# ECE595 / STAT598: Machine Learning I

## Lecture 21.1: Support Vector Machine - Soft SVM

Spring 2020

Stanley Chan

School of Electrical and Computer Engineering
Purdue University

PURDUE
UNIVERSITY

# Outline

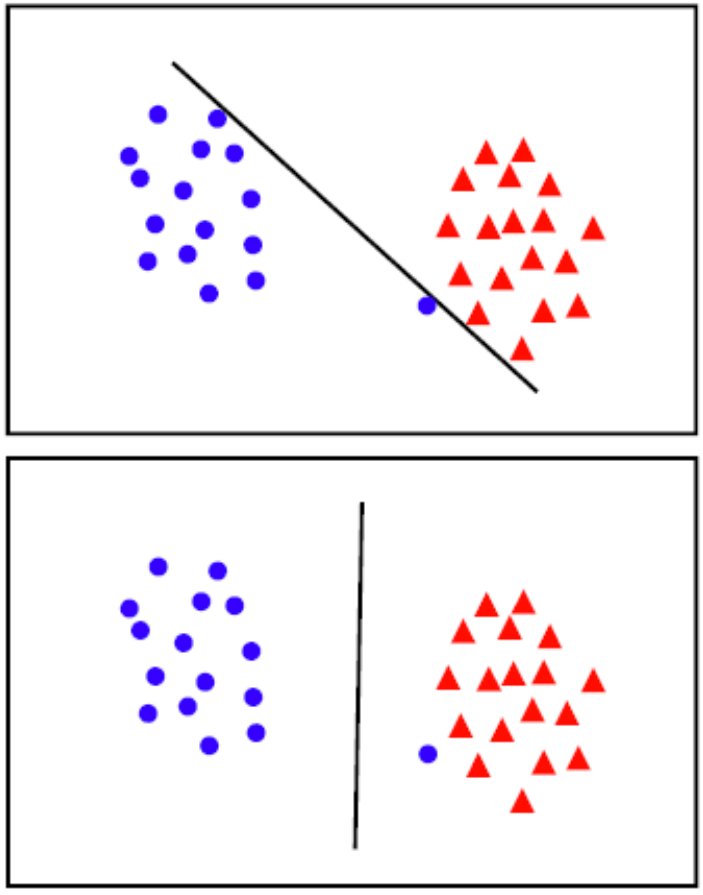
Support Vector Machine

- Lecture 19 SVM 1: The Concept of Max-Margin
- Lecture 20 SVM 2: Dual SVM
- Lecture 21 SVM 3: Soft SVM and Kernel SVM

**This lecture: Support Vector Machine: Soft and Kernel**

- Soft SVM
  - Motivation
  - Formulation
  - Interpretation
- Kernel Trick
  - Nonlinearity
  - Dual Form
  - Kernel SVM

# Linearly Not Separable



- the points can be linearly separated but there is a very narrow margin

- but possibly the large margin solution is better, even though one constraint is violated

http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

# Soft Margin

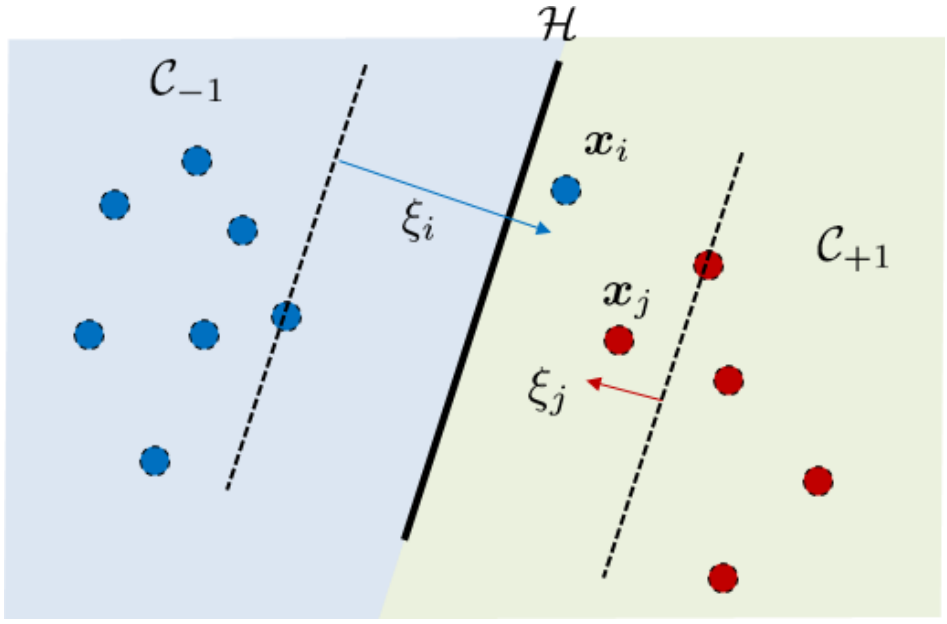
- We want to allow data points to stay inside the margin.
- How about change

$$y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0) \geq 1$$

to this one:

$$y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0) \geq 1 - \xi_j, \quad \text{and} \quad \xi_j \geq 0.$$

- If $\xi_j > 1$, then $\boldsymbol{x}_j$ will be misclassified.



# Soft Margin

- We can consider this problem

$$\begin{aligned} \underset{\boldsymbol{w}, w_0, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{subject to} \quad & y_j(\boldsymbol{w}^T \boldsymbol{x}_j + w_0) \geq 1 - \xi_j, \\ & \xi_j \geq 0, \quad \text{for} \quad j = 1, \ldots, n, \end{aligned}$$

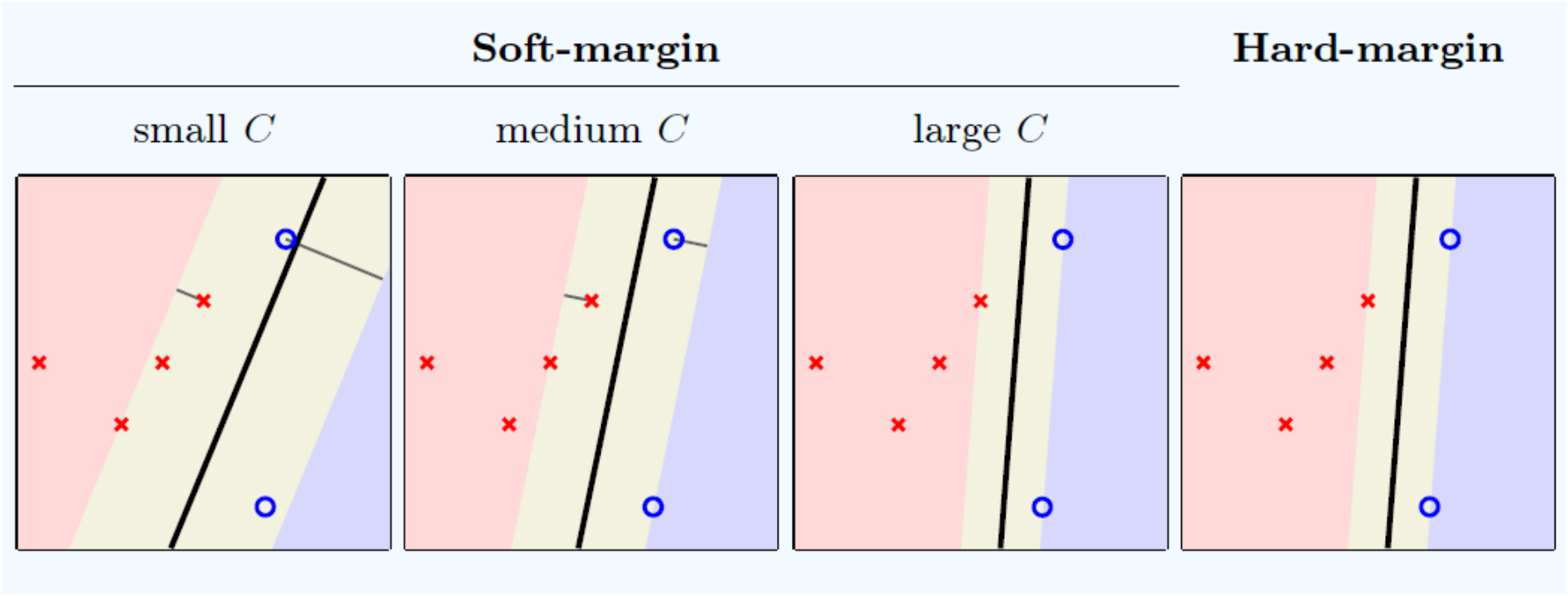
- But we need to control $\boldsymbol{\xi}$, for otherwise the solution will be $\boldsymbol{\xi} = \infty$.
- How about this:

$$\begin{aligned} \underset{\boldsymbol{w}, w_0, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\|\boldsymbol{\xi}\|^2 \\ \text{subject to} \quad & y_j(\boldsymbol{w}^T \boldsymbol{x}_j + w_0) \geq 1 - \xi_j, \\ & \xi_j \geq 0, \quad \text{for} \quad j = 1, \ldots, n, \end{aligned}$$

- Control the energy of $\boldsymbol{\xi}$.

# Role of $C$

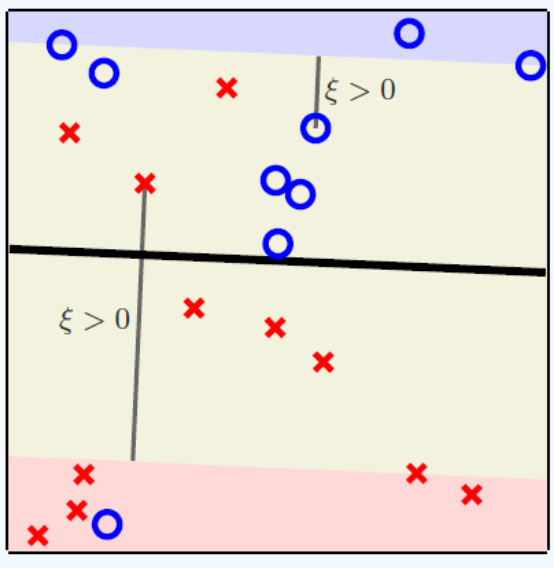
- If $C$ is big, then we enforce $\xi$ to be small.
- If $C$ is small, then $\xi$ can be big.



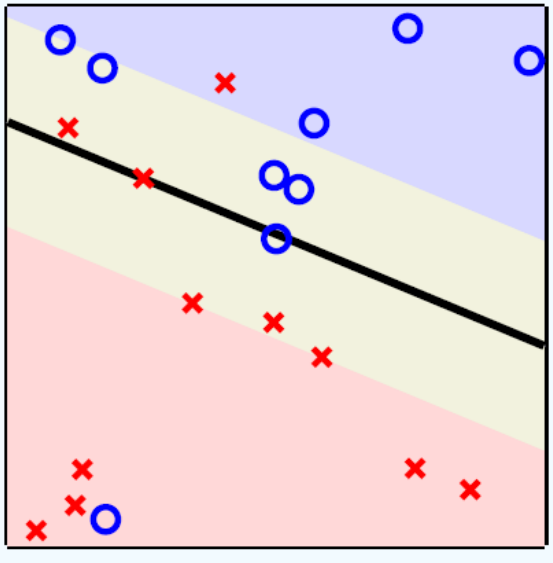
# No Misclassification?

- You can have misclassification in soft SVM
- $\xi_j$ can be big for a few outliers

$$\begin{aligned} \underset{\boldsymbol{w}, w_0, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\|\boldsymbol{\xi}\|^2 \\ \text{subject to} \quad & y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0) \geq 1 - \xi_j, \\ & \xi_j \geq 0, \quad \text{for} \quad j = 1, \ldots, N. \end{aligned}$$




(a) $C = 1$



(b) $C = 500$

# L1 Regularization

- Instead of $\ell_1$-norm, you can also do

$$\begin{aligned} \underset{\boldsymbol{w}, w_0, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\|\boldsymbol{\xi}\|_1 \\ \text{subject to} \quad & y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0) \geq 1 - \xi_j, \\ & \xi_j \geq 0, \quad \text{for} \quad j = 1, \ldots, N. \end{aligned}$$

- This enforces $\boldsymbol{\xi}$ to be sparse.
- Only a few entries samples are allowed to live in the margin.
- The problem remains convex.
- So you can still use CVX to solve the problem.

# Connection with Perceptron Algorithm

- In soft-margin SVM, $\xi_j \geq 0$ and $y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0) \geq 1 - \xi_j$ imply that

$$\xi_j \geq 0, \qquad \text{and} \qquad \xi_j \geq 1 - y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0).$$

- We can combine them to get

$$\begin{aligned}\xi_j &\geq \max\Big\{0,\ \ 1 - y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0)\Big\} \\ &= \Big[1 - y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0)\Big]_+\end{aligned}$$

- So if we use SVM with $\ell_1$ penalty, then

$$\begin{aligned}J(\boldsymbol{w}, w_0, \boldsymbol{\xi}) &= \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{j=1}^N \xi_j \\ &= \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{j=1}^N \Big[1 - y_j(\boldsymbol{w}^T\boldsymbol{x}_j + w_0)\Big]_+\end{aligned}$$

# Connection with Perceptron Algorithm

- This means that the training loss is

$$J(\boldsymbol{w}, w_0) = \sum_{j=1}^{N} \Big[1 - y_j(\boldsymbol{w}^T \boldsymbol{x}_j + w_0)\Big]_+ + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2,$$

  if we define $\lambda = 1/C$.
- Now, you can make $\lambda \to 0$. This means $C \to \infty$
- Then,

$$\begin{aligned} J(\boldsymbol{w}, w_0) &= \sum_{j=1}^{N} \Big[1 - y_j(\boldsymbol{w}^T \boldsymbol{x}_j + w_0)\Big]_+ \\ &= \sum_{j=1}^{N} \max\Big\{0, \;\; 1 - y_j(\boldsymbol{w}^T \boldsymbol{x}_j + w_0)\Big\} \\ &= \sum_{j=1}^{N} \max\{0, \;\; 1 - y_j g(\boldsymbol{x}_j)\} \end{aligned}$$

# Connection with Perceptron Algorithm

- **SVM Loss**:

$$J(\boldsymbol{w}, w_0) = \sum_{j=1}^{N} \max\{0, \ \ 1 - y_j g(\boldsymbol{x}_j)\}$$

- **Perceptron Loss**:

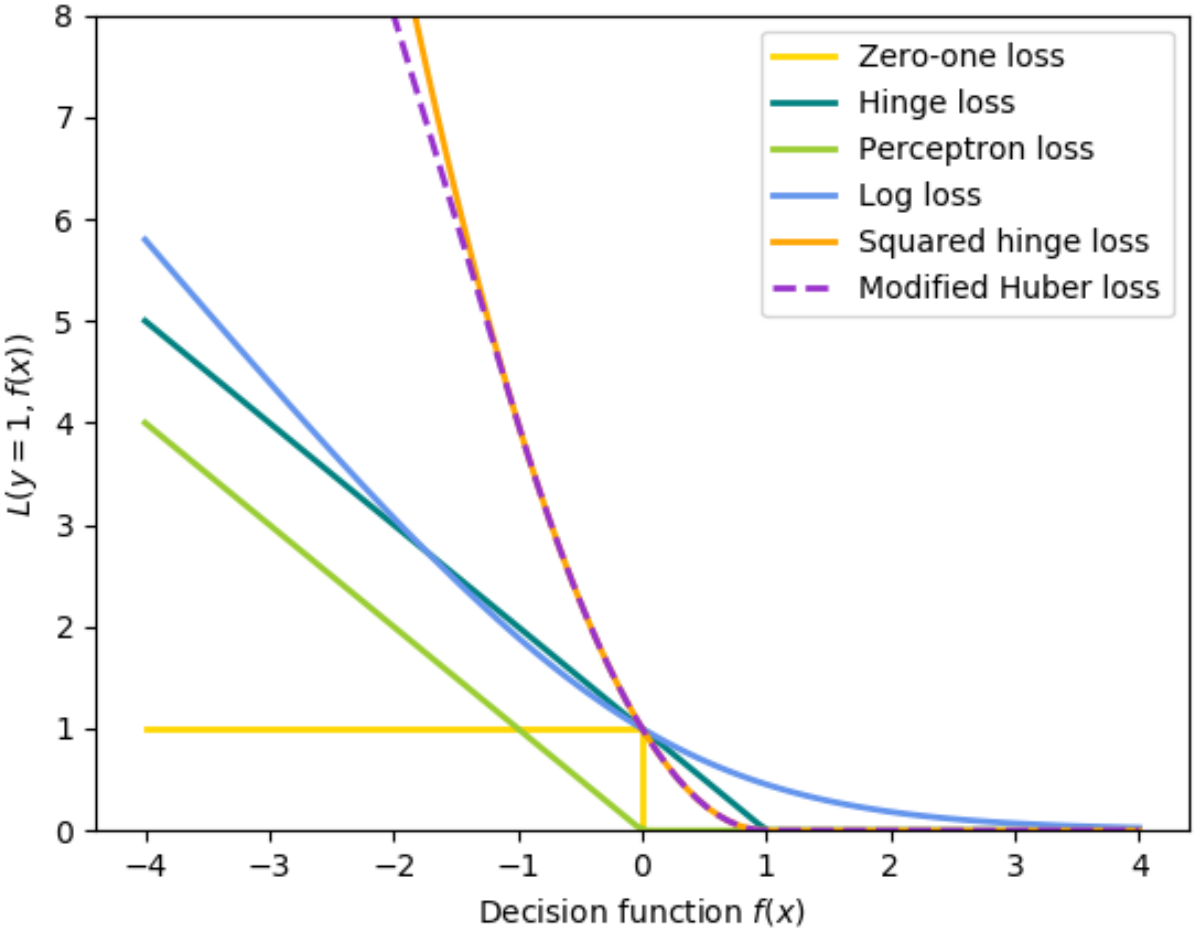
$$J(\boldsymbol{w}, w_0) = \sum_{j=1}^{N} \max\{0, \ \ -y_j g(\boldsymbol{x}_j)\}$$

- Therefore: SVM generalizes perceptron by allowing

$$J(\boldsymbol{w}, w_0) = \sum_{j=1}^{N} \max\{0, \ \ 1 - y_j g(\boldsymbol{x}_j)\} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2.$$

- $\|\boldsymbol{w}\|_2^2$ regularizes the solution.

# Comparing Loss functions




https://scikit-learn.org/dev/auto_examples/linear_model/plot_sgd_loss_functions.html