# CS540 Introduction to Artificial Intelligence

## Lecture 5

Young Wu

Based on lecture slides by Jerry Zhu, Yingyu Liang, and Charles Dyer

July 5, 2022

# Sharing Solutions

- For solution sharing, please make sure it is Piazza note, not a Piazza question.
- For actual questions, please use a different name, *e.g.*"$M2Q1$ Question" or "Question about $M2Q1$".
- Make sure you tag the post correctly: $m2$, $m3$, or $d1$ in order to get the points.
- Please sign up before making the post and please do not sign up for more than 4 questions per week.
- I will either "good note" the post or leave a comment: if I leave a comment, please update your answers, reply to my comment, and remember to make the reply "unresolved" so I can see.

# Margin and Support Vectors

## Motivation

- The perceptron algorithm finds any line $(w, b)$ that separates the two classes.

$$\hat{y}_i = \mathbb{1}_{\{w^T x_i + b \geqslant 0\}}$$

- The margin is the maximum width (thickness) of the line before hitting any data point.
- The instances that the thick line hits are called support vectors.
- The model that finds the line that separates the two classes with the widest margin is called support vector machine (SVM).

# Support Vector Machine

## Description

- The problem is equivalent to minimizing the squared norm of the weights $\|w\|^2 = w^T w$ subject to the constraint that every instance is classified correctly (with the margin).
- Use subgradient descent to find the weights and the bias.

# Finding the Margin

## Definition

- Define two planes: plus plane $w^T x + b = 1$ and minus plane $w^T x + b = -1$.
- The distance between the two planes is $\frac{2}{\sqrt{w^T w}}$.
- If all of the instances with $y_i = 1$ are above the plus plane and all of the instances with $y_i = 0$ are below the minus plane, then the margin is $\frac{2}{\sqrt{w^T w}}$.

# Constrained Optimization

## Definition

- The goal is to maximize the margin subject to the constraint that the plus plane and the minus plane separates the instances with $y_i = 0$ and $y_i = 1$.

$$\max_{w} \frac{2}{\sqrt{w^T w}} \text{ such that } \begin{cases} \left(w^T x_i + b\right) \leqslant -1 & \text{if } y_i = 0 \\ \left(w^T x_i + b\right) \geqslant 1 & \text{if } y_i = 1 \end{cases}, i = 1, 2, ..., n$$

- The two constraints can be combined.

$$\max_{w} \frac{2}{\sqrt{w^T w}} \text{ such that } \left(2 y_i - 1\right) \left(w^T x_i + b\right) \geqslant 1, i = 1, 2, ..., n$$

# Hard Margin SVM

## Definition

$$\max_{w} \frac{2}{\sqrt{w^T w}} \text{ such that } (2y_i - 1)\left(w^T x_i + b\right) \geqslant 1, i = 1, 2, ..., n$$

- This is equivalent to the following minimization problem, called hard margin SVM.

$$\min_{w} \frac{1}{2} w^T w \text{ such that } (2y_i - 1)\left(w^T x_i + b\right) \geqslant 1, i = 1, 2, ..., n$$

# Soft Margin

## Definition

- To allow for mistakes classifying a few instances, slack variables are introduced.
- The cost of violating the margin is given by some constant $\frac{1}{\lambda}$.
- Using slack variables $\xi_i$, the problem can be written as the following.

$$\min_{w} \frac{1}{2} w^T w + \frac{1}{\lambda} \frac{1}{n} \sum_{i=1}^{n} \xi_i$$

$$\text{such that } (2y_i - 1)\left(w^T x_i + b\right) \geqslant 1 - \xi_i, \xi_i \geqslant 0, i = 1, 2, ..., n$$

# Soft Margin SVM

## Definition

$$\min_{w} \frac{1}{2} w^T w + \frac{1}{\lambda} \frac{1}{n} \sum_{i=1}^{n} \xi_i$$

$$\text{such that } (2y_i - 1)\left(w^T x_i + b\right) \geqslant 1 - \xi_i, \xi_i \geqslant 0, i = 1, 2, ..., n$$

- This is equivalent to the following minimization problem, called soft margin SVM.

$$\min_{w} \frac{\lambda}{2} w^T w + \frac{1}{n} \sum_{i=1}^{n} \max\left\{0, 1 - (2y_i - 1)\left(w^T x_i + b\right)\right\}$$

# Subgradient Descent

## Definition

$$\min_{w} \frac{\lambda}{2} w^T w + \frac{1}{n} \sum_{i=1}^{n} \max \left\{ 0, 1 - (2y_i - 1) \left( w^T x_i + b \right) \right\}$$

- The gradient for the above expression is not defined at points with $1 - (2y_i - 1) \left( w^T x_i + b \right) = 0$.
- Subgradient can be used instead of a gradient.

## Subgradient

- The subderivative at a point of a convex function in one dimension is the set of slopes of the lines that are tangent to the function at that point.
- The subgradient is the version for higher dimensions.
- The subgradient $\partial f(x)$ is formally defined as the following set.

$$\partial f(x) = \left\{ v : f\left(x'\right) \geqslant f(x) + v^T \left(x' - x\right) \forall\, x' \right\}$$

# Subgradient Descent Step

## Definition

- One possible set of subgradients with respect to $w$ and $b$ are the following.

$$\partial_w C \ni \lambda w - \sum_{i=1}^{n} (2y_i - 1)\, x_i \mathbb{1}_{\{(2y_i - 1)(w^T x_i + b) \geqslant 1\}}$$

$$\partial_b C \ni - \sum_{i=1}^{n} (2y_i - 1)) \mathbb{1}_{\{(2y_i - 1)(w^T x_i + b) \geqslant 1\}}$$

- The gradient descent step is the same as usual, using one of the subgradients in place of the gradient.

# Class Notation and Bias Term

## Definition

- Usually, for SVM, the bias term is not included and updated. Also, the classes are -1 and +1 instead of 0 and 1. Let the labels be $z_i \in \{-1, +1\}$ instead of $y_i \in \{0, 1\}$. The gradient steps are usually written the following way.

$$w = (1 - \lambda)\, w - \alpha \sum_{i=1}^{n} z_i \mathbb{1}_{\{z_i w^T x_i \geqslant 1\}} x_i$$

$$z_i = 2y_i - 1, i = 1, 2, ..., n$$

# Regularization Parameter

## Definition

$$w = w - \alpha \sum_{i=1}^{n} z_i \mathbb{1}_{\{z_i w^T x_i \geqslant 1\}} x_i - \lambda w$$

$$z_i = 2y_i - 1, i = 1, 2, ..., n$$

- $\lambda$ is usually called the regularization parameter because it reduces the magnitude of $w$ the same way as the parameter $\lambda$ in $L2$ regularization.
- The stochastic subgradient descent algorithm for SVM is called PEGASOS: Primal Estimated sub-GrAdient SOlver for Svm.

# PEGASOS Algorithm

## Algorithm

- Inputs: instances: $\{x_i\}_{i=1}^{n}$ and $\{z_i = 2y_i - 1\}_{i=1}^{n}$
- Outputs: weights: $\{w_j\}_{j=1}^{m}$
- Initialize the weights.

$$w_j \sim \text{ Unif } [0, 1]$$

- Randomly permute (shuffle) the training set and performance subgradient descent for each instance $i$.

$$w = (1 - \lambda)\, w - \alpha z_i \mathbb{1}_{\{z_i w^T x_i \geqslant 1\}} x_i$$

- Repeat for a fixed number of iterations.

# Kernel Trick

## Motivation

- If the classes are not linearly separable, more features can be created.
- For example, a 1 dimensional $x$ can be mapped to $\varphi(x) = (x, x^2)$.
- Another example is to map a 2 dimensional $(x_1, x_2)$ to $\varphi(x = (x_1, x_2)) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$.

# Kernelized SVM

## Definition

- With a feature map $\varphi$, the SVM can be trained on new data points $\{(\varphi(x_1), y_1), (\varphi(x_2), y_2), \ldots, (\varphi(x_n), y_n)\}$.
- The weights $w$ correspond to the new features $\varphi(x_i)$.
- Therefore, test instances are transformed to have the same new features.

$$\hat{y}_i = \mathbb{1}_{\{w^T\varphi(x_i) \geqslant 0\}}$$

# Kernel Matrix

## Definition

- The feature map is usually represented by a $n \times n$ matrix $K$ called the Gram matrix (or kernel matrix).

$$K_{ii'} = \varphi\left(x_i\right)^T \varphi\left(x_{i'}\right)$$

# Examples of Kernel Matrix

## Definition

- For example, if $\varphi(x) = \left(x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)$, then the kernel matrix can be simplified.

$$K_{ii'} = \left(x_i^T x_{i'}\right)^2$$

- Another example is the quadratic kernel $K_{ii'} = \left(x_i^T x_{i'} + 1\right)^2$. It can be factored to have the following feature representations.

$$\varphi(x) = \left(x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right)$$

# Kernel Matrix Characterization

## Discussion

- A matrix $K$ is kernel (Gram) matrix if and only if it is symmetric positive semidefinite.
- Positive semidefiniteness is equivalent to having non-negative eigenvalues.

# Popular Kernels

## Discussion

- Other popular kernels include the following.

1. Linear kernel: $K_{ii'} = x_i^T x_{i'}$
2. Polynomial kernel: $K_{ii'} = \left(x_i^T x_{i'} + 1\right)^d$
3. Radial Basis Function (Gaussian) kernel:
$$K_{ii'} = \exp\left(-\frac{1}{\sigma^2}\left(x_i - x_{i'}\right)^T \left(x_i - x_{i'}\right)\right)$$

- Gaussian kernel has infinite-dimensional feature representations. There are dual optimization techniques to find $w$ and $b$ for these kernels.