

# Introduction to Machine Learning

## Kernel Support Vector Machines

Varun Chandola

Computer Science & Engineering  
State University of New York at Buffalo  
Buffalo, NY, USA  
chandola@buffalo.edu



University at Buffalo  
Department of Computer Science  
and Engineering  
School of Engineering and Applied Sciences



## Support Vector Machines

SVM Learning

Kernel SVM

# Support Vector Machines

- ▶ A hyperplane based classifier defined by  $\mathbf{w}$  and  $b$
- ▶ Like perceptron
- ▶ Find hyperplane with *maximum separation margin* on the training data
- ▶ Assume that data is linearly separable (will relax this later)
  - ▶ Zero training error (loss)

## SVM Prediction Rule

$$y = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$$

## SVM Learning

- ▶ **Input:** Training data  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
- ▶ **Objective:** Learn  $\mathbf{w}$  and  $b$  that maximizes the margin

- ▶ SVM learning task as an optimization problem
- ▶ Find  $\mathbf{w}$  and  $b$  that gives zero training error
- ▶ Maximizes the margin ( $= \frac{2}{\|\mathbf{w}\|}$ )
- ▶ Same as minimizing  $\|\mathbf{w}\|$

## Optimization Formulation

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{\|\mathbf{w}\|^2}{2} \\ & \text{subject to} && y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N. \end{aligned}$$

- ▶ **Optimization** with  $N$  linear inequality constraint

## Optimization Formulation

$$\begin{aligned} & \underset{\mathbf{w}, b}{\text{minimize}} && \frac{\|\mathbf{w}\|^2}{2} \\ & \text{subject to} && y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N. \end{aligned}$$

- ▶ Introducing [Lagrange Multipliers](#),  $\alpha_n$ ,  $n = 1, \dots, N$

## Rewriting as a (primal) Lagrangian

$$\begin{aligned} & \underset{\mathbf{w}, b, \alpha}{\text{minimize}} && L_P(\mathbf{w}, b, \alpha) = \frac{\|\mathbf{w}\|^2}{2} + \sum_{n=1}^N \alpha_n \{1 - y_n(\mathbf{w}^\top \mathbf{x}_n + b)\} \\ & \text{subject to} && \alpha_n \geq 0 \quad n = 1, \dots, N. \end{aligned}$$

# Solving the Lagrangian

- ▶ Set gradient of  $L_P$  to 0

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \alpha_n y_n = 0$$

- ▶ Substituting in  $L_P$  to get the dual  $L_D$

# Solving the Lagrangian

- ▶ Set gradient of  $L_P$  to 0

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \alpha_n y_n = 0$$

- ▶ Substituting in  $L_P$  to get the dual  $L_D$

## Dual Lagrangian Formulation

$$\underset{\mathbf{w}, b, \alpha}{\text{maximize}} \quad L_D(\mathbf{w}, b, \alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{m,n=1}^N \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^\top \mathbf{x}_n)$$

$$\text{subject to} \quad \sum_{n=1}^N \alpha_n y_n = 0, \alpha_n \geq 0 \quad n = 1, \dots, N.$$

# A Key Observation from Dual Formulation

## Dot Product Formulation

- ▶ All training examples ( $\mathbf{x}_n$ 's) occur in *dot/inner products*
- ▶ Also recall the prediction using SVMs

$$\begin{aligned}y^* &= \text{sign}(\mathbf{w}^\top \mathbf{x}^* + b) \\&= \text{sign}\left(\left(\sum_{n=1}^N \alpha_n y_n \mathbf{x}_n\right)^\top \mathbf{x}^* + b\right) \\&= \text{sign}\left(\sum_{n=1}^N \alpha_n y_n \boxed{\mathbf{x}_n^\top \mathbf{x}^*} + b\right)\end{aligned}$$

- ▶ Replace the dot products with kernel functions
  - ▶ Kernel or non-linear SVM



# Widely used variant of SVM

- ▶ Kernel SVM with radial basis function kernel (RBF)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\gamma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

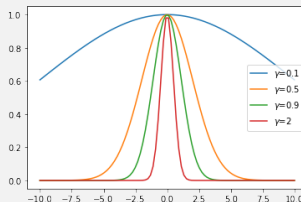
## Setting $\gamma$ and $C$

- ▶  $C$  is the regularization parameter

$$L(\mathbf{w}, b) = \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad \text{subject to constraints}$$

- ▶ For the role of  $\gamma$ , consider the following two aspects:

$$y^* = \text{sign}\left(\sum_{n=1}^N \alpha_n y_n (\mathbf{x}_n^\top \mathbf{x}^*) + b\right)$$



# What should $\gamma$ and $C$ be?

- ▶  $\gamma$  determines the influence of a training example - inverse of the radius of influence of support vectors
- ▶  $C$  determines the trade-off between the total slack (errors on training data) and the size of the margin (regularization)
- ▶ Setting  $\gamma$  too large makes the decision boundary too complex,  $C$  will not prevent overfitting
- ▶ Setting  $\gamma$  very small makes the decision boundary simple (linear)
- ▶ Usually a grid search is performed to identify optimal  $C$  and  $\gamma$

# Concluding Remarks on SVM

- ▶ Training time for SVM training is  $O(N^3)$
- ▶ Many *faster* but approximate approaches exist
  - ▶ Approximate QP solvers
  - ▶ Online training
- ▶ SVMs can be extended in different ways
  1. Non-linear boundaries (**kernel trick**)
  2. Multi-class classification
  3. Probabilistic output
  4. Regression (Support Vector Regression)

# References