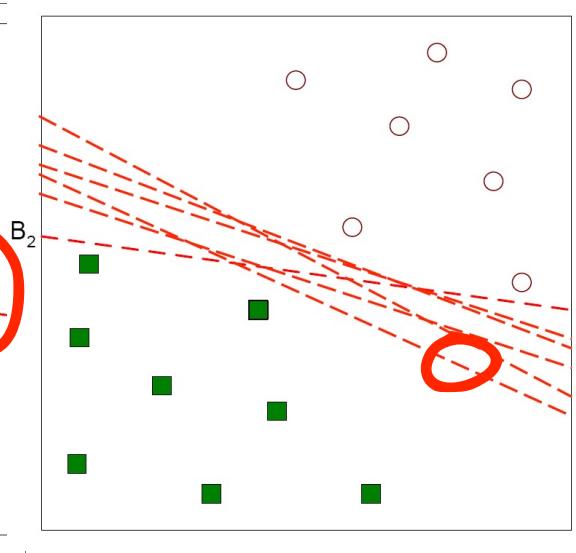
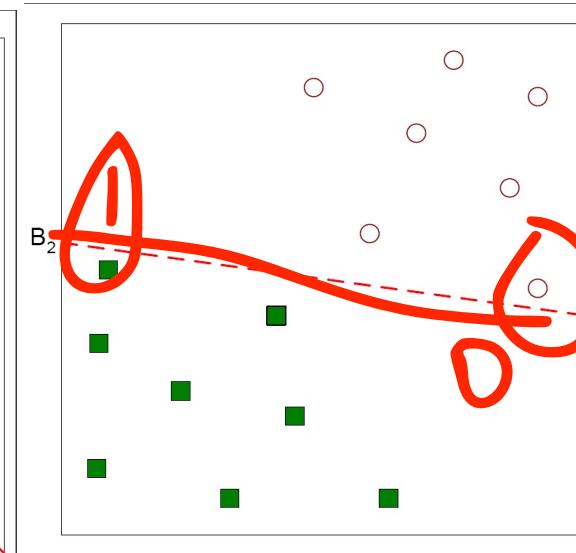
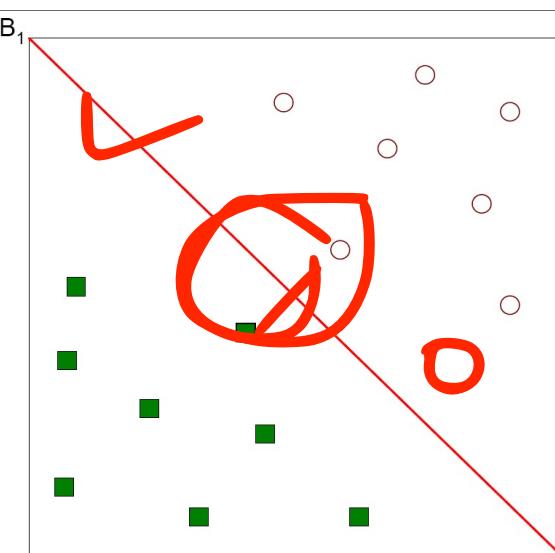
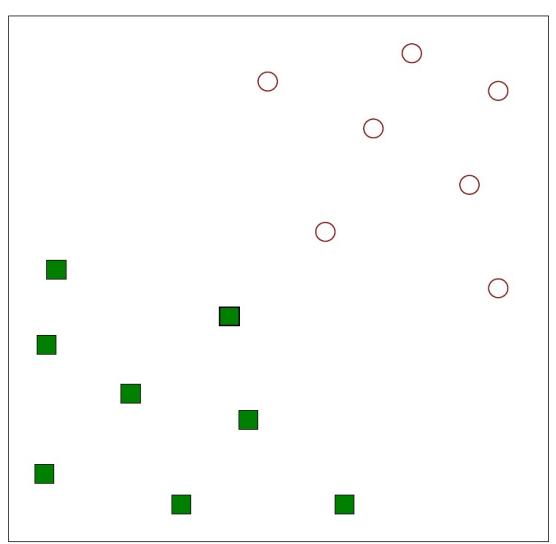


Lecture 8: Support Vector Machines (SVM), scikit-learn

Dr. Huiping Cao

Support Vector Machines - introduction

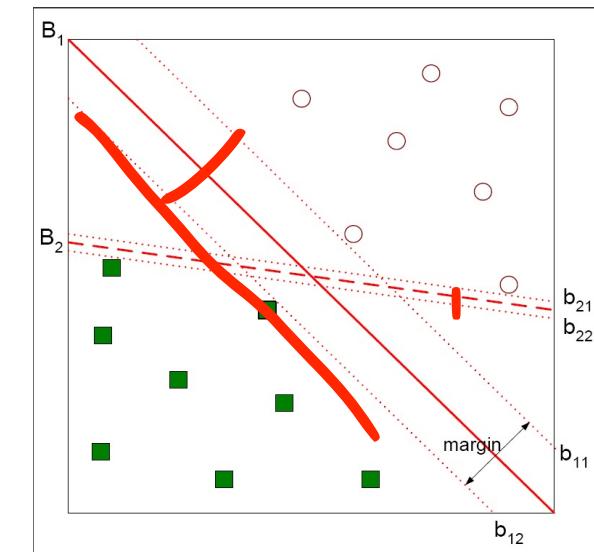
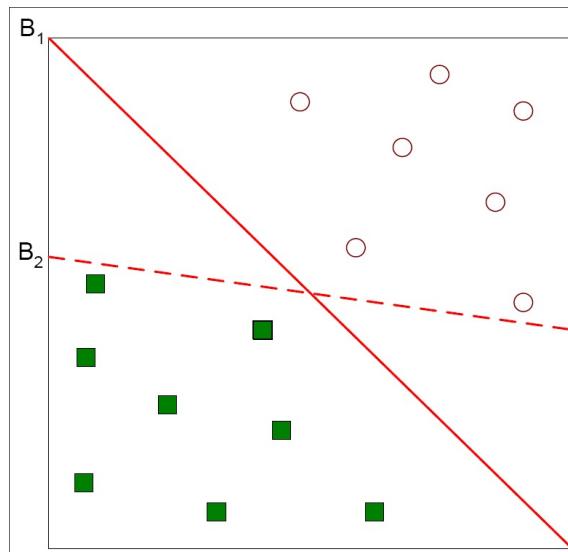
- SVM is an extension of perceptron. The perceptron model minimizes the misclassification error. SVM **maximizes the margin**.
- Find a linear hyperplane (decision boundary) that will separate the data.



Applied Machine

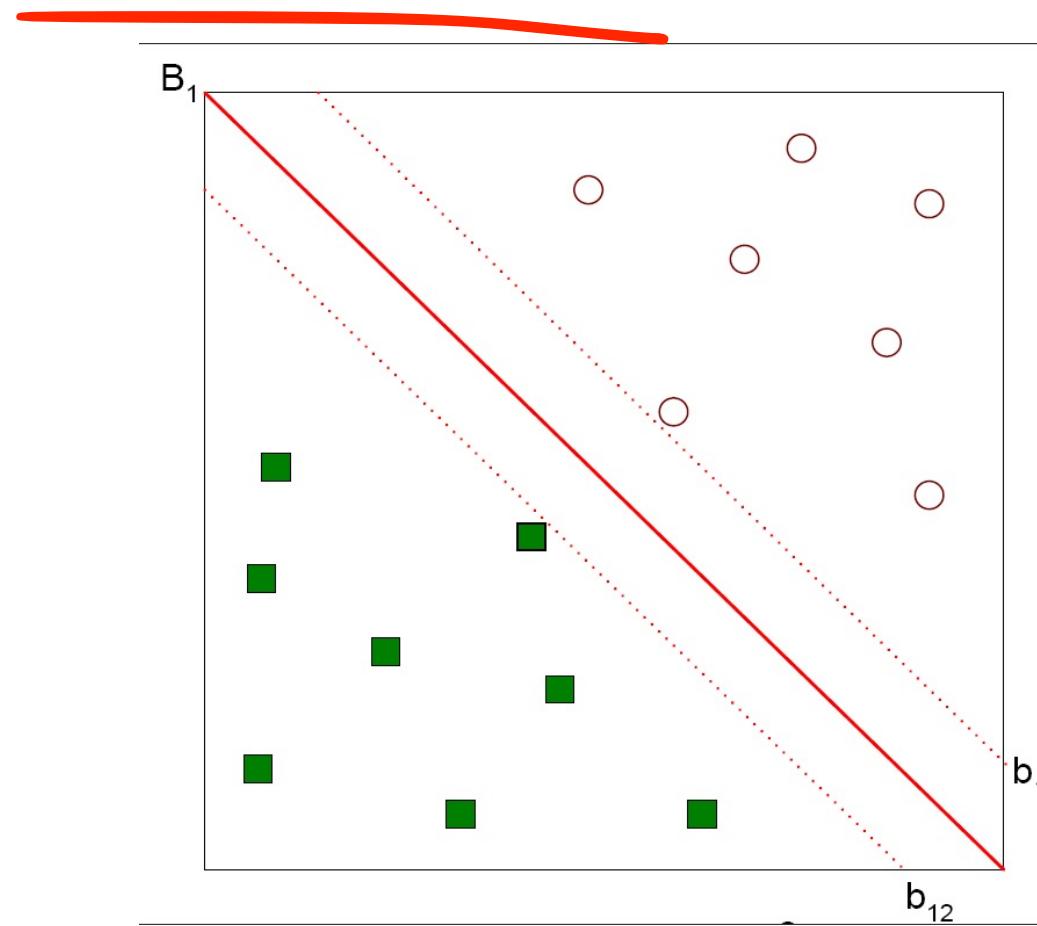
Support Vector Machines - idea

- Which one is better? B1 or B2?
- The bigger the margin, the better.
- Find hyperplane that maximizes the margin; so B1 is better than B2



Support Vector Machines - formulation

- How do we formulate B_1 and its margin, b_{11} , and b_{12} ?

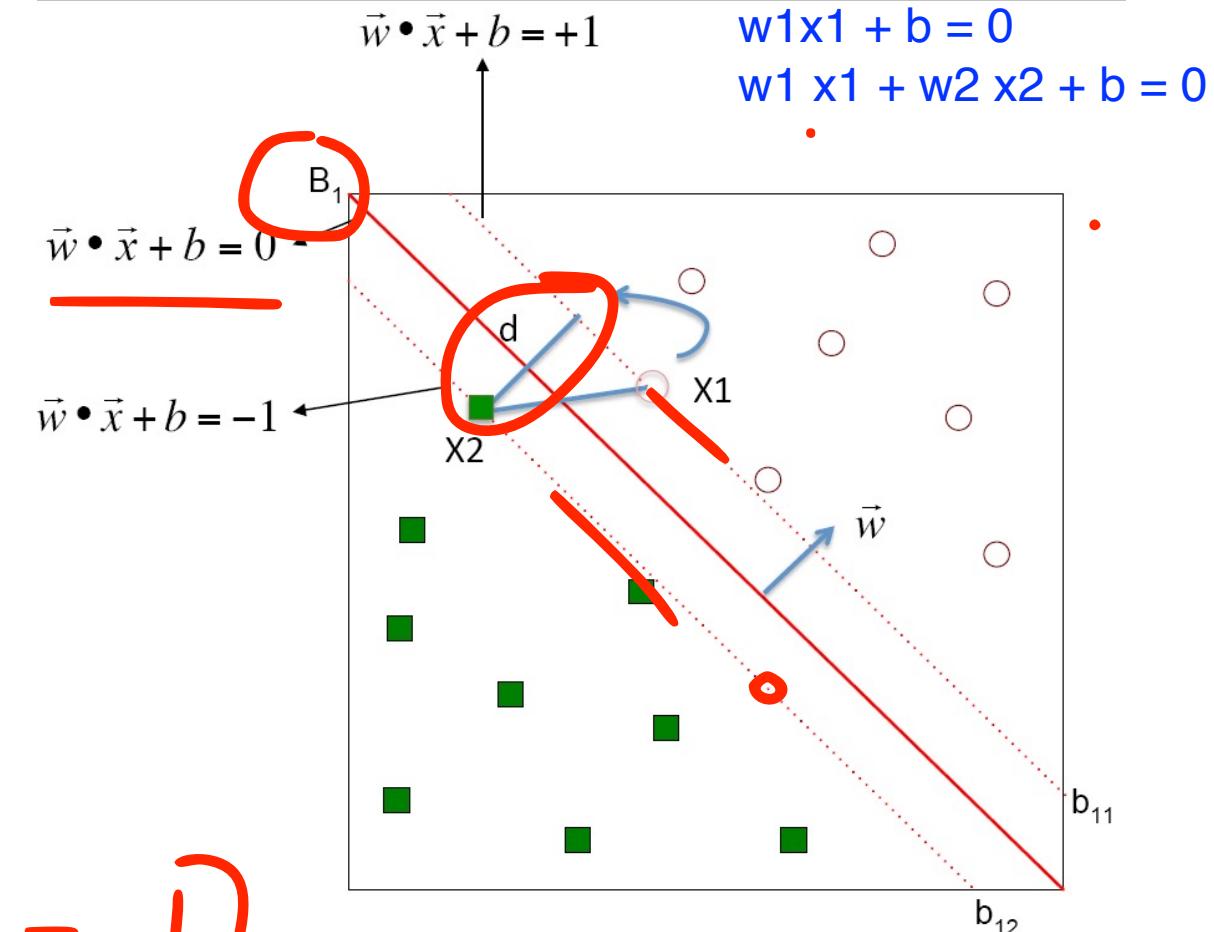


Support Vector Machines - Concepts

~~$\vec{x} + b$~~

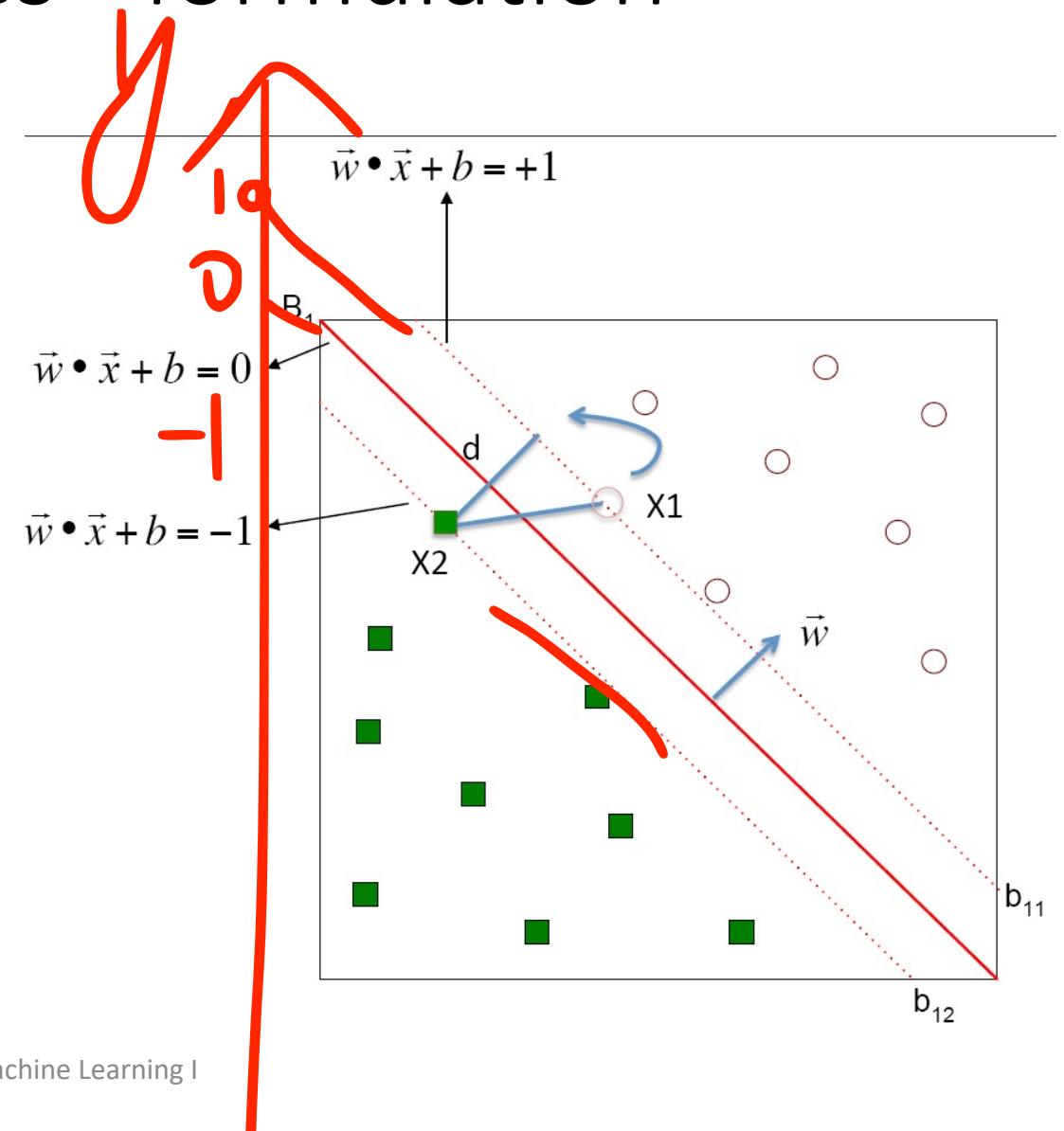
- **Decision boundary:** the separating hyperplane
- **Margin:** the distance between the decision boundary and the training samples that are closest to the decision boundaries
- **Support vectors:** the samples that are closest to the decision boundaries.

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



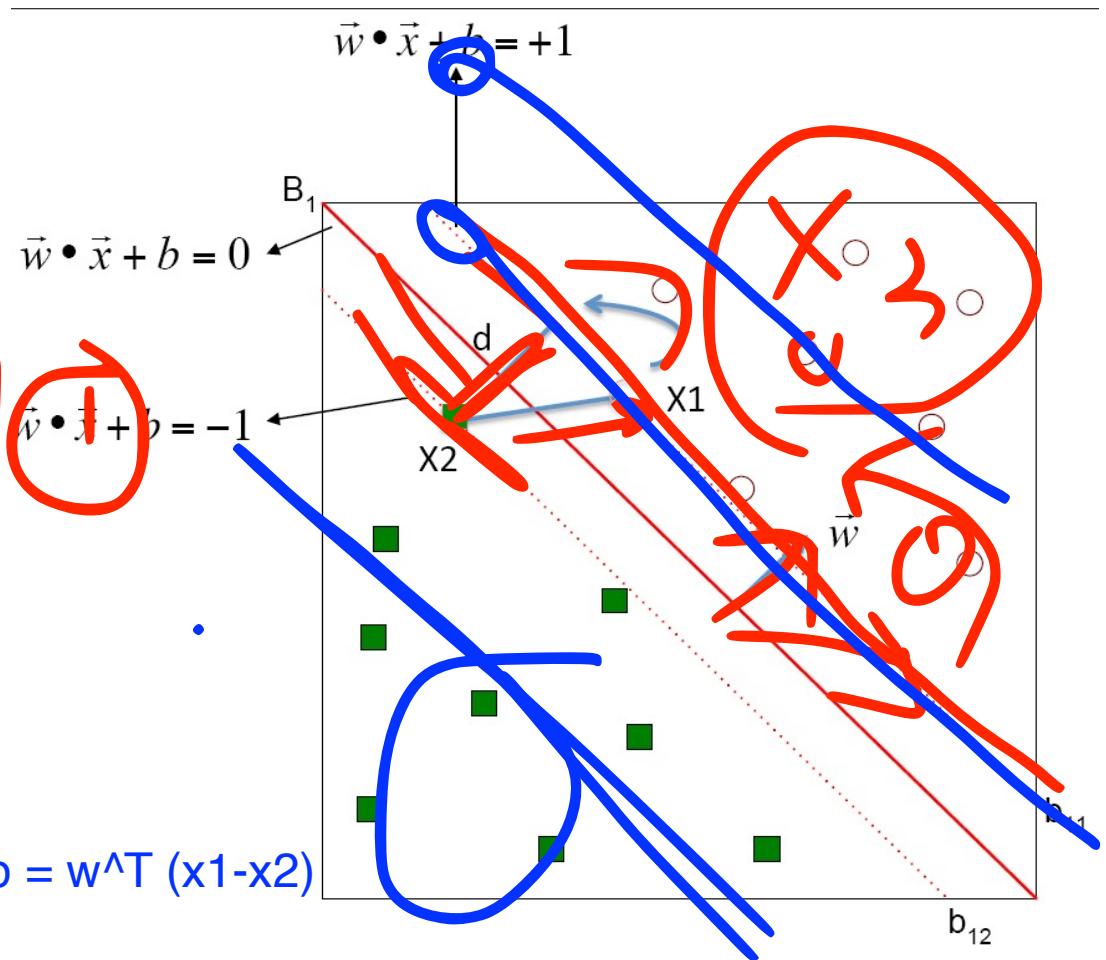
Support Vector Machines - formulation

- **Decision boundary**
 - $B_1: \underline{\mathbf{w}^\top \mathbf{x} + b = 0}$
- Hyperplanes defining margins
 - b_{11} (*positive hyperplane*)
 - $\mathbf{w}^\top \mathbf{x} + b = 1$
 - b_{12} (*negative hyperplane*)
 - $\mathbf{w}^\top \mathbf{x} + b = -1$



Margin of the decision boundaries

- Let d be the distance between the two hyperplanes b_{11} and b_{12}
 - The margin of the decision boundaries
 - $b_{11}: \mathbf{w}^\top \mathbf{x} + b = 1$
 - $b_{12}: \mathbf{w}^\top \mathbf{x} + b = -1$
 - Let x_1 be on b_{11} and x_2 be on b_{12} . Then, we have $\mathbf{w}^\top x_1 + b = 1$ and $\mathbf{w}^\top x_2 + b = -1$. Thus, $\mathbf{w}^\top(x_1 - x_2) = 2$
 - $\mathbf{w}^\top(x_1 - x_2) = \|\mathbf{w}\| \times \|(x_1 - x_2)\| \times \cos(\theta)$ where $\|\cdot\|$ is the norm of a vector
 - $\|\mathbf{w}\| \times \|(x_1 - x_2)\| \times \cos(\theta) = \|\mathbf{w}\| \times d$, where d is the length of vector $x_1 - x_2$ in the direction of vector \mathbf{w} .
 - $\|\mathbf{w}\| \times d = 2$
 - $d = \frac{2}{\|\mathbf{w}\|}$
- $(\mathbf{w}^\top x_1 + b) - (\mathbf{w}^\top x_2 + b) = \mathbf{w}^\top (x_1 - x_2) + b - b = \mathbf{w}^\top (x_1 - x_2)$
 $1 - (-1) = 2$



Learn a linear SVM Model

$$\bar{w}^\top \bar{x} + b$$

- The training phase of SVM is to estimate the parameters w and b .
- The parameters must follow two conditions

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

- These two conditions mean: all negative samples should fall on one side of the negative hyperplane, whereas all the positive samples should fall behind the positive hyperplane.

Formulate the problem

- We want to maximize the margin $d = \frac{2}{\|\mathbf{w}\|}$
 - This is equivalent to minimize $L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}$
 - Subjected to the following constraints:

$$y^{(i)} = \begin{cases} 1, & \text{if } \mathbf{w}^\top \mathbf{x}^{(i)} + b \geq 1 \\ -1, & \text{if } \mathbf{w}^\top \mathbf{x}^{(i)} + b \leq -1 \end{cases}$$

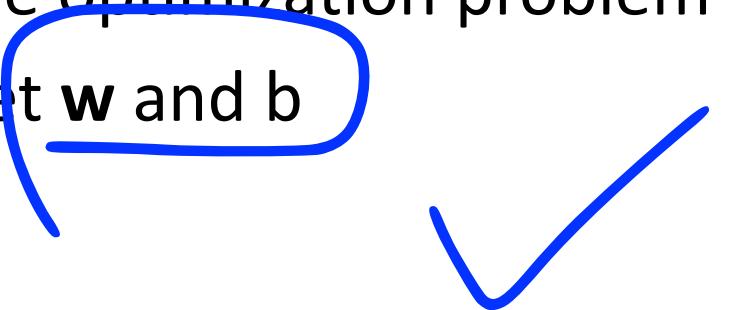
- which is equivalent to $y^{(i)} (\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1, i \dots, n$

Formulate the problem

- Constrained optimization problem.
 - Objective function: $\underset{2}{\text{Minimize}} \frac{\|\mathbf{w}\|^2}{2}$
 - subject to $y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1, i \dots, n$
- **Convex optimization problem**
 - Objective function is quadratic
 - Constraints are linear
 - Can be solved using the standard **Lagrange multiplier** method, which is skipped.

Get w and b

- Solving the optimization problem
- We can get w and b



Prediction

- Given a test data point \underline{z} , we can calculate
 - $y_z = \underline{\text{sign}(\mathbf{w}^\top \mathbf{z} + b)}$
- If $y_z = 1$, the test instance is classified as positive class
- If $y_z = -1$, the test instance is classified as negative class

Different implementations in scikit-learn

- SVC class uses the LIBSVM library (which is an equivalent C/C++ library)

1

```
from sklearn.svm import SVC  
  
svm = SVC(kernel='linear', C=1.0, random_state=1)  
svm.fit(X_train_std, y_train)
```



4

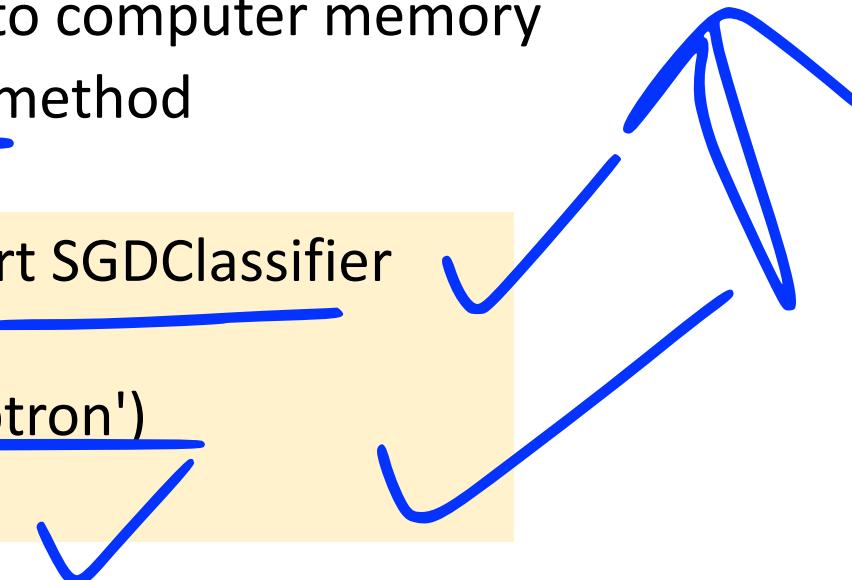


Different implementations in scikit-learn

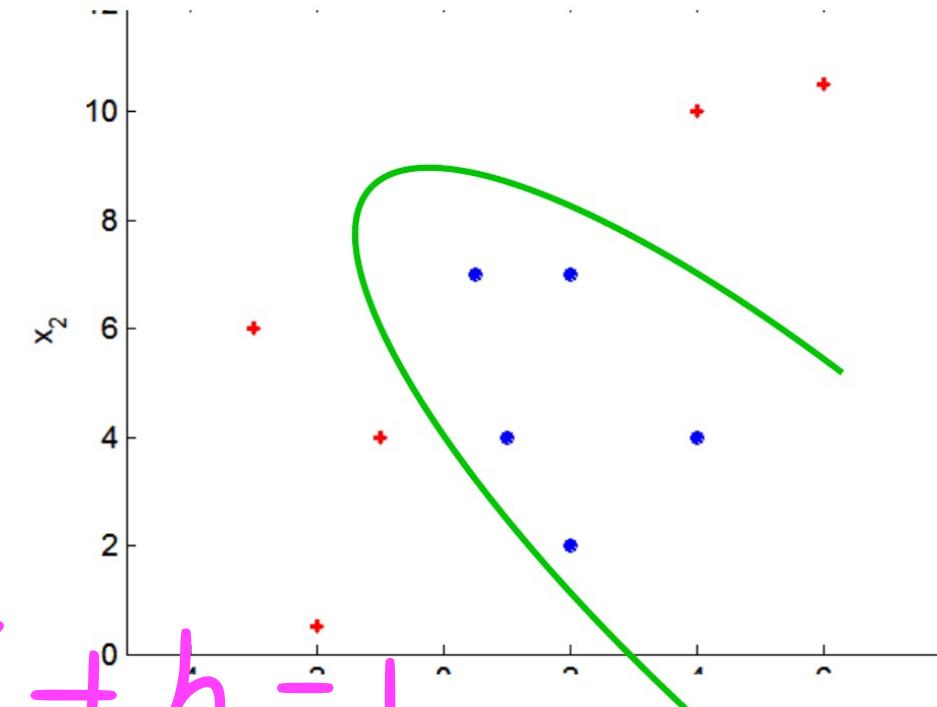
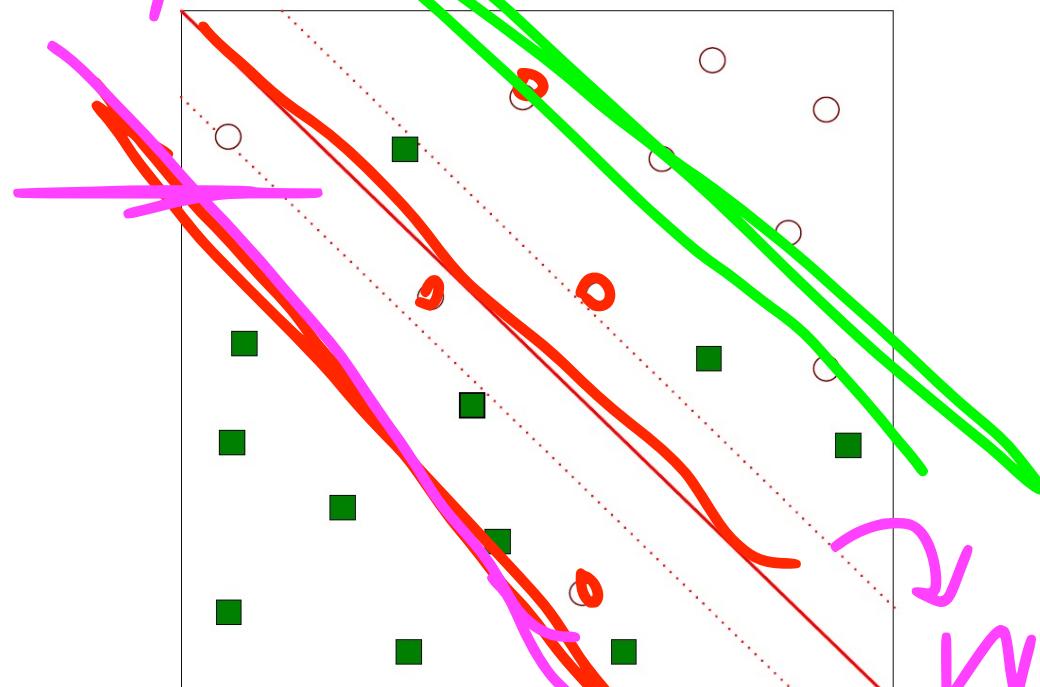
- **SGDClassifier** class

- It is a linear classifier (SVM, logistic regression) with SGD training.
- Datasets are too large to fit into computer memory
- Online learning via `partial_fit` method

```
from sklearn.linear_model import SGDClassifier  
  
ppn = SGDClassifier(loss='perceptron')  
svm = SGDClassifier(loss='hinge')
```



What is the problem if not linearly separable?



$$w^T x + b = 1$$
$$w^T x + b = -1$$

Dealing with a nonlinearly separable case using slack variables

- Introduce a slack variable ξ , to generate the so-called **soft-margin classification**.
- Subjected to the following constraints:

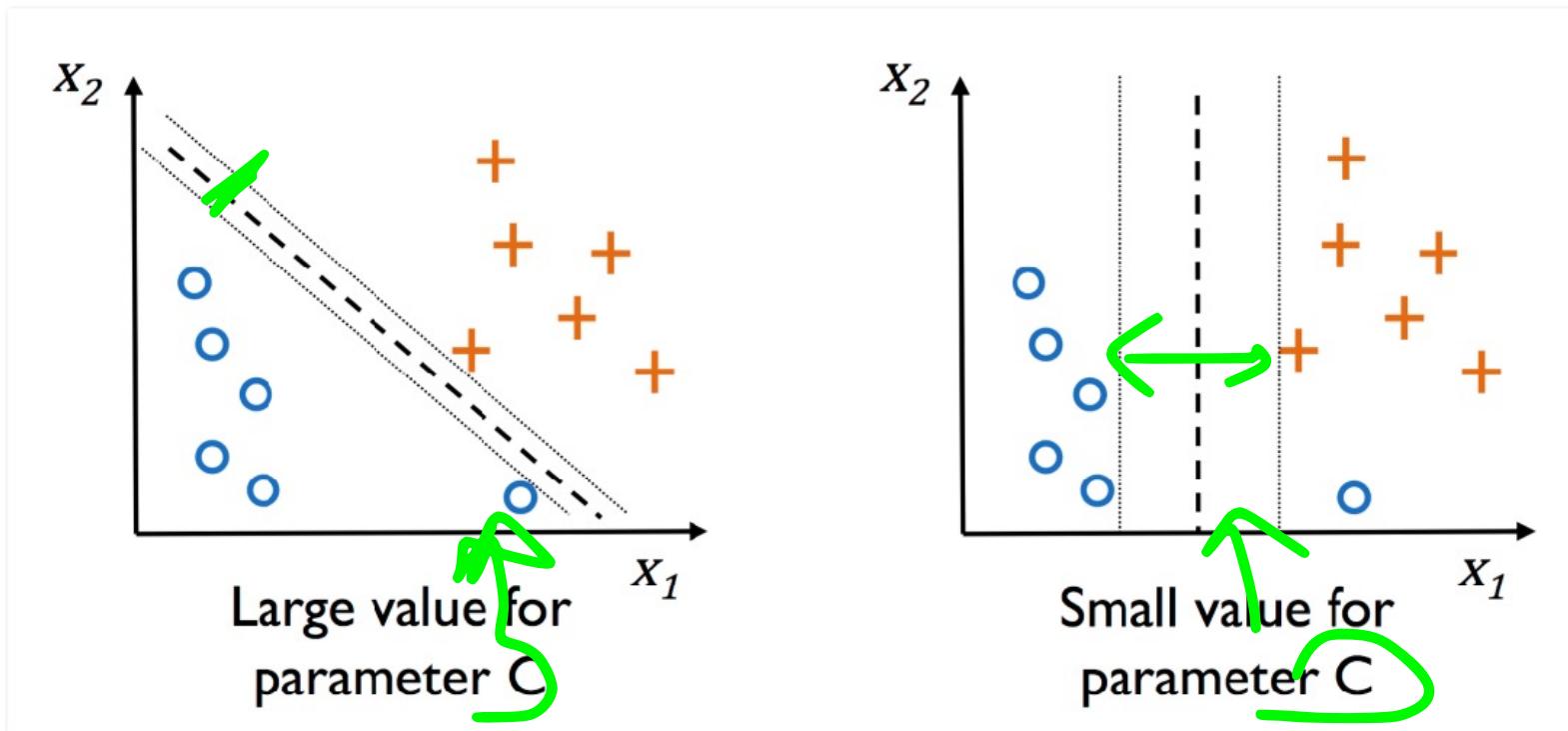
$$y^{(i)} = \begin{cases} 1, & \text{if } \mathbf{w}^\top \mathbf{x}^{(i)} + b \geq 1 - \xi^{(i)} \\ -1, & \text{if } \mathbf{w}^\top \mathbf{x}^{(i)} + b \leq -1 + \xi^{(i)} \end{cases}$$

- Objective function:

$$\text{Minimize } \frac{\|\mathbf{w}\|^2}{2} + C \left(\sum_{i=1}^N \xi^{(i)} \right)$$

Effect of C

- Control the penalty for misclassification
 - Larger value of C: for large error penalty
 - Smaller value of C: less strict about misclassification errors.



Kernel SVM: Use SVM to solve nonlinear problems

- SVM can be kernelized to solve nonlinear classification problem.
- Kernel SVM. The basic idea behind kernel methods is to create nonlinear combinations of the original features (X space) to project data points onto a higher-dimensional space (Z space) via a mapping function φ . In the higher-dimensional space, the data becomes linearly separable. We can train a linear SVM model to classify the data in the new space.
 - In particular, in the quadratic programming (QP) task, the SVM model replaces the dot product $(\mathbf{x}^{(i)})^\top \mathbf{x}^{(j)}$ with $\varphi(\mathbf{x}^{(i)})^\top \varphi(\mathbf{x}^{(j)})$.
 - Support vectors live in Z space

$$\mathbf{z}_i^\top \mathbf{z}_j$$

Kernel SVM

- During prediction, we use the same mapping function to transform the new data point to the new space and classify it using the linear SVM model.
 - E.g., $\phi(x_1, x_2) = (z_1, z_2, z_3) = \underline{(x_1, x_2, x_1^2 + x_2^2)}$
- **Issue with the mapping approach:** It is relatively expensive to calculate the new features, especially for high-dimensional data.

Kernel trick

- Kernel trick is to address this issue.
- Define a kernel function $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^\top \phi(\mathbf{x}^{(j)})$
- Radial Basis Function (RBF) kernel (also called Gaussian kernel)
 - $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}|^2}{2\sigma^2}\right)$
 - Let $\gamma = \frac{1}{2\sigma^2}$, this kernel function is represented as $\sum_{i=0}^{\infty} \dots$
 - $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma |\mathbf{x}^{(i)} - \mathbf{x}^{(j)}|^2)$
- RBF kernel is formed by taking an infinite sum over polynomial kernels ([proof](#))

Explanation

$$\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma |\mathbf{x}^{(i)} - \mathbf{x}^{(j)}|^2)$$

- $|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}|^2$ measures distance, then $-\gamma |\mathbf{x}^{(i)} - \mathbf{x}^{(j)}|^2$ measures similarity
- Due to the exponential term, the similarity score is in the range of 0 (dissimilar) to 1 (similar).
- The extreme case is $\mathbf{x}^{(i)} = \mathbf{x}^{(j)}$, where $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(0) = 1$
- When the distance is bigger, $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is smaller

Prediction

- In kernel SVM, we map the data points into a possibly infinite dimension Hilbert space Z . It turns out that w has the form

$$w = \sum_{i=1}^n \lambda_i y_i \varphi(x^{(i)})$$

- where $\varphi(x^{(i)})$ is the mapped data point in the feature space
- the λ_i s are from the optimization solver (and if $x^{(i)}$ is not a support vector then $\lambda_i=0$).
- One of the things that makes SVM practical is that w is a finite linear combination of $\varphi(x^{(i)})$.

Prediction

- We classify a new data point \mathbf{x} by determining which side of the hyperplane x is on.
- Given the kernel function K , the calculation:

$$\begin{aligned} \text{sign}(b + \mathbf{w}^T \mathbf{x}) &= \text{sign} \left(b + \left(\sum_{i=1}^n \lambda_i y_i \varphi(x^{(i)}) \right)^T \varphi(x) \right) \\ &= \text{sign} \left(b + \left(\sum_{i=1}^n \lambda_i y_i \varphi(x^{(i)})^T \right) \varphi(x) \right) \\ &= \text{sign} \left(b + \sum_{i=1}^n \lambda_i y_i K(x^{(i)}, x) \right) \end{aligned}$$

Discussions

- Note that we never actually need to calculate any $\phi(x)$. We instead only need inner products which are provided by K . There is no way around computing this, but for certain kernels there may be tricks to compute it faster.
- Time calculation: the prediction of one data point needs to use all consuming the training instances.

Scikit-learn implementation

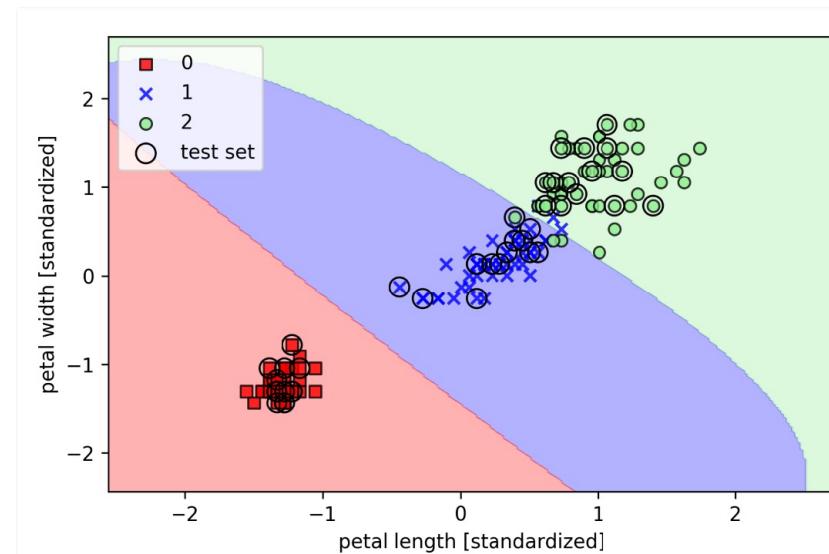
- Parameter **kernel='rbf'** to represent the radial basis function kernel
 - Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. If none is given, ‘rbf’ will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).
- Parameter **gamma**: kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’.

```
from sklearn.svm import SVC
```

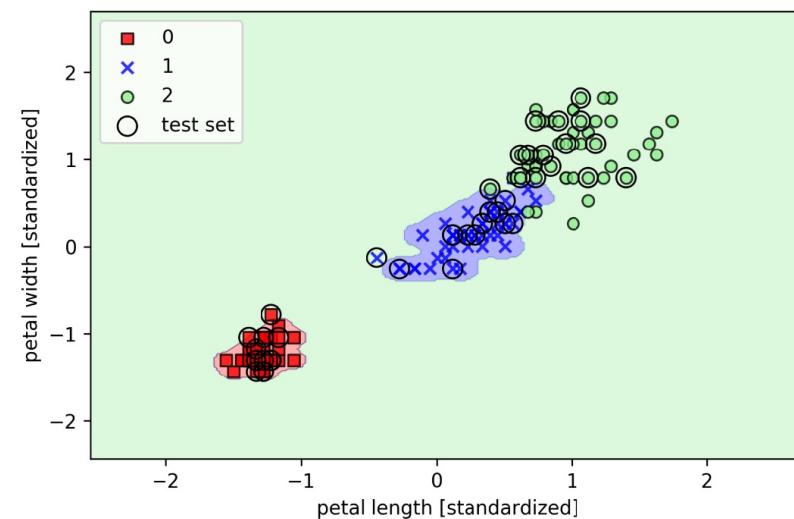
```
svm = SVC(kernel='rbf', random_state=1, gamma=0.10, C=10.0)
svm.fit(X, y)
```

γ parameter

- γ parameter is treated as a cut-off parameter. The higher value leads to a tighter and bumpier decision boundary. A smaller value leads to a relatively soft boundary.



$\gamma = 0.2$



$\gamma = 100.0$