CS5489
Lecture 4.2: Support Vector Machines: Part II

Kede Ma

City University of Hong Kong (Dongguan)

Slide template by courtesy of Benjamin M. Marlin

Outline

Linear Classifiers

- So far, we have mainly looked at linear classifiers
  - Separate classes using a hyperplane (line, plane)
  - E.g., linear discriminant analysis, logistic regression, support vector machine

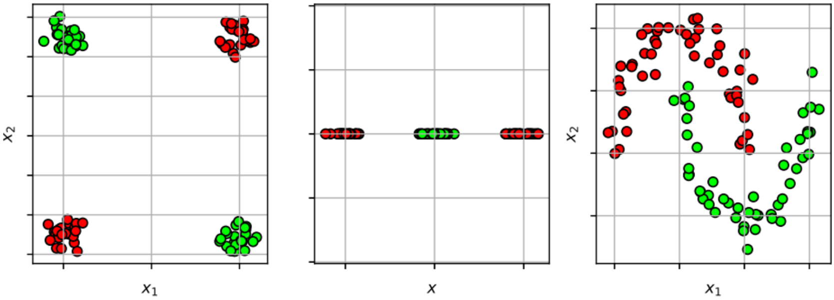
- SVM (primal form, soft-margin):

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{M}\xi_i \\ \text{subject to} \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i,\ i = 1, \ldots, M \\ & \xi_i \geq 0,\ i = 1, \ldots, M \end{aligned}$$

Outline

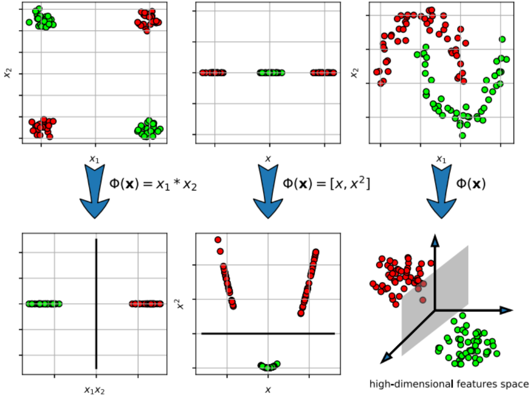
Non-Linear Decision Boundary

- What if the data is separable, but not linearly separable?



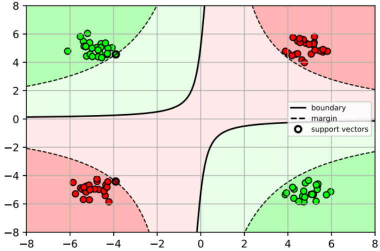
Idea - Transform the Input Space

- Map $\mathbf{x} \in \mathbb{R}^N$ to a new high-dimensional space $\mathbf{z} \in \mathbb{R}^L$
  - $\mathbf{z} = \phi(\mathbf{x})$, where $\phi(\mathbf{x})$ is the transform function
- Learn the linear classifier in the new space



Example

- Let's try it...
  - 2-dimension vector inputs
    - $\mathbf{x} = [x_1, x_2]^T$
  - Transformation consists of quadratic terms
    - $\mathbf{z} = [x_1^2, x_1 x_2, x_2^2]^T$
  - SVM:



SVM with Transformed Input

- Given a training set $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{M}$, the original SVM:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad \text{s.t.} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad 1 \leq i \leq M$$

- Apply high-dimensional transform to input $\mathbf{x} \rightarrow \phi(\mathbf{x})$

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad \text{s.t.} \quad y^{(i)}(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + b) \geq 1, \quad 1 \leq i \leq M$$

- The hyperplane $\mathbf{w} \in \mathbb{R}^L$ is now in the high-dimensional space!
  - If L is very large
    - Calculating feature vector $\phi(\mathbf{x})$ could be time consuming
    - Optimization could be very inefficient in high-dimensional space

SVM: From Primal to Dual

- Primal form:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{M}\xi_i \\ \text{subject to} \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i,\ i = 1, \ldots, M \\ & \xi_i \geq 0,\ i = 1, \ldots, M \end{aligned}$$

- We form the Lagrangian:

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) =& \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\alpha_i[y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \\ & - 1 + \xi_i] - \sum_{i=1}^{M}\beta_i\xi_i \end{aligned}$$

SVM: From Primal to Dual

- Take the derivatives of L w.r.t. $\mathbf{w}, b, \boldsymbol{\xi}$ and set them to zero:

$$\nabla_{\mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{M} \alpha_i y^{(i)} \mathbf{x}^{(i)} = 0$$

$$\nabla_b L = \sum_{i=1}^{M} \alpha_i y^{(i)} = 0$$

$$\nabla_{\xi_i} L = C - \alpha_i - \beta_i = 0, \quad i = 1, \ldots, M$$

- Plug them back into the Lagrangian

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y^{(i)} y^{(j)} \alpha_i \alpha_j (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

SVM: The Dual Problem

- Put this together with the constraints and eliminate β_i:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y^{(i)} y^{(j)} \alpha_i \alpha_j (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

$$\text{subject to} \quad \sum_{i=1}^{M} \alpha_i y^{(i)} = 0,$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, M$$

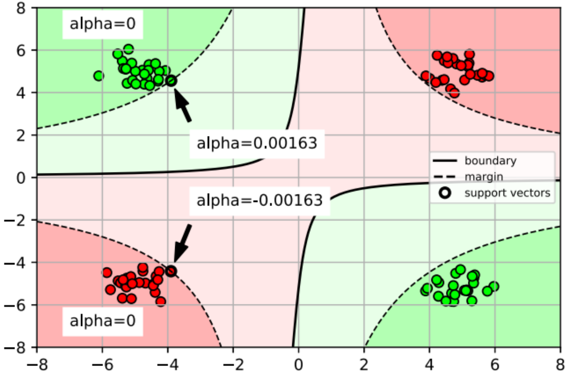
- α_i corresponds to the ith training sample $(\mathbf{x}^{(i)}, y^{(i)})$
- Recover $\mathbf{w}$ as a sum of the training points: $\mathbf{w} = \sum_{i=1}^{M} \alpha_i y^{(i)} \mathbf{x}^{(i)}$
- Classify a new point $\mathbf{x}^*$

$$y^* = \text{sign}(\mathbf{w}^T \mathbf{x}^* + b) = \text{sign}(\sum_{i=1}^{M} \alpha_i y^{(i)} (\mathbf{x}^{(i)})^T \mathbf{x}^* + b)$$

Interpretation of α_i

- Combining complementary slackness with primal feasibility and dual feasibility, we can show that
  - $\alpha_i^\star = 0 \Rightarrow y^{(i)}((\mathbf{x}^{(i)})^T\mathbf{w}^\star + b^\star) \geq 1$
    - I.e., the sample $\mathbf{x}^{(i)}$ is not on the margin
  - $0 < \alpha_i^\star < C \Rightarrow y^{(i)}((\mathbf{x}^{(i)})^T\mathbf{w}^\star + b^\star) = 1$
    - I.e., the sample $\mathbf{x}^{(i)}$ is on the margin
  - $\alpha_i^\star = C \Rightarrow y^{(i)}((\mathbf{x}^{(i)})^T\mathbf{w}^\star + b^\star) \leq 1$
    - I.e., the sample $\mathbf{x}^{(i)}$ violates the margin

Example



alpha=0
alpha=0.00163
alpha=-0.00163
alpha=0
boundary
margin
support vectors
8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8

Kernel Function

- Dual SVM with transformed input:

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{M}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{M} y^{(i)}y^{(j)}\alpha_i\alpha_j\boldsymbol{\phi}(\mathbf{x}^{(i)})^T\boldsymbol{\phi}(\mathbf{x}^{(j)}) \\ \text{subject to} \quad & \sum_{i=1}^{M}\alpha_i y^{(i)} = 0, \\ & 0 \le \alpha_i \le C, \quad i = 1, \ldots, M\end{aligned}$$

- Completely written in terms of inner product $\boldsymbol{\phi}(\mathbf{x}^{(i)})^T\boldsymbol{\phi}(\mathbf{x}^{(j)})$
- Rather than explicitly calculate the high-dimensional $\boldsymbol{\phi}(\mathbf{x}^{(i)})$
  - Only need to calculate the inner product between two vectors
- **Kernel function**: $\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \boldsymbol{\phi}(\mathbf{x}^{(i)})^T\boldsymbol{\phi}(\mathbf{x}^{(j)})$
  - Much less expensive to compute than explicitly calculating the high dimensional feature vector and the inner product

Example: Polynomial Kernel

- Input vector $\mathbf{x} = [x_1, \ldots, x_N]^T \in \mathbb{R}^N$
- Kernel between two vectors is a p-th order polynomial:
  - $\mathcal{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}')^p = (\sum_{i=1}^N x_i x_i')^p$
- For example, $p = 2$,

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}')^2 = \left(\sum_{i=1}^{N} x_i x_i'\right)^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}(x_i x_i' x_j x_j') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$$

  - Transformed space is the quadratic terms of input $\mathbf{x}$

$$\phi(\mathbf{x}) = [x_1x_1, x_1x_2, \ldots, x_2x_1, x_2x_2, \ldots, x_Nx_1, \ldots, x_Nx_N]$$

- Comparison of number of multiplications
  - For kernel: $\mathcal{O}(N)$
  - Explicit transformation ϕ: $\mathcal{O}(N^2)$

Kernel Trick

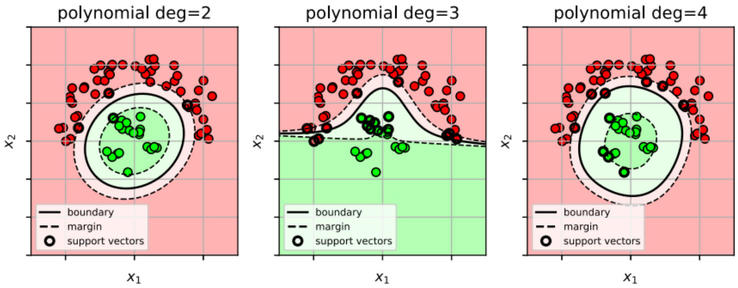
- Replacing the inner product with a kernel function in optimization is called the **kernel trick**
  - Turns a linear classifier into a non-linear one
  - The shape of the decision boundary is determined by the kernel
- **Kernel SVM**:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y^{(i)} y^{(j)} \alpha_i \alpha_j \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{subject to} \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, M \end{aligned}$$

  - Prediction: $y^* = \text{sign}\left(\sum_{i=1}^{M} \alpha_i y^{(i)} \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^*) + b\right)$

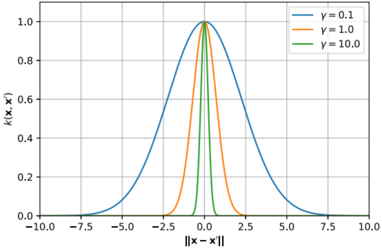
Example: Kernel SVM with Polynomial Kernel

- Decision surface is a "cut" of a polynomial surface
- Higher polynomial-order yields more complex decision boundaries



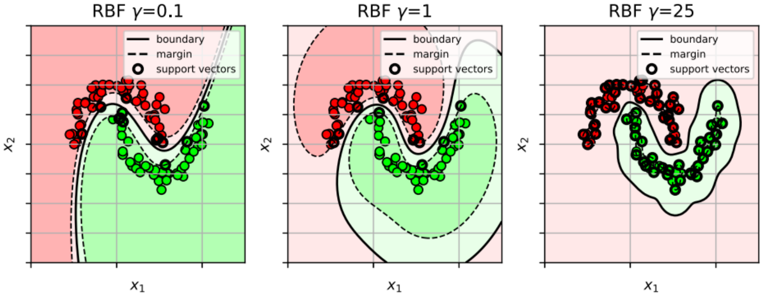
RBF Kernel

- RBF kernel (radial basis function)
  - $\mathcal{K}(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x}-\mathbf{x}'\|^2}$
  - Similar to a Gaussian
- Gamma $\gamma > 0$ is the inverse bandwidth parameter of the kernel
  - Controls the smoothness of the function
  - Small $\gamma \rightarrow$ wide Gaussian $\rightarrow$ smooth functions
  - Large $\gamma \rightarrow$ thin Gaussian $\rightarrow$ wiggly functions
- Corresponds to feature transform function of infinite dimension



Kernel SVM with RBF Kernel

- Try different γ
  - Each γ yields different levels of smoothness of the decision boundary



Kernel SVM Summary

- **Kernel classifier**:
  - Kernel function defines the shape of the non-linear decision boundary
    - Implicitly transforms input feature into high-dimensional space
    - Uses linear classifier in high-dim space
    - The decision boundary is non-linear in the original input space
- **Training**:
  - Maximize the margin of the training data
    - I.e., maximize the separation between the points and the decision boundary
  - Use cross-validation to pick the hyperparameter and the kernel hyperparameters

Kernel SVM Summary

- **Advantages**:
  - Non-linear decision boundary for more complex classification problems
  - Some intuition from the type of kernel function used
- **Disadvantages**:
  - Sensitive to the kernel function used
  - Sensitive to the C and kernel hyperparameters
  - Computationally expensive to do cross-validation
  - Need to calculate the kernel matrix
    - M^2 terms where M is the size of the training set
    - For large M, uses a large amount of computation and memory

Outline

Solving SVM: Coordinate Descent

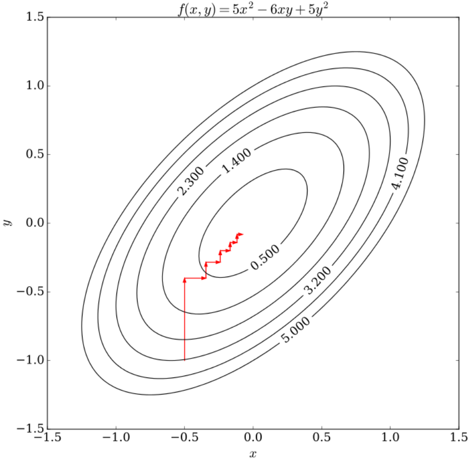
- Consider the unconstrained optimization problem

$$\min_{\boldsymbol{\theta}} \ell(\theta_1, \theta_2, \ldots, \theta_N)$$

- The coordinate descent method first selects an initial point $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^N$ and repeats:
  - Choose an index i from 1 to N
  - $\theta_i^{(t+1)} = \arg\min_{\hat{\theta}_i} \ell(\theta_1^{(t)}, \ldots, \theta_{i-1}^{(t)}, \hat{\theta}_i, \theta_{i+1}^{(t)}, \ldots, \theta_N^{(t)})$
- At each iteration, the algorithm determines a coordinate, and minimizes over the corresponding coordinate direction while fixing all other coordinates
- Coordinate descent is applicable in both differentiable and non-differentiable contexts

Example

- Consider minimizing $f(x, y) = 5x^2 - 6xy + 5y^2$



Solving SVM: Sequential Minimal Optimization (SMO)

- SMO is an algorithm for solving the quadratic programming problem that arises during the training of SVMs, invented by John C. Platt in 1998
- Sequential
  - Not parallel
  - Optimize a set of two Lagrange multipliers
- Minimal
  - Optimize the smallest possible sub-problem at each step
- Optimization
  - Satisfy the constraints for the chosen pair of Lagrange multipliers

SMO

- Recall the dual form of kernelized SVM with soft margin

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y^{(i)} y^{(j)} \alpha_i \alpha_j \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\
\text{s.t.} \quad & \sum_{i=1}^{M} \alpha_i y^{(i)} = 0, \\
& 0 \le \alpha_i \le C, \quad i = 1, \ldots, M
\end{aligned}
$$

- SMO is derived by decomposing the problem to its extreme and optimizing a minimal subset of just two dual variables (i.e., two data points) at each iteration

SMO

- The sub-problem for two data points admits an analytical solution, eliminating the need to use an iterative quadratic programming optimizer as part of the algorithm
- The condition $\sum_{i=1}^{M} \alpha_i y^{(i)} = 0$ is enforced throughout the iterations, implying that the smallest number of multipliers that can be optimized at each step is two: whenever one multiplier is updated, at least one other multiplier needs to be adjusted in order to keep the condition true
- At each step SMO chooses two elements α_i and α_j to jointly optimize, finds the optimal values for those two parameters given that all the others are fixed

SMO

- The choice of the two points is determined by a heuristic, while the optimization of the two multipliers is performed analytically
- Despite needing more iterations to converge, each iteration uses so few operations that the algorithm exhibits an overall speed-up of some orders of magnitude

Outline

Classification Summary

- **Classification task**:
  - Observation $\mathbf{x}$: typically a real vector of feature values, $\mathbf{x} \in \mathbb{R}^N$
  - Class y: from a set of possible classes, e.g., $\mathcal{Y} = \{-1, +1\}$
  - **Goal**: Given an observation $\mathbf{x}$, predict its class y
- **K-Nearest Neighbors**
  - **Type**: discriminative
  - **Classes**: multi-class
  - **Decision function**: non-linear
  - **Training**: no training needed
- **Bayes' classifier**
  - **Type**: generative
  - **Classes**: multi-class
  - **Decision function**: (generally) non-linear
  - **Training**: estimate class-conditional densities $p(\mathbf{x}|y)$ by maximizing likelihood of data

Classification Summary

- **Logistic regression**
  - **Type**: discriminative
  - **Classes**: binary
  - **Decision function**: linear
  - **Training**: maximize likelihood of data in $p(y|\mathbf{x})$
- **Support vector machine**
  - **Type**: discriminative
  - **Classes**: binary
  - **Decision function**: linear
  - **Training**: maximize the margin (distance between the decision surface and the closest point)
- **Kernel SVM**
  - **Type**: discriminative
  - **Classes**: binary
  - **Decision function**: non-linear (kernel function)
  - **Training**: maximize the margin

Regularization and Overfitting

- Some models have terms to prevent overfitting the training data
  - This can improve generalization to new data
- There is a parameter to control the regularization effect
  - Select this parameter using cross-validation on the training set

Other Things

- Multiclass classification
  - Can use binary classifiers to do multi-class using 1-vs-rest formulation
- Feature normalization
  - Normalize each feature dimension so that some feature dimensions with larger ranges do not dominate the optimization process
- Data imbalance
  - If more data in one class, then apply weights to each class to balance objectives
- Class importance
  - Mistakes on some classes are more critical
  - Reweight class to focus classifier on correctly predicting one class at the expense of others

Other Things

- **Applications**:
  - Web document classification, spam classification
  - Face gender recognition, face detection, digit classification
- **Features**
  - Choice of features is important!
    - Using uninformative features may confuse the classifier
    - Use domain knowledge to pick the best features to extract from the data

Which Classifier Is Best?

"No Free Lunch" Theorem (Wolpert and Macready)

"If an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems

- In other words, there is **no best** classifier for all tasks. The best classifier depends on the particular problem