m$
- To classify x^* , compute $\hat{y} = \arg \max_{c \in \{1, \dots, C\}} \text{decision_value}(f_c(x^*))$

SVM with linear decision boundaries,

`sklearn.svm.LinearSVC`

Call options include...

penalty : {'l1', 'l2'}, default='l2'

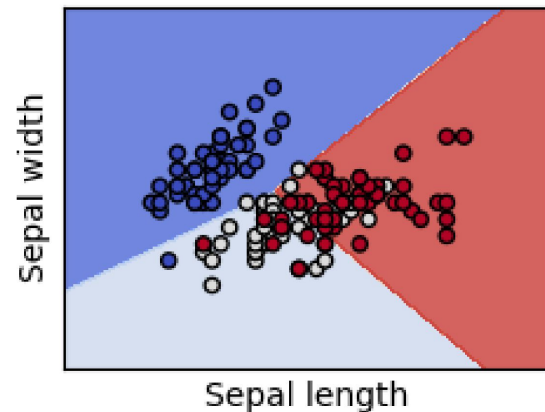
Specifies the norm used in the penalization. The 'l2' penalty is the standard used in SVC. The 'l1' leads to `coef_` vectors that are sparse.

dual : bool, default=True

Select the algorithm to either solve the dual or primal optimization problem. Prefer dual=False when `n_samples > n_features`.

C : float, default=1.0

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.



kernel : {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

gamma : {'scale', 'auto'} or float, default='scale'

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses $1 / (n_features * X.var())$ as value of gamma,
- if 'auto', uses $1 / n_features$.

for RBF,

small γ : complex decision boundary

large γ : more like linear decision boundary

max_iter : int, default=-1

Hard limit on iterations within solver, or -1 for no limit.

verbose : bool, default=False

Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.

class_weight : dict or 'balanced', default=None

Set the parameter C of class i to `class_weight[i]*C` for SVC. If not given, all classes are supposed to have weight one. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`.

Classify among 3 species of Iris flowers...



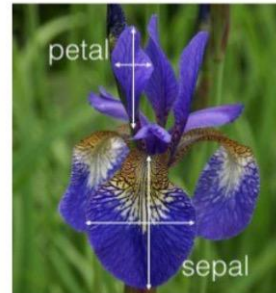
Iris setosa



Iris versicolor

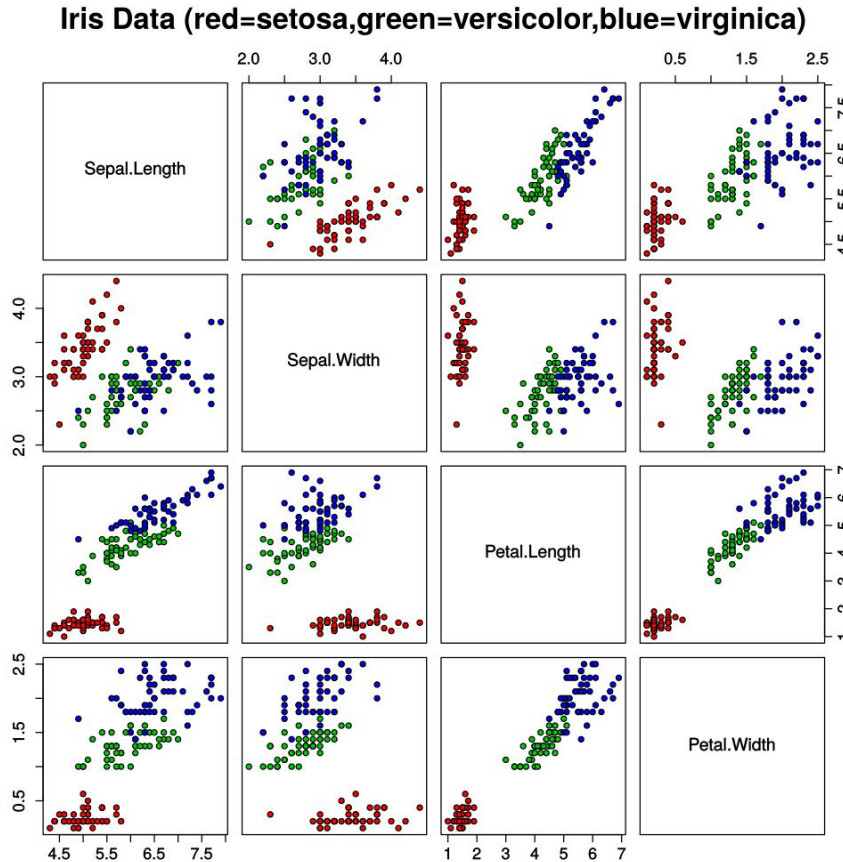


Iris virginica



Four features (in centimeters)

- Petal length / width
- Sepal length / width



*Fairly easy to separate
setosa from others using a
linear classifier*

*Need to use nonlinear basis /
kernel representation to
better separate other classes*

Train 8-degree polynomial kernel SVM classifier,

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(X_train, y_train)
```

Generate predictions on held-out test data,

```
y_pred = svclassifier.predict(X_test)
```

Show confusion matrix and classification accuracy,

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.92	0.96	13
Iris-virginica	0.86	1.00	0.92	6
avg / total	0.97	0.97	0.97	30