ECE595 / STAT598: Machine Learning I
Lecture 21 Support Vector Machine: Soft & Kernel

Spring 2020

Stanley Chan

School of Electrical and Computer Engineering
Purdue University

Outline

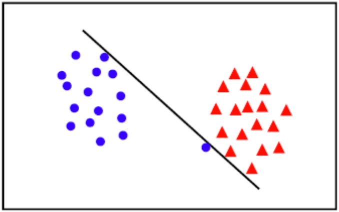
Support Vector Machine

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

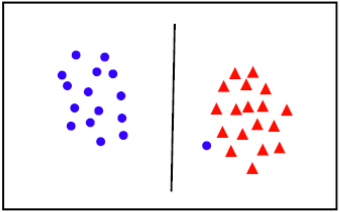
This lecture: Support Vector Machine: Soft and Kernel

- 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

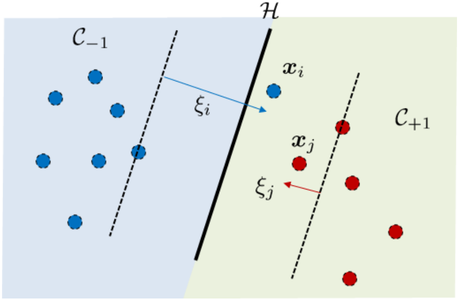
- We want to allow data points to stay inside the margin.
- How about change

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

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

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

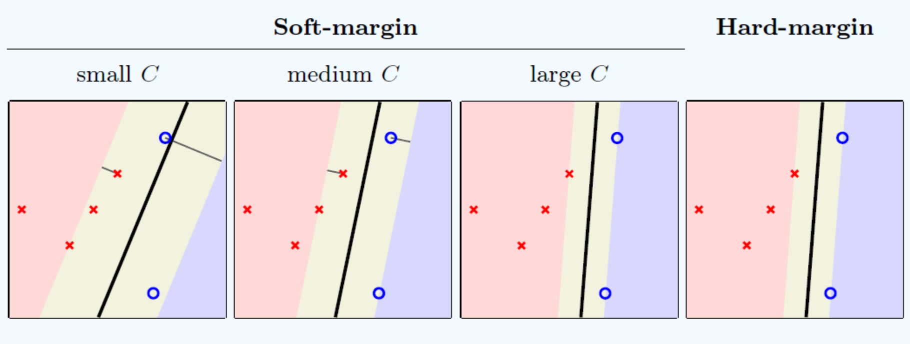
- But we need to control $\boldsymbol{\xi}$, for otherwise the solution will be $\boldsymbol{\xi} = \infty$.
- How about this:

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

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

Role of C

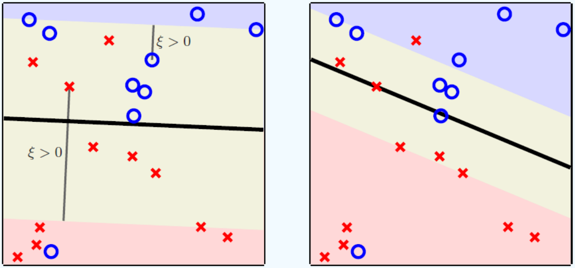
- If C is big, then we enforce ξ to be small.
- If C is small, then ξ can be big.



No Misclassification?

- You can have misclassification in soft SVM
- ξ_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 ℓ_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

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

- So if we use SVM with ℓ_1 penalty, then

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

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**:

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

- **Perceptron Loss**:

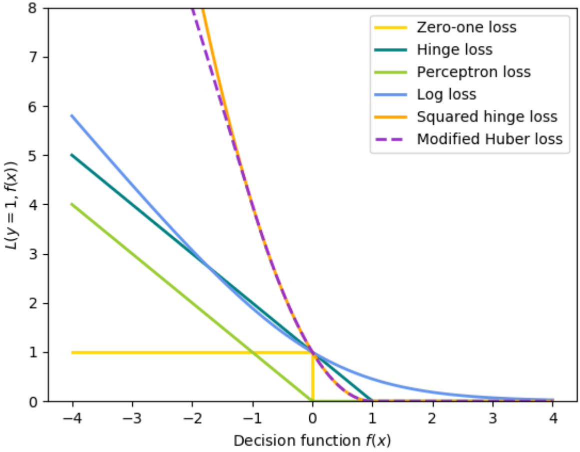
$$J(\boldsymbol{w}, w_0) = \sum_{j=1}^{N} \max\left\{0,\ \ -y_j g(\boldsymbol{x}_j)\right\}$$

- Therefore: SVM generalizes perceptron by allowing

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

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

Comparing Loss functions



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

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

The Kernel Trick

- A trick to turn linear classifier to nonlinear classifier.
- Dual SVM

$$\underset{\boldsymbol{\lambda} \geq 0}{\text{maximize}} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_{j=1}^{n}\lambda_j$$

$$\text{subject to} \quad \sum_{j=1}^{n}\lambda_j y_j = 0.$$

- Kernel Trick

$$\underset{\boldsymbol{\lambda} \geq 0}{\text{maximize}} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j) + \sum_{j=1}^{n}\lambda_j$$

$$\text{subject to} \quad \sum_{j=1}^{n}\lambda_j y_j = 0.$$

- You have to do this in dual. Primal is hard. See next slide.

The Kernel Trick

- Define

$$K(\boldsymbol{x}_i, vx_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j).$$

- The matrix $\boldsymbol{Q}$ is

$$\boldsymbol{Q} = \begin{bmatrix} y_1 y_1 \boldsymbol{x}_1^T \boldsymbol{x}_1 & \dots & y_1 y_N \boldsymbol{x}_1^T \boldsymbol{x}_N \\ y_2 y_1 \boldsymbol{x}_2^T \boldsymbol{x}_1 & \dots & y_2 y_N \boldsymbol{x}_2^T \boldsymbol{x}_N \\ \vdots & \vdots & \vdots \\ y_N y_1 \boldsymbol{x}_N^T \boldsymbol{x}_1 & \dots & y_N y_N \boldsymbol{x}_N^T \boldsymbol{x}_N \end{bmatrix}$$

- By Kernel Trick:

$$\boldsymbol{Q} = \begin{bmatrix} y_1 y_1 K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & y_1 y_N K(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ y_2 y_1 K(\boldsymbol{x}_2, \boldsymbol{x}_1) & \dots & y_2 y_N K(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ y_N y_1 K(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & y_N y_N K(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

Kernel

- The inner product $\Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$ is called a **kernel**

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j).$$

- Second-Order Polynomial kernel

$$K(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^T \boldsymbol{v})^2.$$

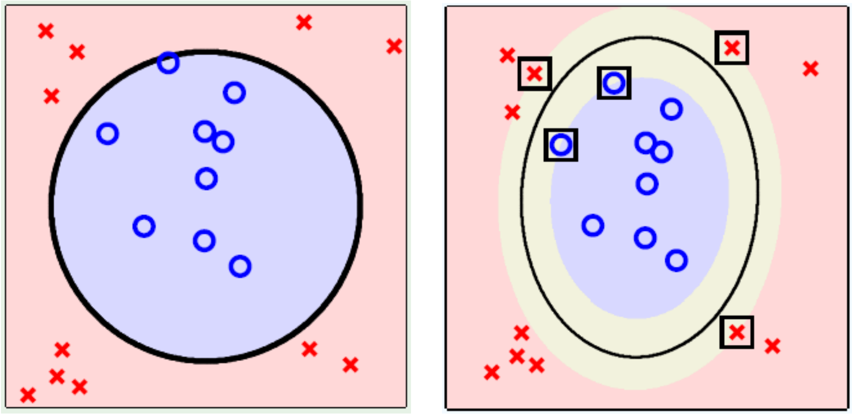
- Degree-Q Polynomial kernel

$$K(\boldsymbol{u}, \boldsymbol{v}) = (\gamma \boldsymbol{u}^T \boldsymbol{v} + c)^Q.$$

- Gaussian Radial Basis Function (RBF) Kernel

$$K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left\{ -\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{2\sigma^2} \right\}.$$

SVM with Second Order Kernel

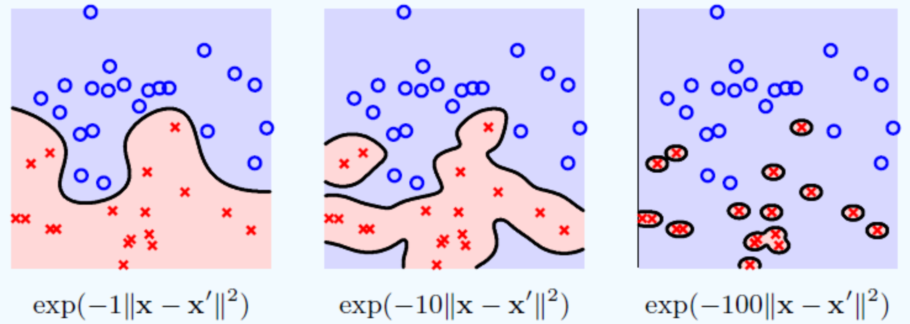


Boxed samples = Support vectors.

Radial Basis Function

Radial Basis Function takes the form of

$$K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left\{-\gamma\|\boldsymbol{u} - \boldsymbol{v}\|^2\right\}.$$



- Typical $\gamma \in [0, 1]$.
- γ too big: Over-fit.

Non-Linear Transform for RBF?

- Let us consider scalar $u \in \mathbb{R}$.

$$
\begin{aligned}
K(u,v) &= \exp\{-(u-v)^2\} \\
&= \exp\{-u^2\}\exp\{2uv\}\exp\{-v^2\} \\
&= \exp\{-u^2\}\left(\sum_{k=0}^{\infty}\frac{2^k u^k v^k}{k!}\right)\exp\{-v^2\} \\
&= \exp\{-u^2\}\left(1, \sqrt{\frac{2^1}{1!}}u, \sqrt{\frac{2^2}{2!}}u^2, \sqrt{\frac{2^3}{3!}}u^3, \ldots, \right)^T \\
&\qquad \times \left(1, \sqrt{\frac{2^1}{1!}}v, \sqrt{\frac{2^2}{2!}}v^2, \sqrt{\frac{2^3}{3!}}v^3, \ldots, \right)\exp\{-v^2\}
\end{aligned}
$$

- So Φ is

$$
\Phi(x) = \exp\{-x^2\}\left(1, \sqrt{\frac{2^1}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \sqrt{\frac{2^3}{3!}}x^3, \ldots, \right)
$$

So You Need

Example. Radial Basis Function

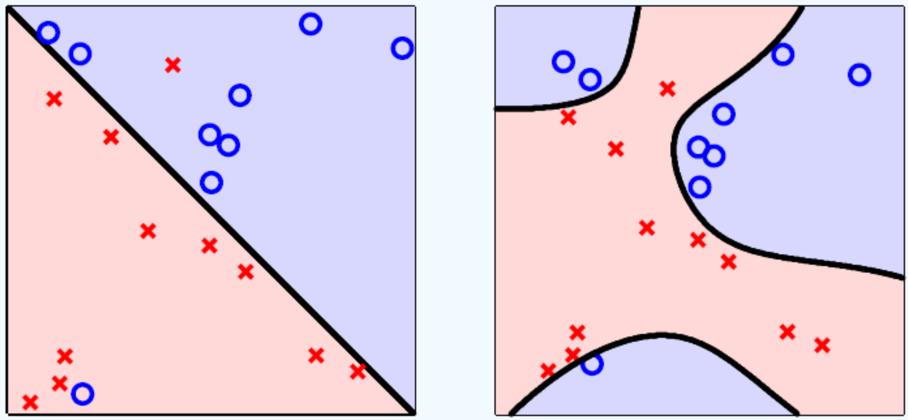
$$K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left\{-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|^2\right\}.$$

The non-linear transform is:

$$\Phi(x) = \exp\{-x^2\}\left(1, \sqrt{\frac{2^1}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \sqrt{\frac{2^3}{3!}}x^3, \ldots,\right)$$

- You need infinite dimensional non-linear transform!
- But to compute the kernel $K(\boldsymbol{u}, \boldsymbol{v})$ you do not need Φ.
- Another Good thing about Dual SVM: You can do infinite dimensional non-linear transform.
- Cost of computing $K(\boldsymbol{u}, \boldsymbol{v})$ is bottleneck by $\|\boldsymbol{u} - \boldsymbol{v}\|^2$.

Is RBF Always Better than Linear?



(a) linear classifier

(b) Gaussian-RBF kernel

- Noisy dataset: Linear works well.
- RBF: Over fit.

Testing with Kernels

- Recall:

$$\boldsymbol{w}^* = \sum_{n=1}^{N} \lambda_n^* y_n \boldsymbol{x}_n.$$

- The hypothesis function is

$$\begin{aligned} h(\boldsymbol{x}) &= \text{sign}\left(\boldsymbol{w}^{*T}\boldsymbol{x} + w_0^*\right) \\ &= \text{sign}\left(\left(\sum_{n=1}^{N} \lambda_n^* y_n \boldsymbol{x}_n\right)^T \boldsymbol{x} + w_0^*\right) \\ &= \text{sign}\left(\sum_{n=1}^{N} \lambda_n^* y_n \boldsymbol{x}_n^T \boldsymbol{x} + w_0^*\right). \end{aligned}$$

- Now you can replace $\boldsymbol{x}_n^T\boldsymbol{x}$ by $K(\boldsymbol{x}_n, \boldsymbol{x})$.

Reading List

Support Vector Machine

- Mustafa, *Learning from Data*, e-Chapter
- Duda-Hart-Stork, *Pattern Classification*, Chapter 5.5
- Chris Bishop, *Pattern Recognition*, Chapter 7.1
- UCSD Statistical Learning
  http://www.svcl.ucsd.edu/courses/ece271B-F09/