Support Vector Machines (II)

CMSC 422
MARINE CARPUAT
marine@cs.umd.edu

Slides credit: Piyush Rai
What we know about SVM so far

REVIEW
The Maximum Margin Principle

• Find the hyperplane with *maximum separation margin* on the training data
Support Vector Machine (SVM)

A hyperplane based linear classifier defined by \( \mathbf{w} \) and \( b \)

Prediction rule: \( y = \text{sign}(\mathbf{w}^T \mathbf{x} + b) \)

**Given:** Training data \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \)

**Goal:** Learn \( \mathbf{w} \) and \( b \) that achieve the maximum margin
Characterizing the margin

Let’s assume the entire training data is correctly classified by \((\mathbf{w},b)\) that achieve the maximum margin.

- Assume the hyperplane is such that
  - \(\mathbf{w}^T \mathbf{x}_n + b \geq 1\) for \(y_n = +1\)
  - \(\mathbf{w}^T \mathbf{x}_n + b \leq -1\) for \(y_n = -1\)
  - Equivalently, \(y_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1\)
    \[\Rightarrow \min_{1 \leq n \leq N} |\mathbf{w}^T \mathbf{x}_n + b| = 1\]

- The hyperplane’s margin:
  \[\gamma = \min_{1 \leq n \leq N} \frac{|\mathbf{w}^T \mathbf{x}_n + b|}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||}\]
Solving the SVM Optimization Problem  
(assuming linearly separable data)

Our optimization problem is:

Minimize  \( f(w, b) = \frac{||w||^2}{2} \) 
subject to  \( 1 \leq y_n(w^T x_n + b), \quad n = 1, \ldots, N \)

Introducing Lagrange Multipliers \( \alpha_n \ (n = \{1, \ldots, N\}) \), one for each constraint, leads to the Lagrangian:

Minimize  \( L(w, b, \alpha) = \frac{||w||^2}{2} + \sum_{n=1}^{N} \alpha_n \{1 - y_n(w^T x_n + b)\} \) 
subject to  \( \alpha_n \geq 0; \quad n = 1, \ldots, N \)
Solving the SVM Optimization Problem
(assuming linearly separable data)

Take (partial) derivatives of $L_P$ w.r.t. $w$, $b$ and set them to zero

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

Substituting these into the Primal Lagrangian $L_P$ gives the Dual Lagrangian

Maximize

$$
L_D(w, b, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (x_m^T x_n)
$$

subject to

$$
\sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \geq 0; \quad n = 1, \ldots, N
$$
SVM: the solution!
(assuming linearly separable data)

Once we have the $\alpha_n$'s, $\mathbf{w}$ and $b$ can be computed as:

$$
\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n
$$

$$
b = -\frac{1}{2} \left( \min_{n: y_n = +1} \mathbf{w}^T \mathbf{x}_n + \max_{n: y_n = -1} \mathbf{w}^T \mathbf{x}_n \right)
$$

**Note:** Most $\alpha_n$'s in the solution are zero (sparse solution)

- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal $\alpha_n$'s

  $$
  \alpha_n \{1 - y_n (\mathbf{w}^T \mathbf{x}_n + b)\} = 0
  $$

  $\alpha_n$ is non-zero only if $\mathbf{x}_n$ lies on one of the two margin boundaries, i.e., for which $y_n (\mathbf{w}^T \mathbf{x}_n + b) = 1$

- These examples are called **support vectors**
- Support vectors “support” the margin boundaries
What if the data is not separable?

GENERAL CASE SVM SOLUTION
SVM in the non-separable case

• no hyperplane can separate the classes perfectly

• We still want to find the max margin hyperplane, but
  – We will allow some training examples to be misclassified
  – We will allow some training examples to fall within the margin region
Recall: For the separable case (training loss = 0), the constraints were:

$$y_n(w^T x_n + b) \geq 1 \quad \forall n$$

For the non-separable case, we relax the above constraints as:

$$y_n(w^T x_n + b) \geq 1 - \xi_n \quad \forall n$$

$\xi_n$ is called slack variable (distance $x_n$ goes past the margin boundary)

$\xi_n \geq 0, \forall n$, misclassification when $\xi_n > 1$
SVM Optimization Problem

Non-separable case: We will allow misclassified training examples
• but we want their number to be minimized
  ⇒ by minimizing the sum of slack variables \((\sum_{n=1}^{N} \xi_n)\)

The optimization problem for the non-separable case

Minimize \( f(w, b) = \frac{||w||^2}{2} + C \sum_{n=1}^{N} \xi_n \)

subject to \( y_n(w^T x_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad n = 1, \ldots, N \)

C hyperparameter dictates which term dominates the minimization
• Small C => prefer large margins and allows more misclassified examples
• Large C => prefer small number of misclassified examples, but at the expense of a small margin
Introducing Lagrange Multipliers...

Our optimization problem is:

Minimize \( f(w, b, \xi) = \frac{||w||^2}{2} + C \sum_{n=1}^{N} \xi_n \)

subject to \( 1 \leq y_n(w^T x_n + b) + \xi_n, \quad 0 \leq \xi_n \quad n = 1, \ldots, N \)

Introducing Lagrange Multipliers \( \alpha_n, \beta_n \ (n = \{1, \ldots, N\}) \), for the constraints, leads to the Primal Lagrangian:

Minimize \( L_P(w, b, \xi, \alpha, \beta) = \frac{||w||^2}{2} + C \sum_{n=1}^{N} \xi_n + \sum_{n=1}^{N} \alpha_n \{1 - y_n(w^T x_n + b) - \xi_n\} - \sum_{n=1}^{N} \beta_n \xi_n \)

subject to \( \alpha_n, \beta_n \geq 0; \quad n = 1, \ldots, N \)

Terms in red are those that were not there in the separable case!
Formulating the dual objective

Take (partial) derivatives of $L_P$ w.r.t. $\mathbf{w}$, $b$, $\xi_n$ and set them to zero

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n,$$
$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0,$$
$$\frac{\partial L_P}{\partial \xi_n} = 0 \Rightarrow C - \alpha_n - \beta_n = 0$$

Using $C - \alpha_n - \beta_n = 0$ and $\beta_n \geq 0 \Rightarrow \alpha_n \leq C$

Substituting these in the Primal Lagrangian $L_P$ gives the Dual Lagrangian

Maximize $L_D(\mathbf{w}, b, \xi, \alpha, \beta) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^T \mathbf{x}_n)$

subject to $\sum_{n=1}^{N} \alpha_n y_n = 0$, $0 \leq \alpha_n \leq C$; $n = 1, \ldots, N$

Note
- Given $\alpha$ the solution for $\mathbf{w}$, $b$ has the same form as in the separable case
- $\alpha$ is again sparse, nonzero $\alpha_n$’s correspond to support vectors
Support Vectors in the Non-Separable Case

We now have 3 types of support vectors!

1. Lying on the margin boundaries $\mathbf{w}^T \mathbf{x} + b = -1$ and $\mathbf{w}^T \mathbf{x} + b = +1$ ($\xi_n = 0$)
2. Lying within the margin region ($0 < \xi_n < 1$) but still on the correct side
3. Lying on the wrong side of the hyperplane ($\xi_n \geq 1$)
Notes on training

• Solving the quadratic problem is $O(N^3)$
  – Can be prohibitive for large datasets

• But many options to speed up training
  – Approximate solvers
  – Learn from what we know about training linear models
Recall: Learning a Linear Classifier as an Optimization Problem

Objective function

\[ \min_{w, b} L(w, b) = \min_{w, b} \sum_{n=1}^{N} \mathbb{I}(y_n(w^T x_n + b) < 0) + \lambda R(w, b) \]

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well
Recall: Learning a Linear Classifier as an Optimization Problem

\[
\min_{\mathbf{w}, b} L(\mathbf{w}, b) = \min_{\mathbf{w}, b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)
\]

- **Problem**: The 0-1 loss above is NP-hard to optimize exactly/approximately in general

- **Solution**: Different loss function approximations and regularizers lead to specific algorithms (e.g., perceptron, support vector machines, etc.)
Recall: Approximating the 0-1 loss with surrogate loss functions

• Examples (with $b = 0$)
  – Hinge loss: $[1 - y_nw^T x_n]_+ = \max\{0, 1 - y_nw^T x_n\}$
  – Log loss: $\log[1 + \exp(-y_nw^T x_n)]$
  – Exponential loss: $\exp(-y_nw^T x_n)$

• All are convex upper-bounds on the 0-1 loss
What is the SVM loss function?

No penalty ($\xi_n = 0$) if $y_n(w^T x_n + b) \geq 1$

Linear penalty ($\xi_n = 1 - y_n(w^T x_n + b)$) if $y_n(w^T x_n + b) < 1$

It’s precisely the hinge loss $\max\{0, 1 - y_n(w^T x_n + b)\}$
Recall: What is the perceptron optimizing?

**Algorithm 5 PerceptronTrain(D, MaxIter)**

1. $w_d \leftarrow 0$, for all $d = 1 \ldots D$  // initialize weights
2. $b \leftarrow 0$  // initialize bias
3. **for iter = 1 \ldots MaxIter do**
4.  **for all** $(x,y) \in D$ **do**
5.    $a \leftarrow \sum_{d=1}^{D} w_d \cdot x_d + b$  // compute activation for this example
6.    **if** $ya \leq 0$ **then**
7.      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$  // update weights
8.      $b \leftarrow b + y$  // update bias
9.    **end if**
10.  **end for**
11. **end for**
12. **return** $w_0, w_1, \ldots, w_D, b$

- Loss function is a variant of the hinge loss
  \[
  \max\{0, -y_n(w^T x_n + b)\}
  \]
SVM + KERNELS
Kernelized SVM training

Recall the SVM dual Lagrangian:

Maximize \( L_D(w, b, \xi, \alpha, \beta) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (x_m^T x_n) \)

subject to \( \sum_{n=1}^{N} \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C; \quad n = 1, \ldots, N \)

Replacing \( x_m^T x_n \) by \( \phi(x_m)^T \phi(x_n) = k(x_m, x_n) = K_{mn} \), where \( k(., .) \) is some suitable kernel function

Maximize \( L_D(w, b, \xi, \alpha, \beta) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n K_{mn} \)

subject to \( \sum_{n=1}^{N} \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C; \quad n = 1, \ldots, N \)

SVM now learns a linear separator in the kernel defined feature space \( \mathcal{F} \)
Kernelized SVM prediction

Prediction for a test example \( \mathbf{x} \) (assume \( b = 0 \))

\[
y = \text{sign}(\mathbf{w}^\top \mathbf{x}) = \text{sign}\left( \sum_{n \in SV} \alpha_n y_n \mathbf{x}_n^\top \mathbf{x} \right)
\]

\( SV \) is the set of support vectors (i.e., examples for which \( \alpha_n > 0 \))

Replacing each example with its feature mapped representation (\( \mathbf{x} \to \phi(\mathbf{x}) \))

\[
y = \text{sign}\left( \sum_{n \in SV} \alpha_n y_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}) \right) = \text{sign}\left( \sum_{n \in SV} \alpha_n y_n k(\mathbf{x}_n, \mathbf{x}) \right)
\]

The weight vector for the kernelized case can be expressed as:

\[
\mathbf{w} = \sum_{n \in SV} \alpha_n y_n \phi(\mathbf{x}_n) = \sum_{n \in SV} \alpha_n y_n k(\mathbf{x}_n, \cdot)
\]

Note
- Kernelized SVM needs the support vectors at test time!
- While unkernelized SVM can just store \( \mathbf{w} \)
Example: decision boundary of an SVM with an RBF Kernel
What you should know

• What are Support Vector Machines
• How to train SVMs
  – Which optimization problem we need to solve
• Geometric interpretation
  - What are support vectors and what is their relationship with parameters $w, b$?
• How do SVM relate to the general formulation of linear classifiers
• Why/how can SVMs be kernelized