Support Vector Machine

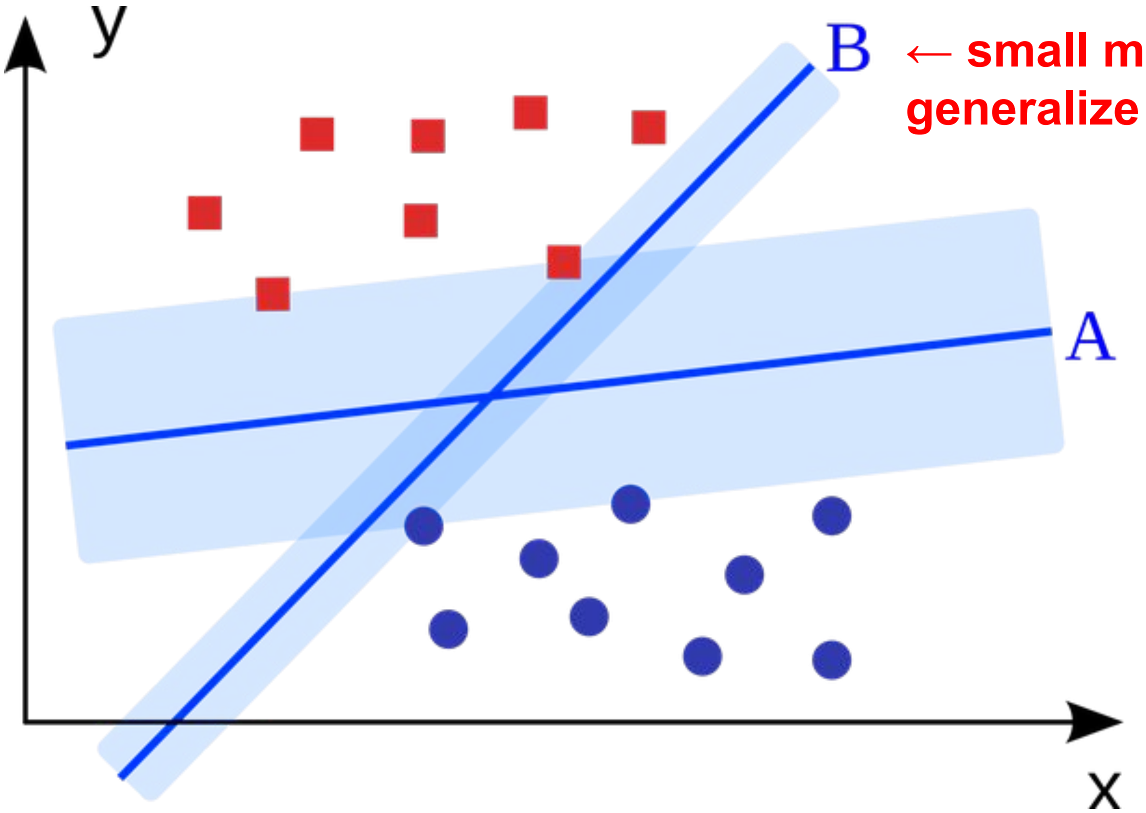
Lecture 6

Today: Learning Objectives

1. Understand **large margin** classification
2. Derive **objective function** for Linear SVM
3. Handle **soft-margin** classification with Hinge Loss

1. Large margin classification

Large Margin Classifier



← small margin does not generalize well

← more robust classification with larger margin

Introducing Support Vector Machine

- A large margin classifier
- Capable of non-linear **classification**, **regression**, and **outlier detection**
- Particularly suited for classification of **complex** and **mid-sized datasets**
- Those who are interested in ML should have Support Vector Machine (SVM) in their toolbox

History of SVM

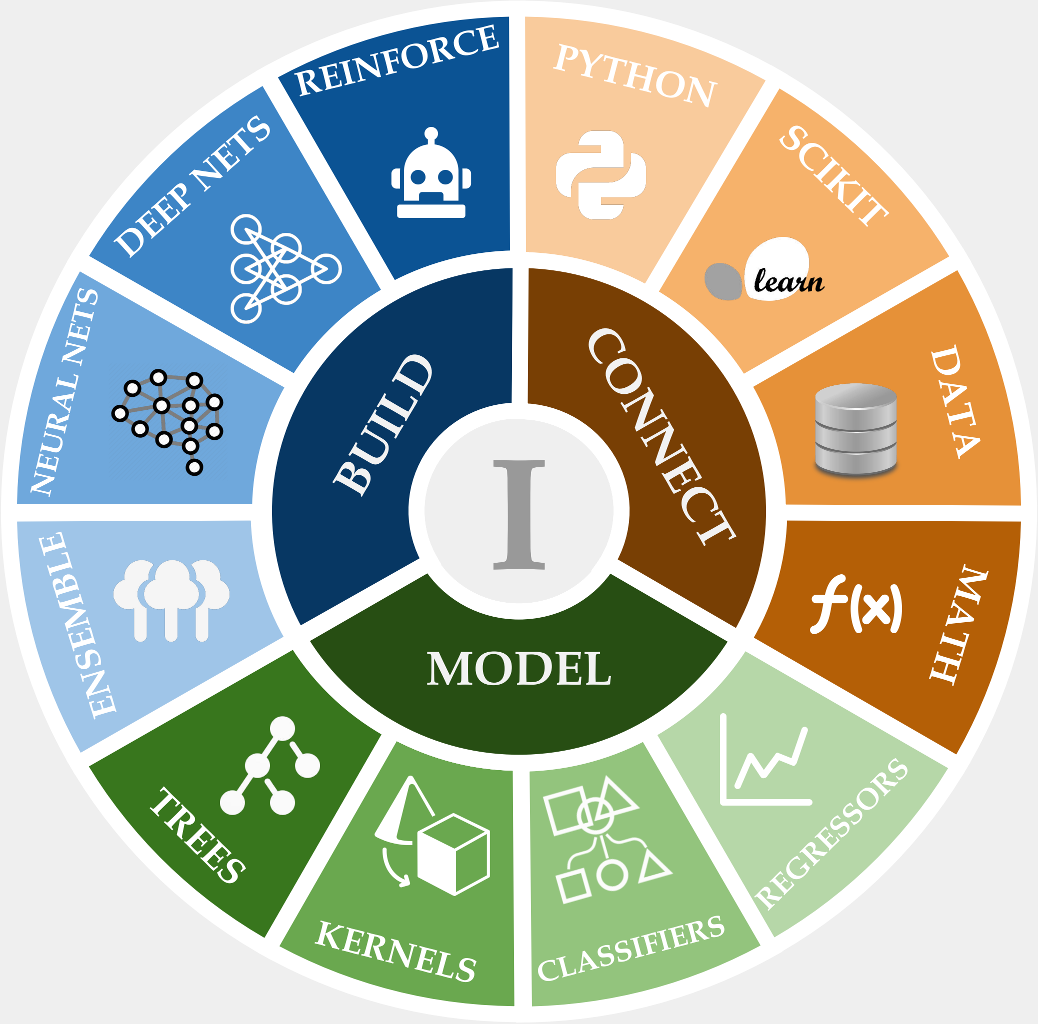
First introduced in 1992 inspired from a statistical learning theory*

Became popular because of its success in MNIST digit recognition (1994)

Had lots successful applications in Computer Vision, Text Categorization, Ranking, Time Series Analysis, and BioInformatics, ect.

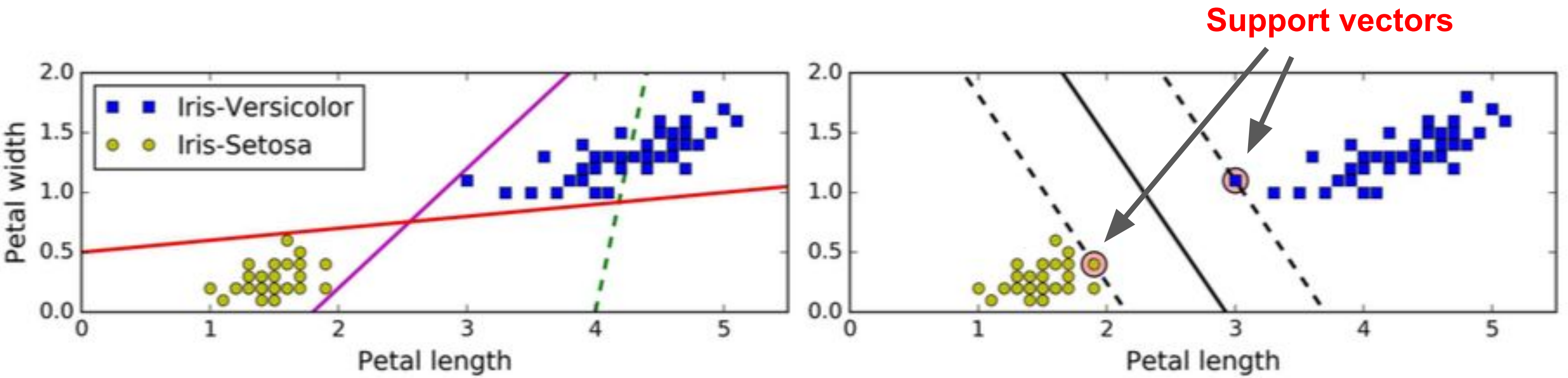
Regarded as an important example of "**kernel methods**", arguably the hottest area in machine learning in the early 2000s

* B.E. Boser et al. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.


I
BUILD
CONNECT
MODEL
REINFORCE
DEEP NETS
NEURAL NETS
ENSEMBLE
TREES
KERNELS
CLASSIFIERS
REGRESSORS
MATH
f(x)
DATA
SCIKIT
learn
PYTHON

Linear SVM

Linear SVM classifier separate two classes but also stay as far away from the closest training samples as possible → **maximized the margin**



Decision boundary is fully determined (or supported) by the samples located “on the edge of the street” → support vectors

A bit on a fun note...

What did one support vector say to another?

….

Man, I feel so marginalized!

SVM Model

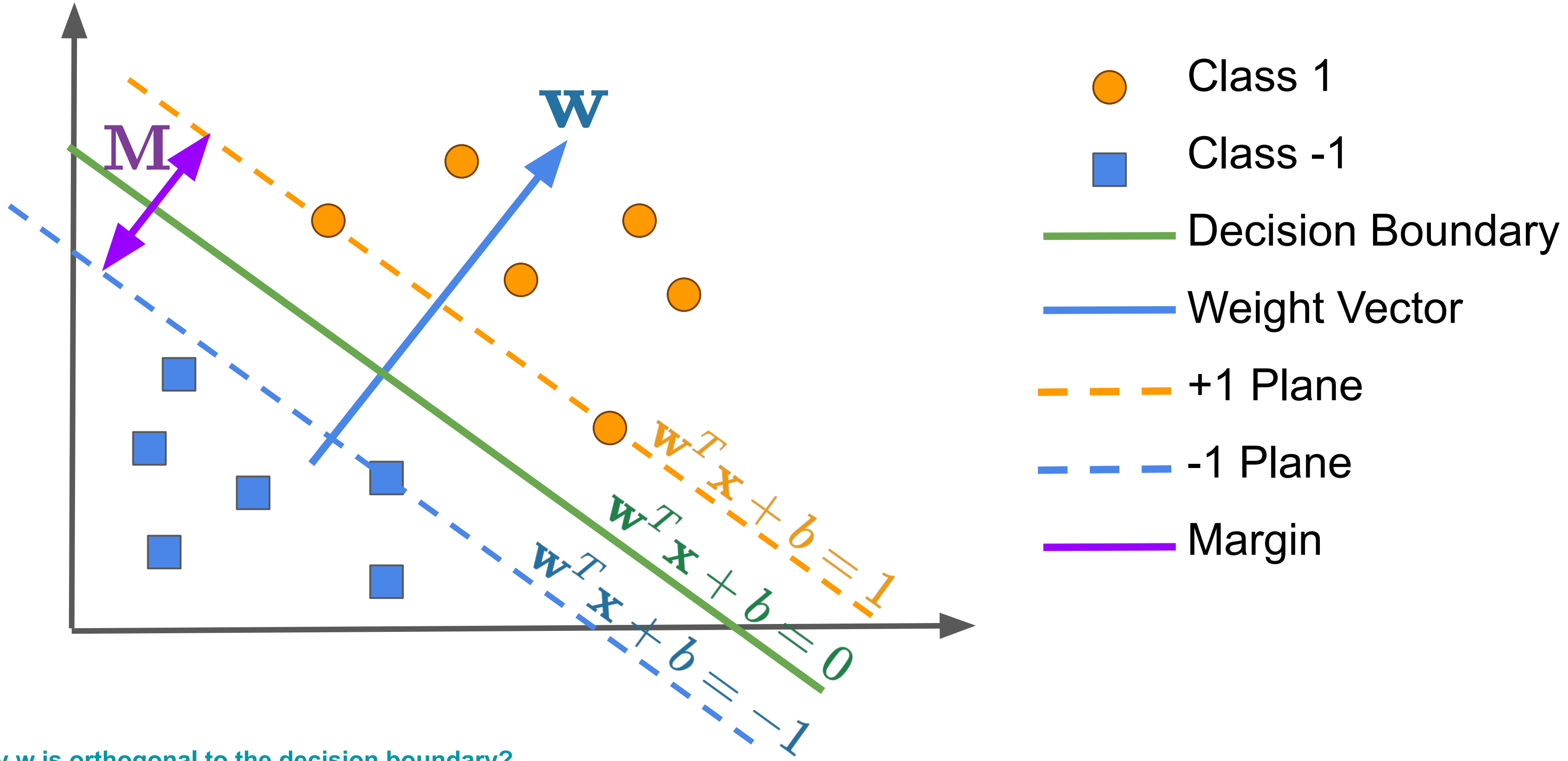
Notation **change**:

- Bias term will be called b (no longer θ_0)
- Feature weight vector will be call $\mathbf{w}$ (no longer θ)

Linear SVM classifier model predicts the class of sample $\mathbf{x}$ by computing:

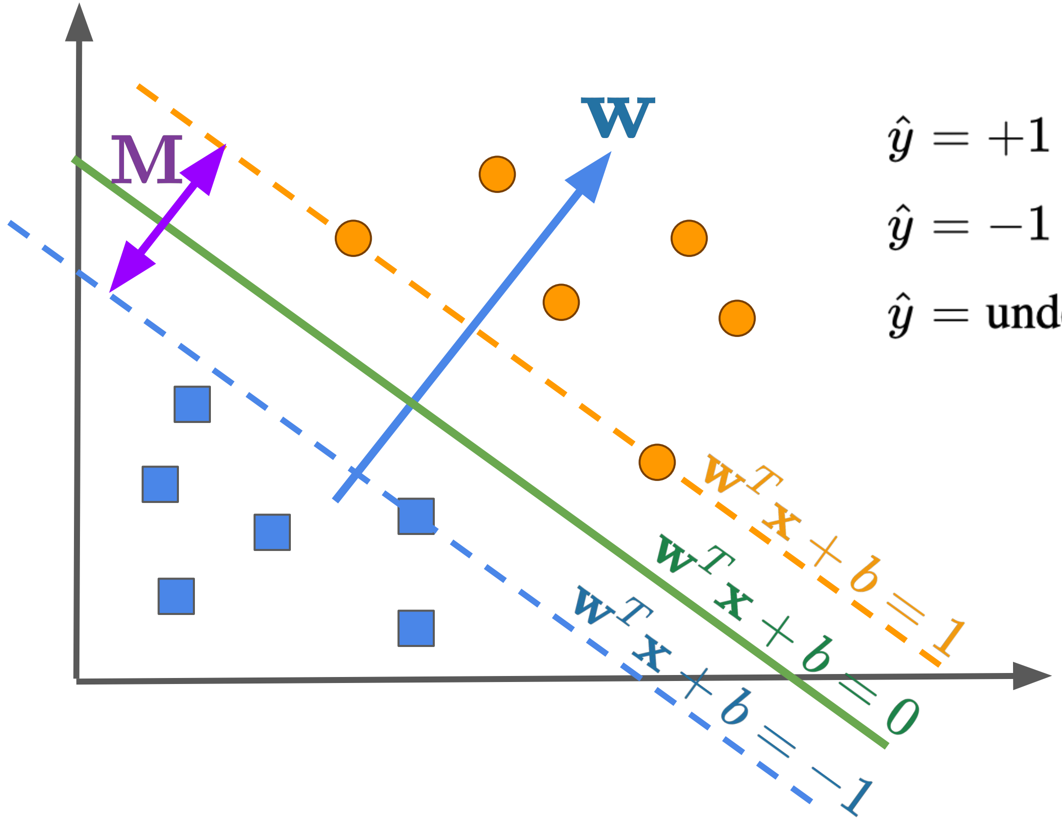
$$\hat{y} = \mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

Geometry Interpretation



[*See why w is orthogonal to the decision boundary?]()

Predicting a label



$$\hat{y} = +1 \text{ if } \mathbf{w}^T\mathbf{x} + b \geq +1$$

$$\hat{y} = -1 \text{ if } \mathbf{w}^T\mathbf{x} + b \leq -1$$

$$\hat{y} = \text{undefined} \quad \text{if } \; -1 \leq \mathbf{w}^T\mathbf{x} + b \leq +1$$

An Example in Python

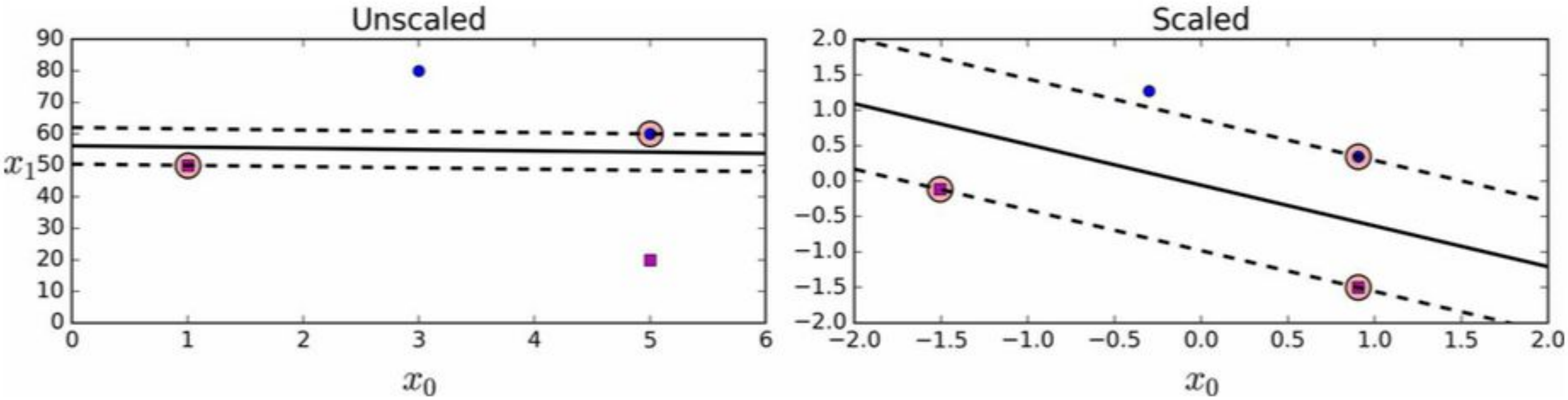
```python
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = datasets.load_iris()
X = iris["data"][:, (2, 3)]  # petal length, petal width
y = (iris["target"] == 2).astype(np.float64)  # Iris-Virginica

svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge", random_state=42)),
    ])

svm_clf.fit(X, y)
```

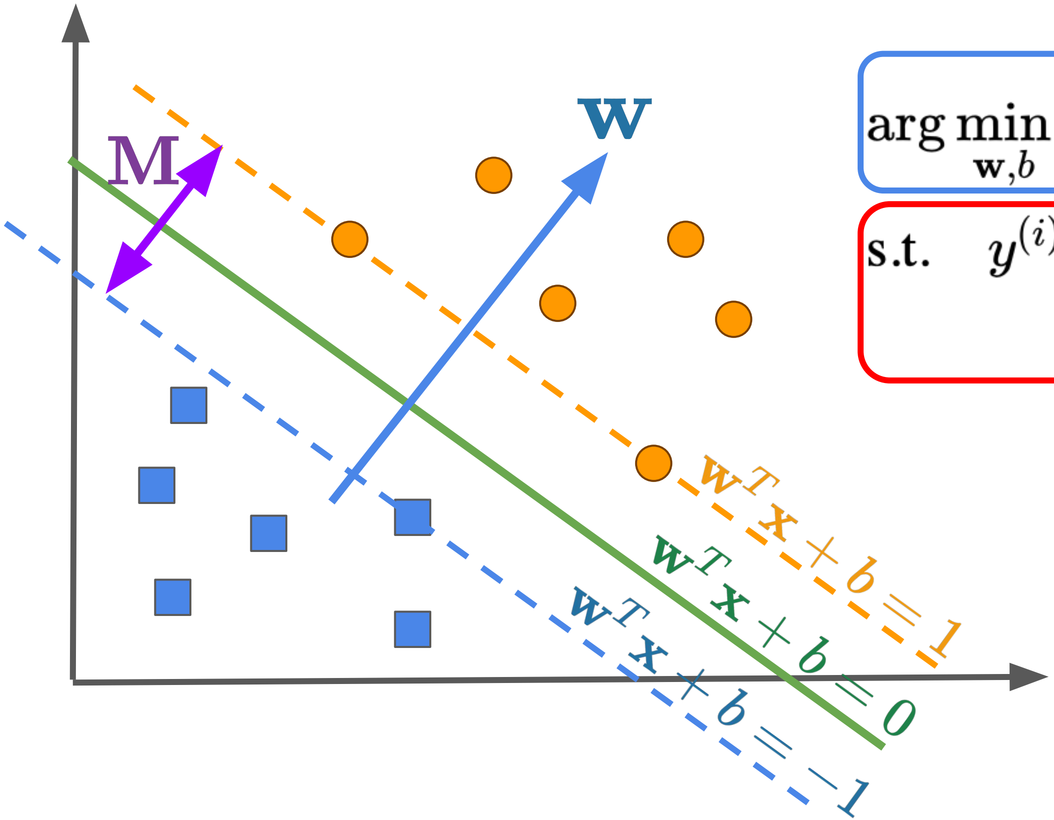
SVM margin is sensitive to feature scales



Make sure to use feature scaling (with StandardScaler)

2. Formulating SVM objective function

SVM Objective Function (aka how to find w, b)



$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

SVM Objective

$$\text{s.t.} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \text{ for } i = 1, \ldots, m$$

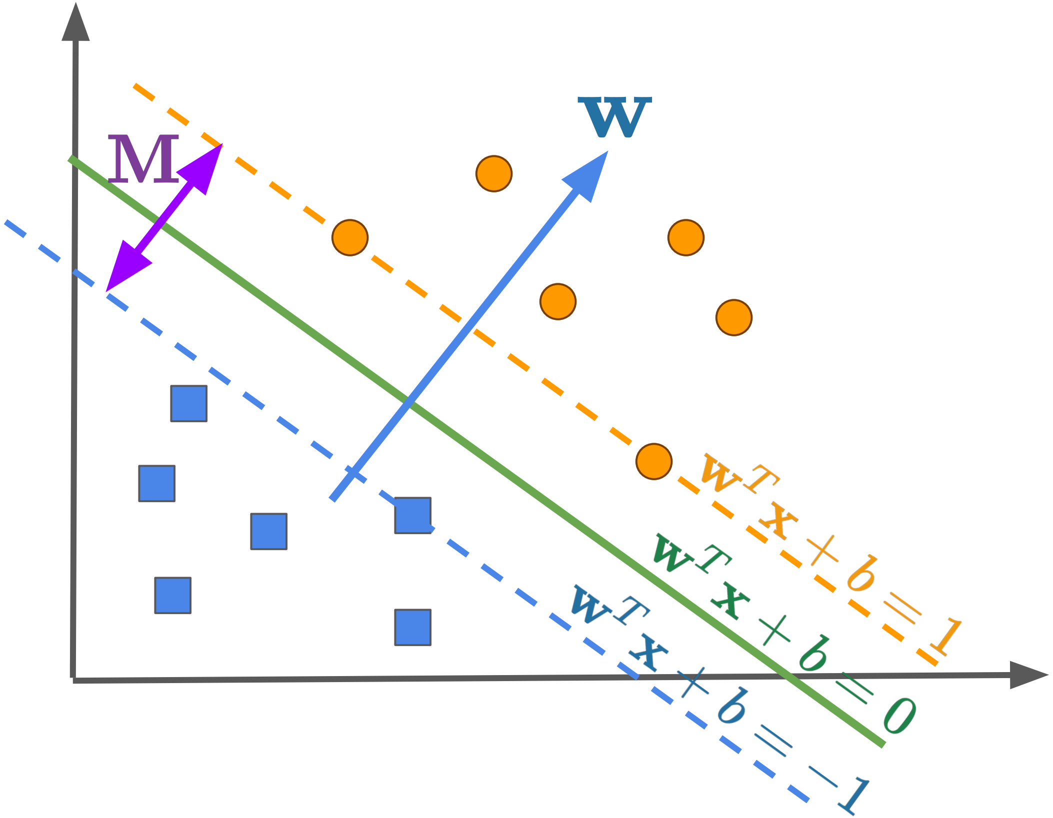
SVM Constraints

In English, *this means trying to maximize the margin while correctly classifying all training examples.*

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

SVM **Constraints**

$$\boxed{\text{s.t.} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \text{ for } i = 1, \ldots, m}$$



For all x in positive class:

$$y = +1 \text{ if } \mathbf{w}^T\mathbf{x} + b \geq +1$$

$$\Rightarrow y(\mathbf{w}^T\mathbf{x} + b) \geq 1$$

For all x in negative class:

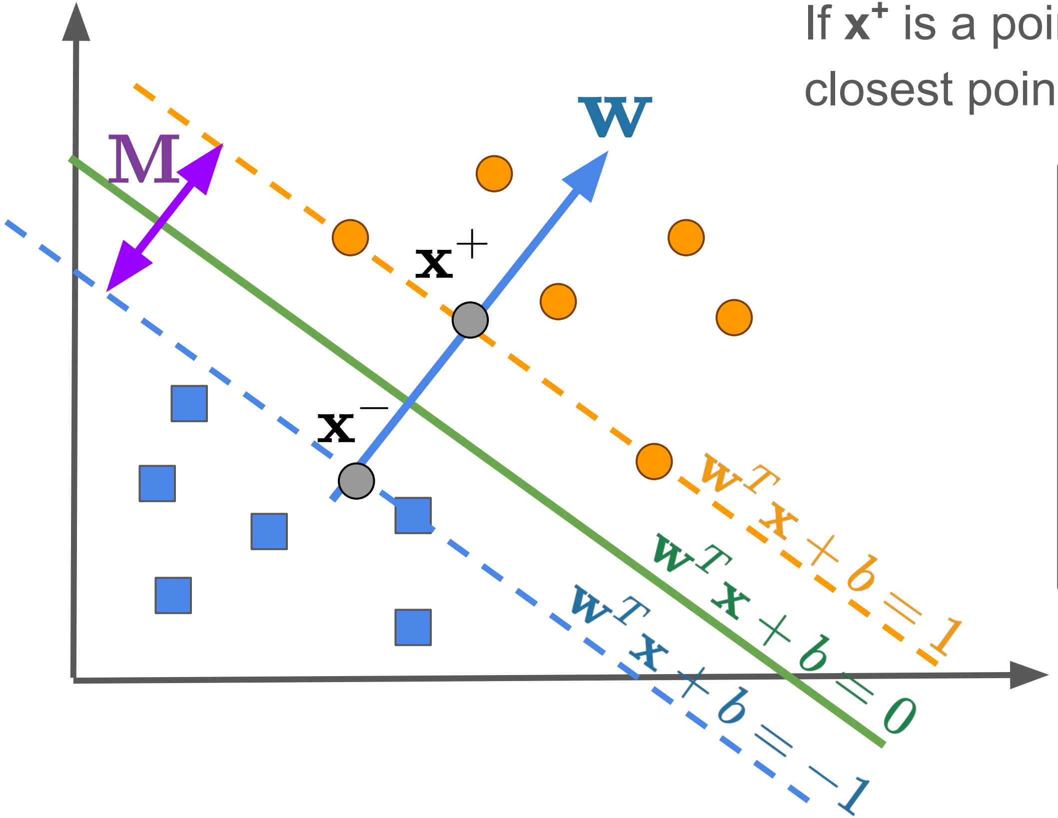
$$y = -1 \text{ if } \mathbf{w}^T\mathbf{x} + b \leq -1$$

$$\Rightarrow y(\mathbf{w}^T\mathbf{x} + b) \geq 1$$

Therefore, for all $x^{(i)}$, $y^{(i)}$ in train set:

$$\boxed{y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1}$$

Geometric observations



If $\mathbf{x}^+$ is a point on the +1 plane and $\mathbf{x}^-$ is the closest point to $\mathbf{x}^+$ on the -1 plane, then:

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

$$\mathbf{w}^T\mathbf{x}^- + b = -1 \quad (2)$$

$$\mathbf{x}^+ = \lambda\mathbf{w} + \mathbf{x}^- \quad (3)$$

$$\mathbf{M} = ||\mathbf{x}^+ - \mathbf{x}^-|| \quad (4)$$

What does λ equal to?

$$\mathbf{w}^T\mathbf{x}^+ + b = +1 \qquad (1)$$
$$\mathbf{w}^T\mathbf{x}^- + b = -1 \qquad (2)$$
$$\mathbf{x}^+ = \lambda\mathbf{w} + \mathbf{x}^- \qquad (3)$$
$$\mathbf{M} = ||\mathbf{x}^+ - \mathbf{x}^-|| \qquad (4)$$

$$\mathbf{w}^T\mathbf{x}^+ + b = +1$$ start with (1)

$$\Rightarrow \mathbf{w}^T(\lambda\mathbf{w} + \mathbf{x}^-) + b = +1$$ because of (3)

$$\Rightarrow \mathbf{w}^T\lambda\mathbf{w} + \mathbf{w}^T\mathbf{x}^- + b = +1$$ multiply out

$$\Rightarrow \mathbf{w}^T\lambda\mathbf{w} + (-1) = +1$$ because of (2)

$$\Rightarrow \lambda = \frac{2}{\mathbf{w}^T\mathbf{w}} = \frac{2}{||\mathbf{w}||^2} \qquad (5)$$ isolate λ

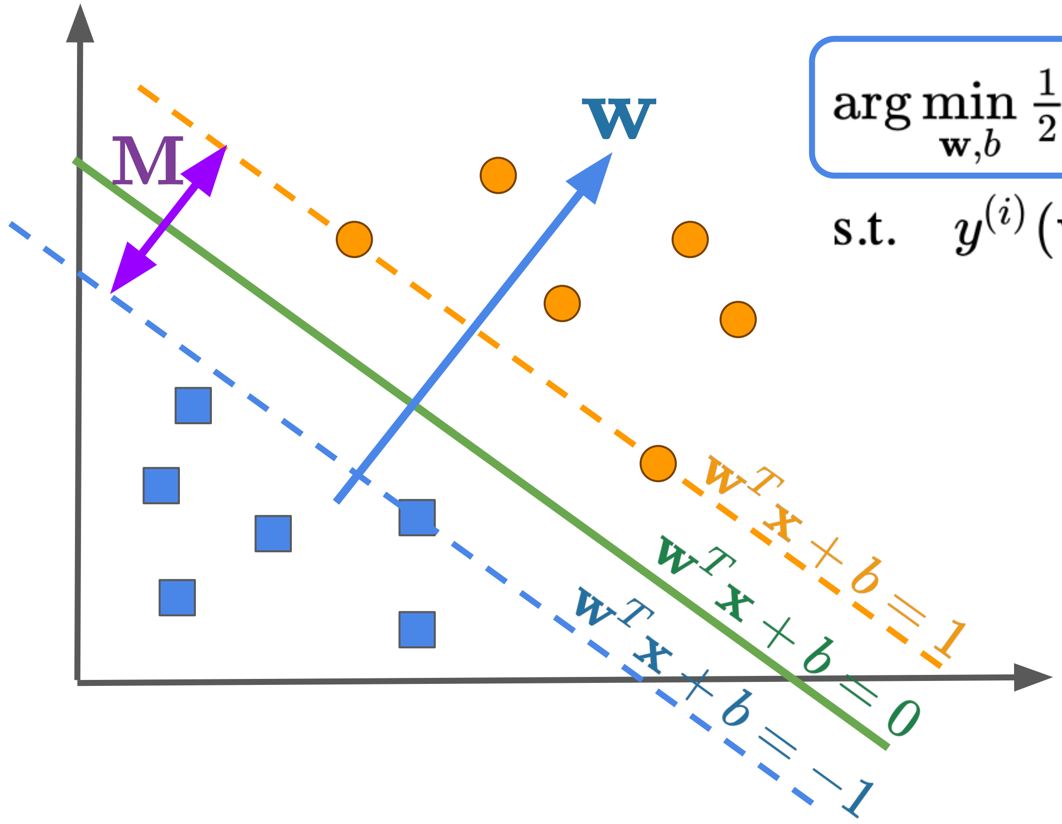
$$\mathbf{w}^T\mathbf{x}^+ + b = +1 \quad (1)$$
$$\mathbf{w}^T\mathbf{x}^- + b = -1 \quad (2)$$
$$\mathbf{x}^+ = \lambda\mathbf{w} + \mathbf{x}^- \quad (3)$$
$$\mathbf{M} = ||\mathbf{x}^+ - \mathbf{x}^-|| \quad (4)$$
$$\lambda = \frac{2}{||\mathbf{w}||^2} \quad (5)$$

Computing the margin M

$$\mathbf{M} = ||\mathbf{x}^+ - \mathbf{x}^-|| \qquad \text{start with (4)}$$
$$= ||(\lambda\mathbf{w} + \mathbf{x}^-) - \mathbf{x}^-|| \qquad \text{because of (3)}$$
$$= \lambda||\mathbf{w}|| \qquad \text{simplify}$$
$$= \frac{2}{||\mathbf{w}||^2}||\mathbf{w}|| \qquad \text{because of (5)}$$
$$\boxed{= \frac{2}{||\mathbf{w}||}} \qquad \text{simplify}$$

So if you want to maximize **M**, just minimize **w. NICE!**

Revisit: SVM objective (margin maximization)



$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

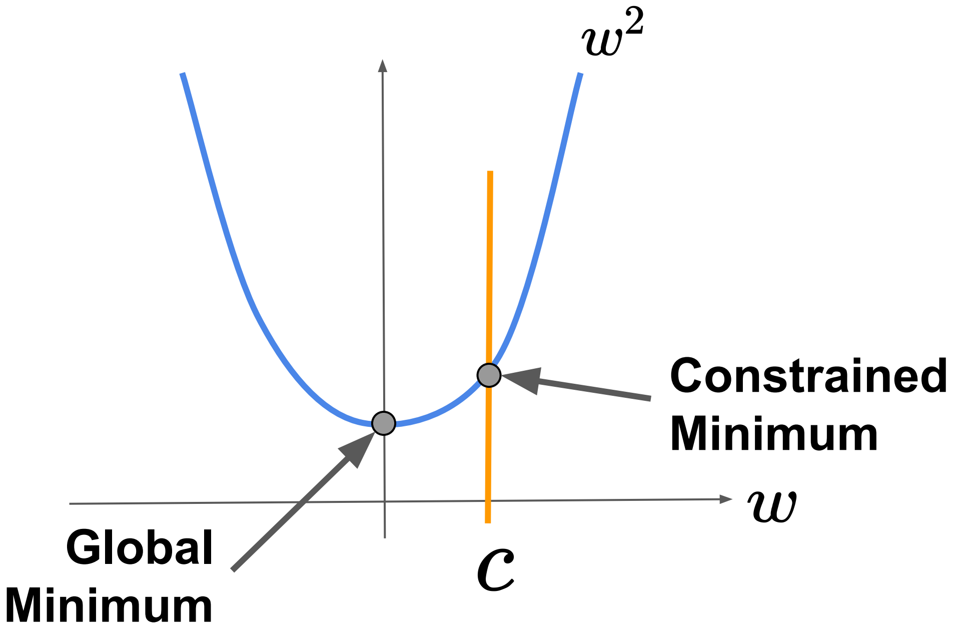
minimize $||w||^2$ insteads of $||w||$

$$\text{s.t.} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \text{ for } i = 1, \ldots, m$$

This can be solved using quadratic programming:

- Quadratic objective
- Linear constraints

Quadratic Optimization



$$\min_{w} w^2$$

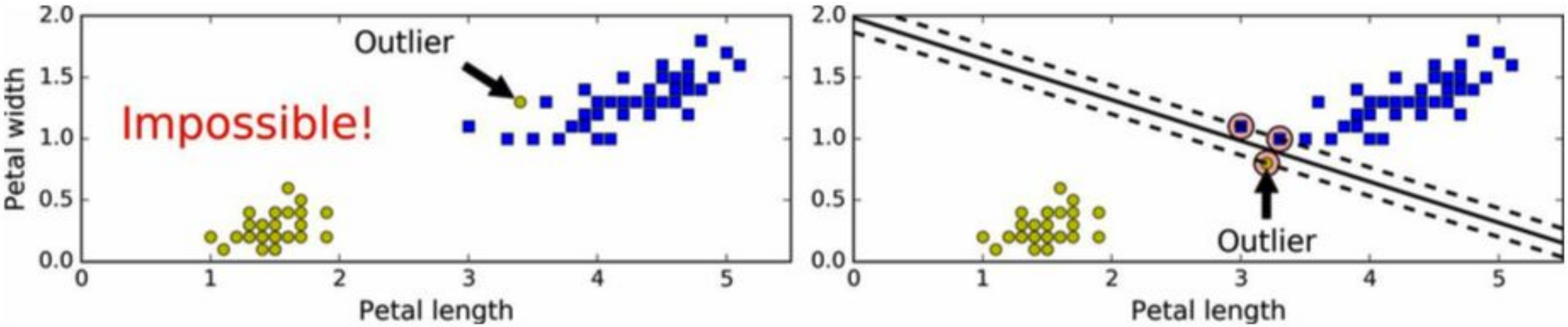
$$\text{s.t.} \quad w \geq c$$

This is a convex quadratic optimization problem which can be solved by **quadratic programming**. We can just use **off-the-self solvers** for this.

Hard Margin Classification

So far, we've used **hard margin** classification: all training samples are on the "correct side of the street":

- Only work if the data is linearly separable (Left Figure)
- Sensitive to **outliers** → not generalize (Right Figure)



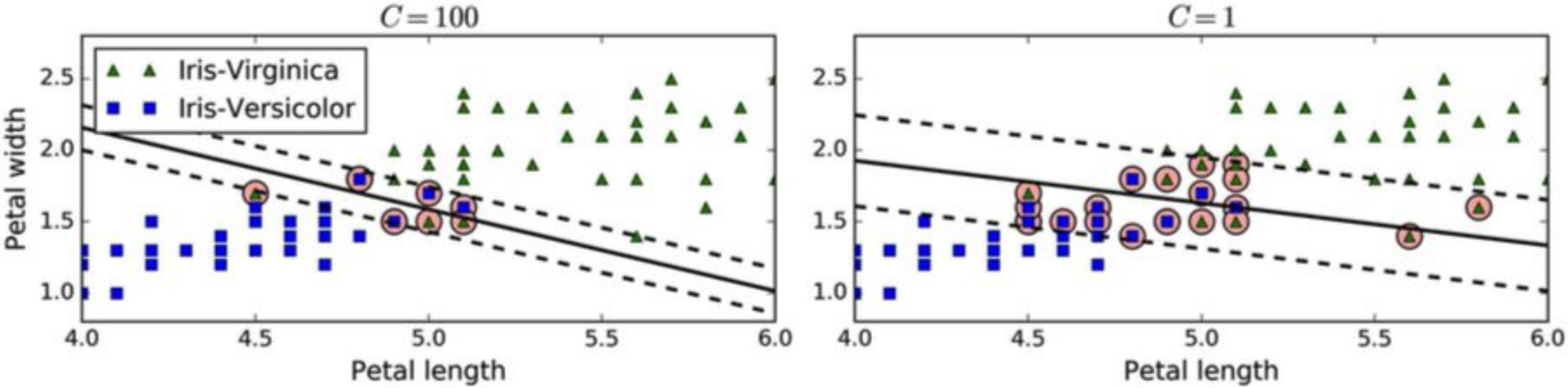
We need to use a more flexible model

3. Soft Margin Classification

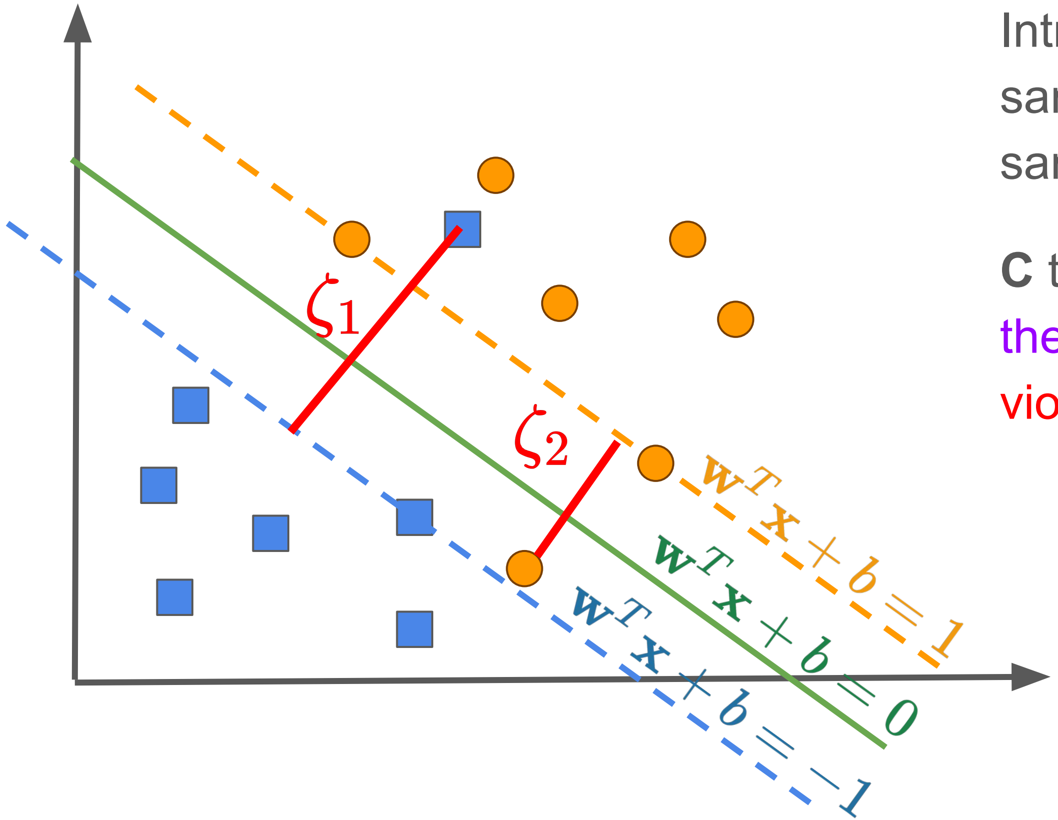
Soft Margin Classification

Objective: find a good balance between keeping the margin as large as possible while limiting the margin violations

Controlled by hyperparameter **C**: smaller value → larger margin but more violation



Soft margin linear SVM (soft SVM)



Introducing a slack variable ζ for each sample to measures how much the sample is allowed to violate the margin.

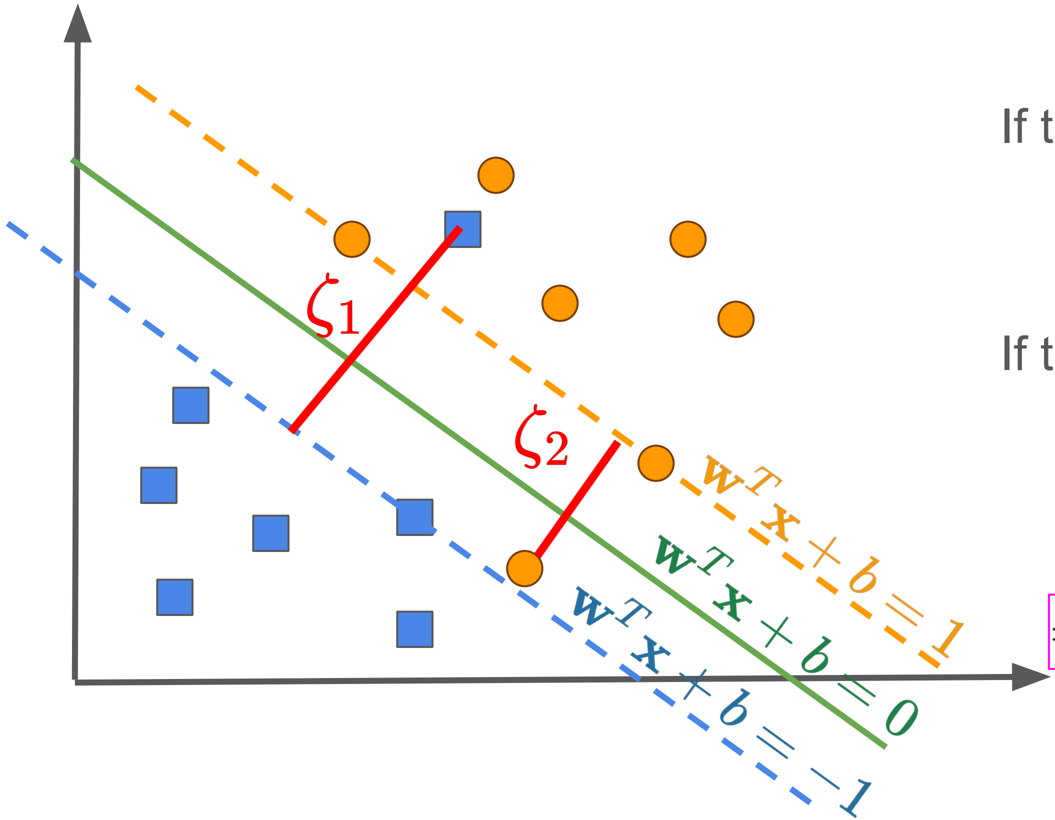
C to control to tradeoff between maximize the margin and minimize the margin violation.

$$\underset{\mathbf{w},b,}{\arg\min} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\zeta_i$$

subject to $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \zeta_i$

and $\zeta_i \geq 0$ for $i = 1, \ldots, m$

Choosing the value of ζ_i



$$\arg\min_{\mathbf{w},b,} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\zeta_i$$
$$\text{subject to } y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta_i$$
$$\text{and } \zeta_i \geq 0 \text{ for } i = 1, \ldots, m$$

If the example is safe:

$$y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1, \text{ so}$$
$$\zeta_i = 0$$

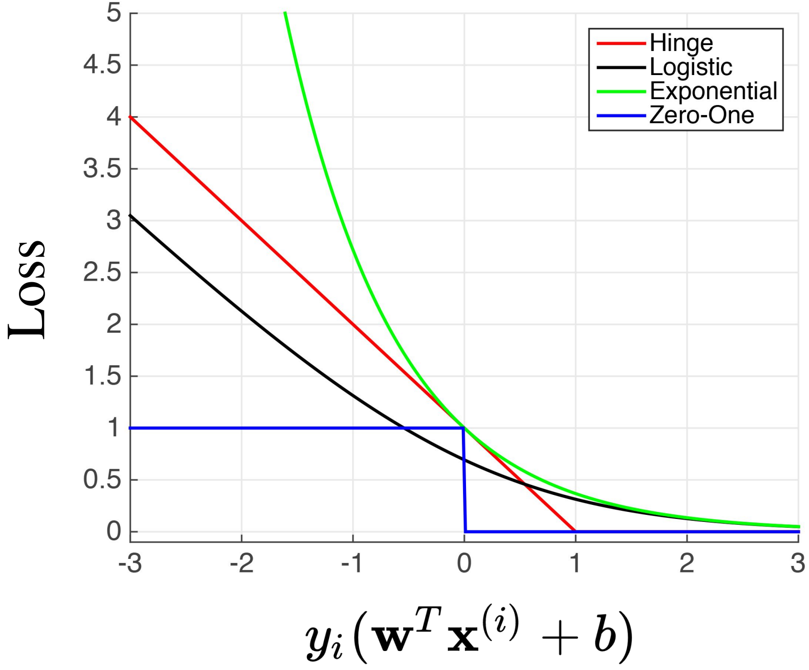
If the example is violating the margin:

$$y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) < 1, \text{ so}$$
$$\zeta_i = 1 - y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)$$

$$\boxed{\Rightarrow \zeta_i = \max(0, 1 - y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right))}$$

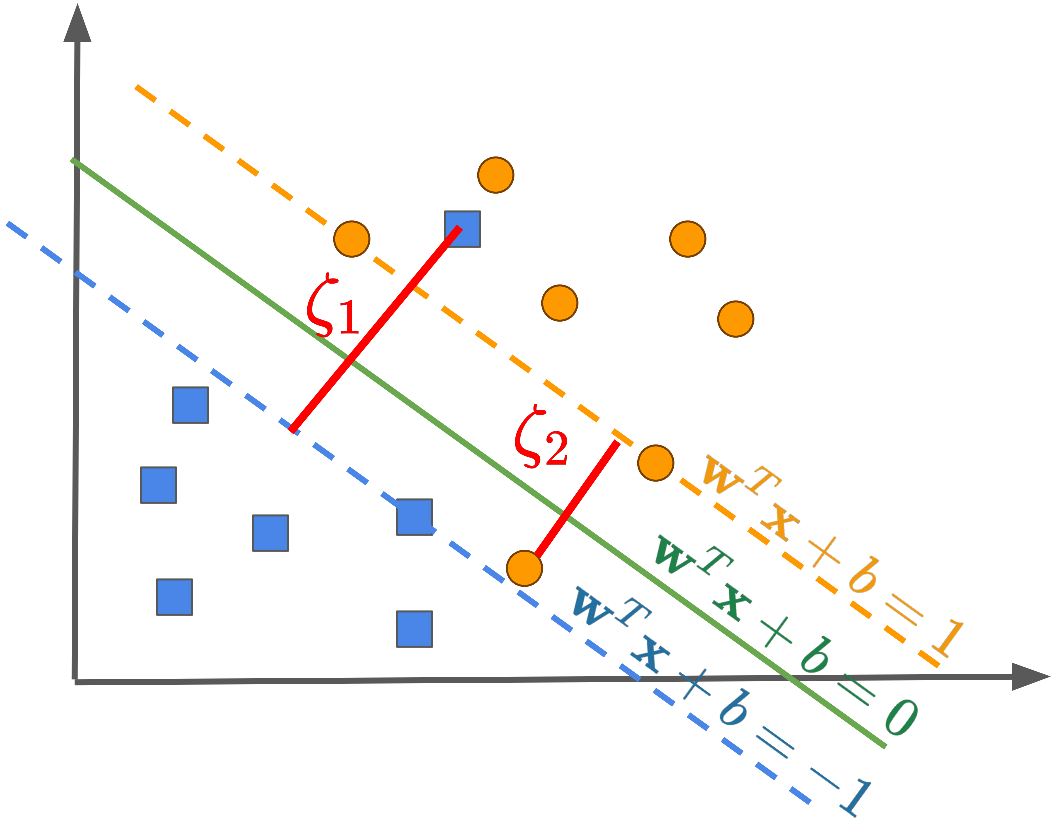
Hinge Loss for data

$$\boxed{\zeta_i = \max(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)) = L_{\text{hinge}}(y^{(i)}, \mathbf{w}^T\mathbf{x}^{(i)} + b)}$$



"**hinge loss**" analogy

Equivalent formulation for objective function



$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\zeta_i$$

$$\arg\min_{\mathbf{w},b} \boxed{\frac{1}{2}\mathbf{w}^T\mathbf{w}} + C\boxed{\sum_{i=1}^{m} L_{\text{hinge}}(y_i, \mathbf{w}^T\mathbf{x}^{(i)} + b)}$$

Regularization

Hinge Loss on Data

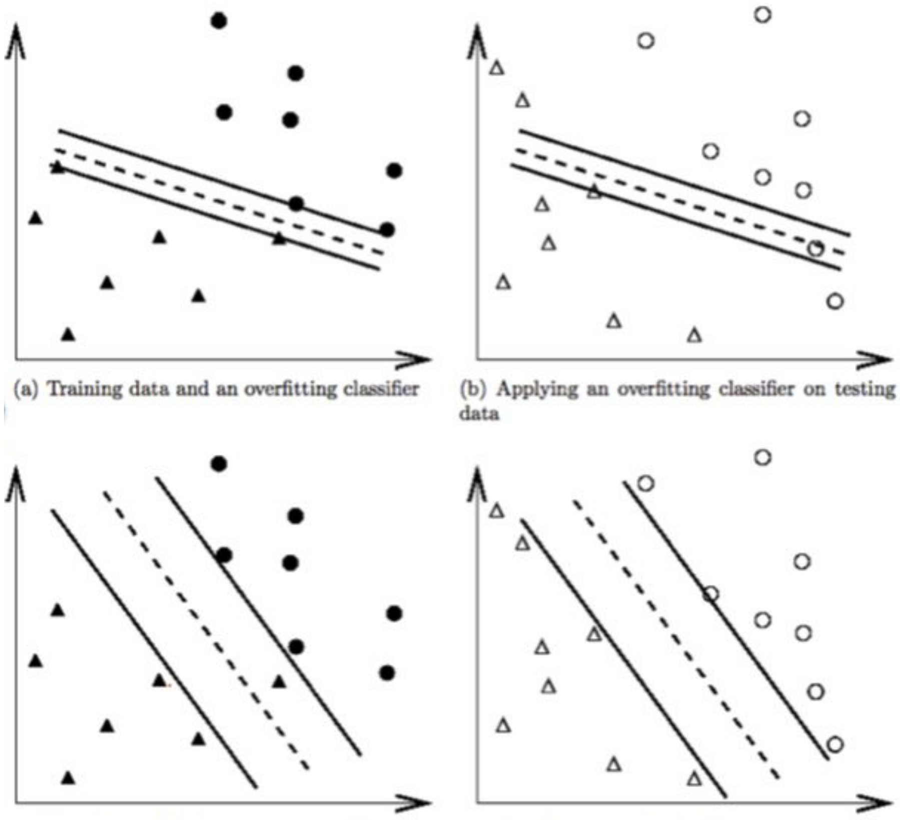
Again, **C** is used to control the tradeoff between maximize the margin and minimize hinge loss.

Finding the right C

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} L_{\text{hinge}}(y_i, \mathbf{w}^T\mathbf{x}^{(i)} + b)$$

Large C: means that misclassification are bad -- resulting in smaller margin and less training error

Small C: Resulting in larger margin and more training error, but hopefully better testing error



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

Online SVM

Online learning: learning incrementally as new training samples arrive.

One way is to use Gradient Descent to minimize the cost function, unfortunately it converges much more slowly than Quadratic Programming based method.

$$J(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b))$$

Summary: Learning Objectives

✓ Know **maximized margin** classification
✓ Derive **objective function** for Linear SVM
✓ Handle **soft-margin** classification with Hinge Loss

Next week

- On Tuesday: Industry Speaker: **Spencer Jenkins, Data Scientist at WillowTree** - “Bottleneck Features and Find-tuning”

  Deep learning (DL) models are popular and powerful tools in the machine learning community. In this talk, we will discuss two simple techniques that leverage pre-trained models, allowing for successful results with limited understanding of neural networks. Fine-tuning a neural network (or using “bottleneck” features from it) allows one to adapt an existing model to a different problem of interest. We will first briefly discuss neural networks at a high level, building off previous knowledge of regression. We will then discuss the ideas behind “bottleneck” features and fine-tuning. Finally, we will go over how simple these techniques are to implement in the Keras framework. At the end of this lecture, you will be able to utilize existing DL models to quickly and successfully solve problems of your own interest.

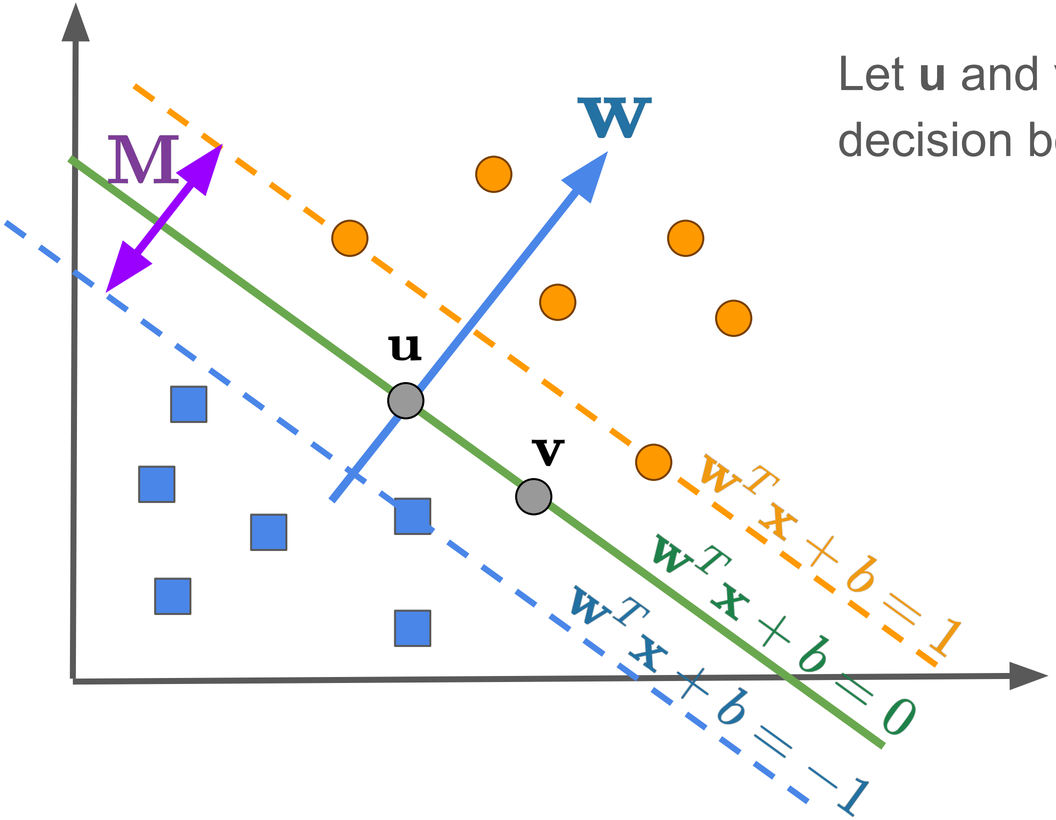
- On Thursday: It’s time for the **midterm**: Best of luck!

After Fall break

- ❑ Continue with SVM: Expand to **non-linear** case
- ❑ Learn Gaussian Radial Basis Function **(RBF) Kernel**
- ❑ Understand the kernel trick

Bonus Materials

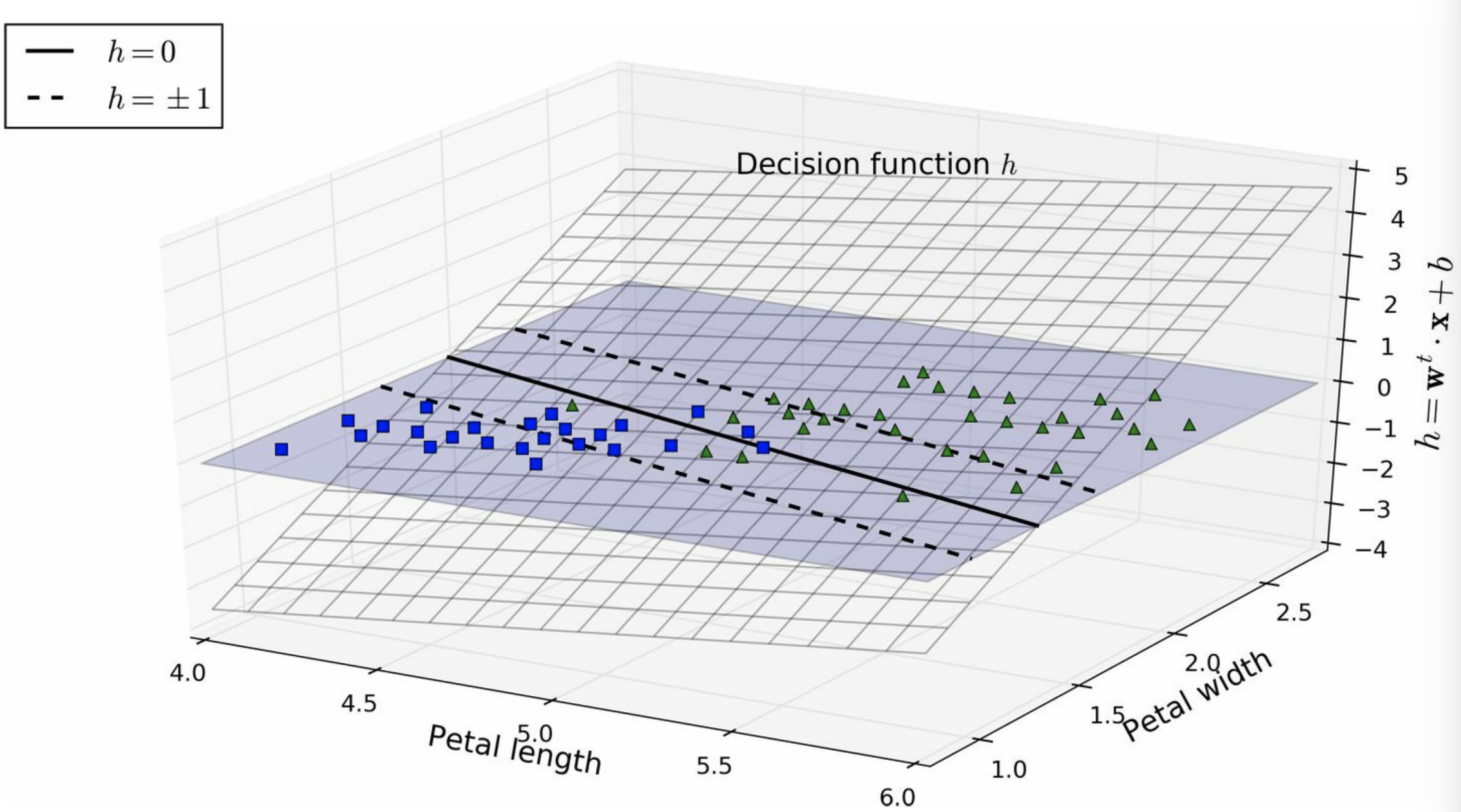
Why w is orthogonal to the decision boundary?



Let **u** and **v** be two points on the decision boundary, then:

$$\mathbf{w}^T\mathbf{u} + b = 0$$
$$\mathbf{w}^T\mathbf{v} + b = 0$$
$$\mathbf{w}^T(\mathbf{u} - \mathbf{v}) = 0$$
$$\rightarrow \mathbf{w} \perp (\mathbf{u} - \mathbf{v})$$

In 3D: Decision Function for Iris dataset

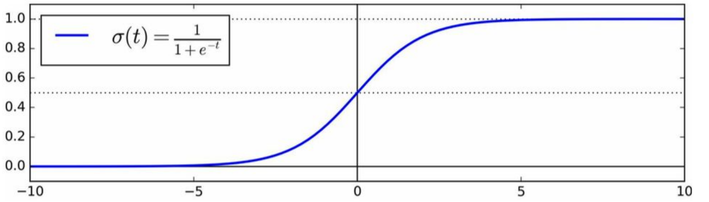


1. Learn about Hinge Loss

Recall from Logistic Regression

Logistic Regression estimates: $$\hat{p} = h_\theta(\mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$$

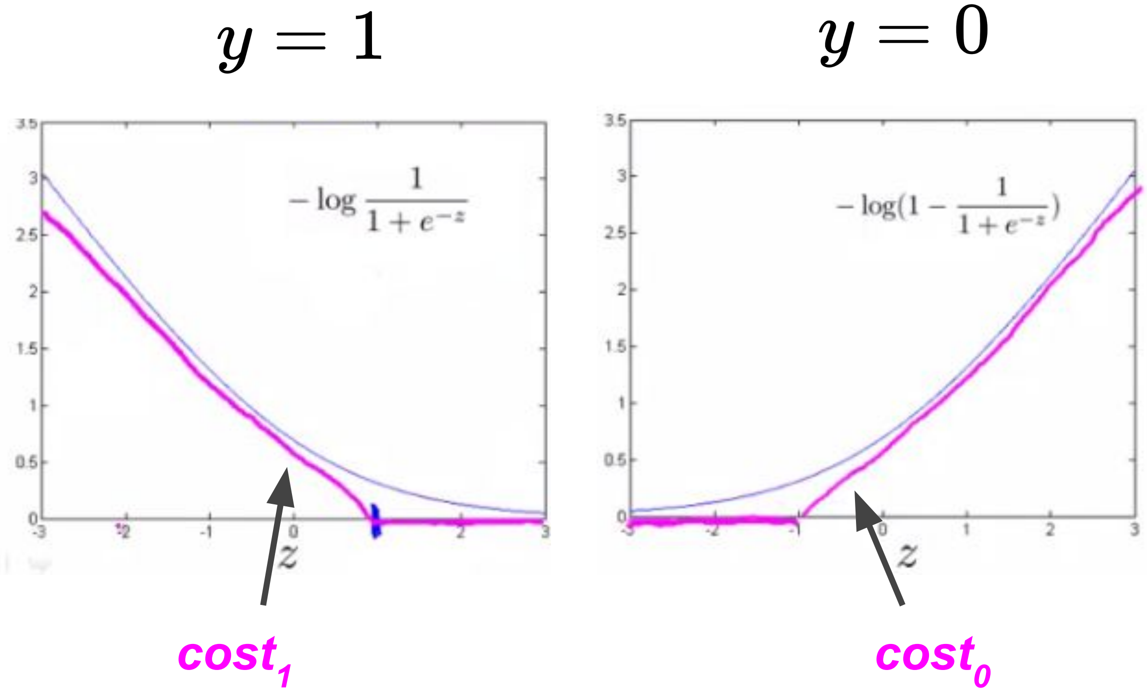
$$y = 1 \rightarrow h_\theta(\mathbf{x}) \approx 1 \rightarrow \theta^T \mathbf{x} \gg 0$$
$$y = 0 \rightarrow h_\theta(\mathbf{x}) \approx 0 \rightarrow \theta^T \mathbf{x} \ll 0$$



Cost Function:

$$\text{Cost }(h_\theta(\mathbf{x}), y) = \begin{cases} -\log(h_\theta(\mathbf{x})) \text{ if } y = 1 \\ -\log(1 - h_\theta(\mathbf{x})) \text{ if } y = 0 \end{cases}$$

New Cost Function



Let's change into lines: One with some slope, the other is flat, which gives us:

- Approximation of regression
- Computational advantage
- Easier optimization

This is also called "**hinge loss**" vs. "log loss."

Rewrite Cost Function

Start with Logistic Regression Cost Function:

$$\min_\theta \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\left(-\log h_\theta(\mathbf{x}^{(i)})\right)+(1-y^{(i)})\left(-\log(1-h_\theta(\mathbf{x}^{(i)}))\right)\right]+\frac{\lambda}{2}\sum_{j=1}^{n}\theta_j^2$$

Rewrite with the new cost and remove (1/m)

$$\min_\theta \underbrace{\sum_{i=1}^{m}\left[y^{(i)}\mathrm{cost}_1(\theta^T\mathbf{x}^{(i)})+(1-y^{(i)})\mathrm{cost}_0(\theta^T\mathbf{x}^{(i)})\right]}_{A}+\frac{\lambda}{2}\underbrace{\sum_{j=1}^{n}\theta_j^2}_{B}$$

$$\text{For } C=\frac{1}{\lambda}: A+\lambda B\rightarrow CA+B$$

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}\mathrm{cost}_1(\theta^T\mathbf{x}^{(i)})+(1-y^{(i)})\mathrm{cost}_0(\theta^T\mathbf{x}^{(i)})\right]+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

SVM Cost Function (Hinge-Loss)

$$\min_{\theta} \boxed{C} \overbrace{\boxed{\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1(\theta^T\mathbf{x}^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T\mathbf{x}^{(i)})\right]}}^{A} + \overbrace{\boxed{\frac{1}{2}\sum_{j=1}^{n}\theta_j^2}}^{B}$$

When C is large enough, cost A will be close to 0, but we still left with cost B:

$$\min_{\theta} \overbrace{\boxed{\frac{1}{2}\sum_{j=1}^{n}\theta_j^2}}^{B}$$

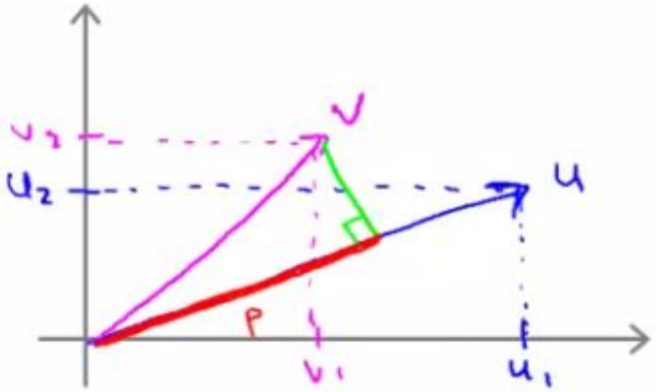
$$\text{s.t.} \quad \underbrace{\boxed{\begin{array}{l}\theta^T\mathbf{x}^{(i)} \geq 1 \text{ if } y^{(i)} = 1 \\ \theta^T\mathbf{x}^{(i)} \leq -1 \text{ if } y^{(i)} = 0.\end{array}}}_{A}$$

If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)
If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

Review: Vector Inner (Dot) Product

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$||u|| = \sqrt{u_1^2 + u_2^2} \in \mathbf{R}$$



$$\begin{aligned} u \cdot v &= ||v||.\,||u||.\cos(\theta) \\ &= ||v||\cos(\theta)||u|| \\ &= \mathrm{proj}_{\,v} u.\,||u|| \end{aligned}$$

$$\begin{aligned} u \cdot v &= u^T v = v^T u \\ &= \mathrm{proj}_{\,v} u \cdot ||u|| \\ &= u_1 v_1 + u_2 v_2 \end{aligned}$$

SVM Optimization Objective

To simplify, set: : $\theta_0 = 0, n = 2$

$$\min_\theta \boxed{\frac{1}{2}\sum_{j=1}^{n}\theta_j^2}$$

$$\boxed{\frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \frac{1}{2}||\theta||^2}$$

$$\text{s.t.} \quad \boxed{\theta^T \mathbf{x}^{(i)}} \geq 1 \text{ if } y^{(i)} = 1$$

$$\boxed{\theta^T \mathbf{x}^{(i)}} \leq -1 \text{ if } y^{(i)} = 0.$$

$$\boxed{\theta^T \mathbf{x}^{(i)} = \text{proj}_{\mathbf{x}^{(i)}}\theta . ||\theta|| = p^{(i)} . ||\theta||}$$

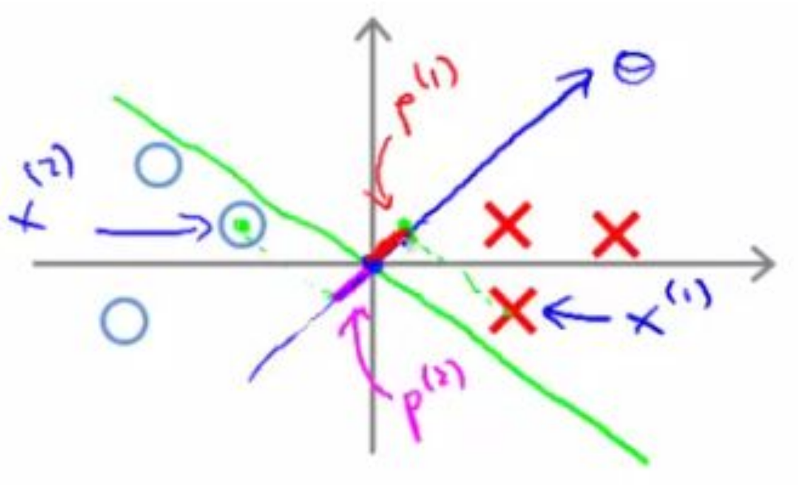
Rewrite as the optimization objective:

$$\min_\theta \boxed{\frac{1}{2}||\theta||^2}$$

$$\text{s.t.} \quad \boxed{p^{(i)} . ||\theta||} \geq 1 \text{ if } y^{(i)} = 1$$

$$\boxed{p^{(i)} . ||\theta||} \leq -1 \text{ if } y^{(i)} = 0.$$

SVM Decision Boundary



$$\min_{\theta} \frac{1}{2}||\theta||^2$$

$$\text{s.t.} \quad p^{(i)} . ||\theta|| \geq 1 \text{ if } y^{(i)} = 1$$

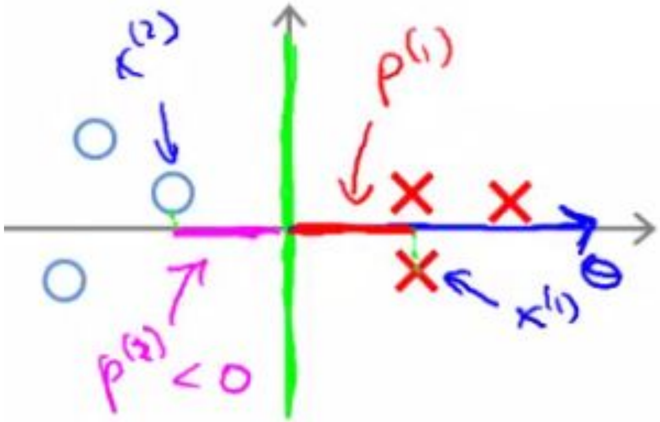
$$p^{(i)} . ||\theta|| \leq -1 \text{ if } y^{(i)} = 0.$$

We need to keep the constraint ≥ 1 (or ≤ -1) with the decision boundary

If $p^{(1)}$ is small, we need ||**θ**|| to be large, but the optimization objective is trying to make ||**θ**|| small.

***Images from Andrew Ng**

SVM Decision Boundary



$$\min_{\theta} \frac{1}{2} ||\theta||^2$$

$$\text{s.t.} \quad p^{(i)} . ||\theta|| \geq 1 \text{ if } y^{(i)} = 1$$

$$p^{(i)} . ||\theta|| \leq -1 \text{ if } y^{(i)} = 0.$$

With this decision boundary, $p^{(1)}$ becomes larger, so $||\boldsymbol{\theta}||$ can become smaller

This is why SVM prefers this hypothesis, and this is how we generate **large margin.**

So by maximizing these p values we minimize $||\boldsymbol{\theta}||$

Why θ is orthogonal to the decision boundary?

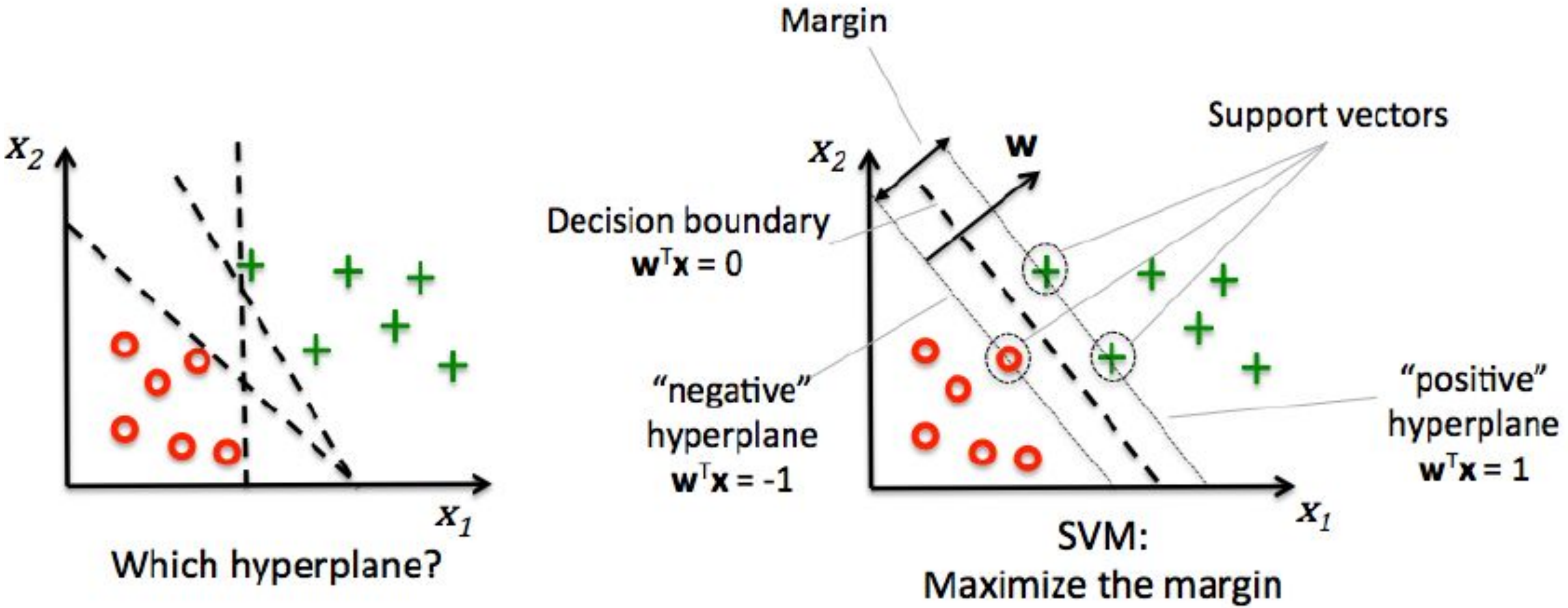
Let u and v be two points on the decision boundary, then:

$$\theta^T u = 0$$
$$\theta^T v = 0$$
$$\theta^T (u - v) = 0$$
$$\rightarrow \theta \perp (u - v)$$

What happen the decision boundary when $\theta_0 = 0$?

The decision boundary would have to pass through the origin which limits the capability of a classifier

Example



x_2
x_1
Which hyperplane?
Margin
Support vectors
x_2
w
Decision boundary
$\mathbf{w}^T\mathbf{x} = 0$
"negative"
hyperplane
$\mathbf{w}^T\mathbf{x} = -1$
"positive"
hyperplane
$\mathbf{w}^T\mathbf{x} = 1$
x_1
SVM:
Maximize the margin

Quadratic Programming

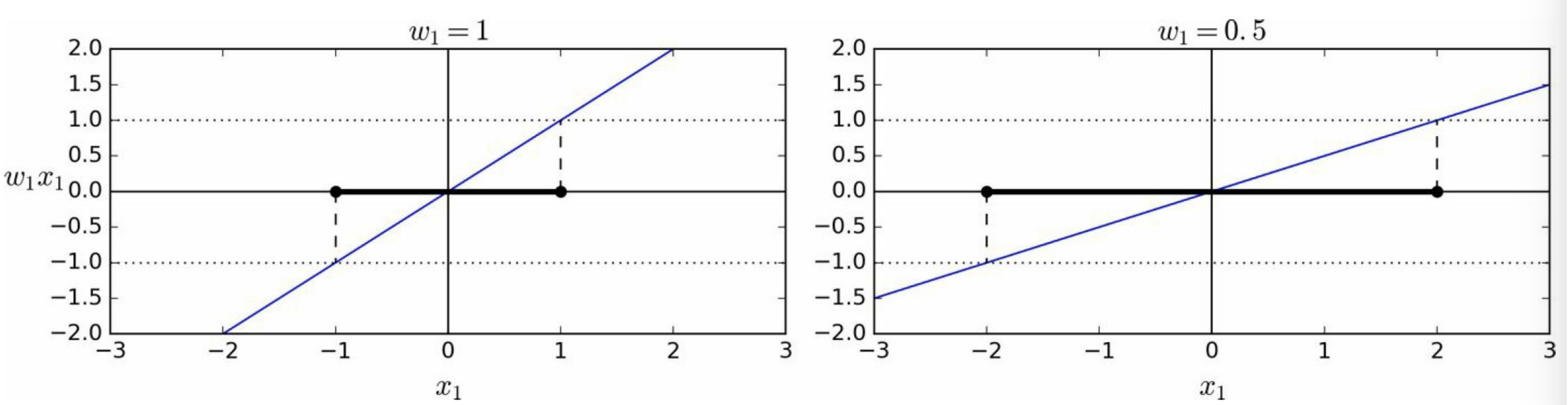
$$\begin{aligned}
&\underset{\boldsymbol{p}}{\text{Minimize}} && \frac{1}{2}\boldsymbol{p}^T \cdot \boldsymbol{H} \cdot \boldsymbol{p} \quad + \quad \boldsymbol{f}^T \cdot \boldsymbol{p} \\
&\text{subject to} && \boldsymbol{A} \cdot \boldsymbol{p} \leq \boldsymbol{b}
\end{aligned}$$

$$\text{where} \quad \begin{cases}
\boldsymbol{p} & \text{is an } n_p\text{-dimensional vector } (n_p = \text{number of parameters}), \\
\boldsymbol{H} & \text{is an } n_p \times n_p \text{ matrix}, \\
\boldsymbol{f} & \text{is an } n_p\text{-dimensional vector}, \\
\boldsymbol{A} & \text{is an } n_c \times n_p \text{ matrix } (n_c = \text{number of constraints}), \\
\boldsymbol{b} & \text{is an } n_c\text{-dimensional vector.}
\end{cases}$$

Alternative Explanation for Large Margin

The slope of the decision function is equal to the norm of the weight vector ||**w**||

The smaller the weight vector w, the larger the margin



So we want to minimize ||w|| to get a large margin