Non-Parametric Methods and Support Vector Machines

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning

Outline

1. **Non-Parametric Methods**
   - K-NN
   - Parzen Windows
   - Local Models
2. **Support Vector Machines**
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick

Outline

1. **Non-Parametric Methods**
   - K-NN
   - Parzen Windows
   - Local Models

2. **Support Vector Machines**
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick

Outline

1. **Non-Parametric Methods**
   - K-NN
   - Parzen Windows
   - Local Models
2. **Support Vector Machines**
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick

K-NN Methods I

- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

K-NN Methods I

- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

1. Choose the number K and a distance metric

K-NN Methods I

- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

1. Choose the number K and a distance metric
2. Find the K nearest neighbors of a given point $\boldsymbol{x}$

K-NN Methods I

- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

1. Choose the number K and a distance metric
2. Find the K nearest neighbors of a given point $\boldsymbol{x}$
3. Predict the label of $\boldsymbol{x}$ by the majority vote (in classification) or average (in regression) of NNs' labels

K-NN Methods I

- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

1. Choose the number K and a distance metric
2. Find the K nearest neighbors of a given point $\boldsymbol{x}$
3. Predict the label of $\boldsymbol{x}$ by the majority vote (in classification) or average (in regression) of NNs' labels

- Distance metric?

K-NN Methods I

- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

1. Choose the number K and a distance metric
2. Find the K nearest neighbors of a given point $\boldsymbol{x}$
3. Predict the label of $\boldsymbol{x}$ by the majority vote (in classification) or average (in regression) of NNs' labels

- Distance metric? E.g., Euclidean distance $d(\boldsymbol{x}^{(i)},\boldsymbol{x}) = \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|$
- Training algorithm?

K-NN Methods I

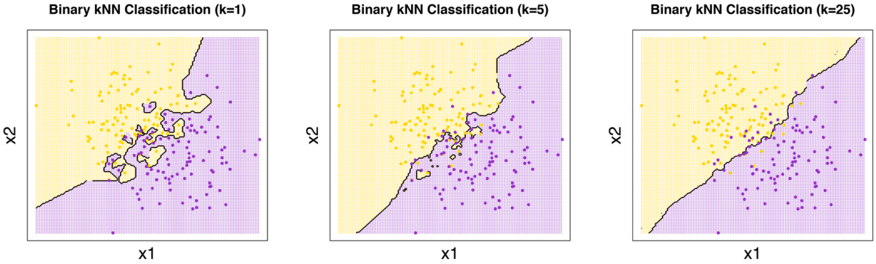
- The ***K-nearest neighbor*** (K-NN) methods are straightforward, but a fundamentally different way, to predict the label of a data point $\boldsymbol{x}$:

1. Choose the number K and a distance metric
2. Find the K nearest neighbors of a given point $\boldsymbol{x}$
3. Predict the label of $\boldsymbol{x}$ by the majority vote (in classification) or average (in regression) of NNs' labels

- Distance metric? E.g., Euclidean distance $d(\boldsymbol{x}^{(i)}, \boldsymbol{x}) = \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|$
- Training algorithm? Simply "remember" $\mathbb{X}$ in storage

K-NN Methods II

- Could be very complex
- K is a hyperparameter controlling the model complexity



Non-Parametric Methods

- K-NN method is a special case of ***non-parametric*** (or ***memory-based***) methods

Non-Parametric Methods

- K-NN method is a special case of ***non-parametric*** (or ***memory-based***) methods
  - Non-parametric in the sense that f are not described by only few parameters

Non-Parametric Methods

- K-NN method is a special case of ***non-parametric*** (or ***memory-based***) methods
  - Non-parametric in the sense that f are not described by only few parameters
    之前都用w來代表f 即為parametric
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process

Non-Parametric Methods

- K-NN method is a special case of ***non-parametric*** (or ***memory-based***) methods
  - Non-parametric in the sense that f are not described by only few parameters
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process
- K-NN is also a ***lazy*** method since the prediction function f is obtained only before the prediction

Non-Parametric Methods

- K-NN method is a special case of ***non-parametric*** (or ***memory-based***) methods
  - Non-parametric in the sense that f are not described by only few parameters
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process
- K-NN is also a ***lazy*** method since the prediction function f is obtained only before the prediction
  - Motivates the development of other ***local models***

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems
- Cons:
  - Storage demanding

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems
- Cons:
  - Storage demanding
  - Sensitive to outliers

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems
- Cons:
  - Storage demanding
  - Sensitive to outliers
  - Sensitive to irrelevant data features (vs. decision trees)

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems
- Cons:
  - Storage demanding
  - Sensitive to outliers
  - Sensitive to irrelevant data features (vs. decision trees)
  - Needs to deal with missing data (e.g., special distances)

Pros & Cons

- Pros:
  - Almost no assumption on f other than ***smoothness***
    - High capacity/complexity
    - High accuracy given a large training set
  - Supports online training (by simply memorizing)
  - Readily extensible to multi-class and regression problems
- Cons:
  - Storage demanding
  - Sensitive to outliers
  - Sensitive to irrelevant data features (vs. decision trees)
  - Needs to deal with missing data (e.g., special distances)
  - Computationally expensive: $O(ND)$ time for making each prediction
    - Can speed up with index and/or approximation

    每次預測都要看這個x和其他人的距離多少O(ND)

Outline

Parzen Windows and Kernels

- Binary KNN classifier:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i:\boldsymbol{x}^{(i)} \in \mathsf{KNN}(\boldsymbol{x})} y^{(i)}\right)$$

  - The "radius" of voter boundary depends on the input $\boldsymbol{x}$

Parzen Windows and Kernels

- Binary KNN classifier:

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{i:\boldsymbol{x}^{(i)} \in \mathsf{KNN}(\boldsymbol{x})} y^{(i)}\right)$$

  - The "radius" of voter boundary depends on the input $\boldsymbol{x}$
- We can instead use the ***Parzen window*** with a fixed radius:

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_i y^{(i)} 1(\boldsymbol{x}^{(i)}; \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\| \leq R)\right)$$

Parzen Windows and Kernels

- Binary KNN classifier:

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{i:\boldsymbol{x}^{(i)}\in \mathsf{KNN}(\boldsymbol{x})} y^{(i)}\right)$$

  - The "radius" of voter boundary depends on the input $\boldsymbol{x}$
- We can instead use the ***Parzen window*** with a fixed radius:

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_i y^{(i)} 1(\boldsymbol{x}^{(i)}; \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\| \leq R)\right)$$

- Parzen windows also replace the hard boundary with a soft one:

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x})\right)$$

  - $k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ is a ***radial basis function (RBF) kernel*** whose value decreases along space radiating outward from $\boldsymbol{x}$

Parzen Windows and Kernels

- Binary KNN classifier:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i:\boldsymbol{x}^{(i)}\in\mathsf{KNN}(\boldsymbol{x})} y^{(i)}\right)$$

  - The "radius" of voter boundary depends on the input $\boldsymbol{x}$
- We can instead use the ***Parzen window*** with a fixed radius:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_i y^{(i)} 1(\boldsymbol{x}^{(i)}; \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\| \leq R)\right)$$

- Parzen windows also replace the hard boundary with a soft one:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x})\right)$$

  - $k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ is a ***radial basis function (RBF) kernel*** whose value decreases along space radiating outward from $\boldsymbol{x}$

Common RBF Kernels

- How to act like soft K-NN?

Common RBF Kernels

- How to act like soft K-NN?
- Gaussian RBF kernel:

$$k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^{(i)} - \boldsymbol{x}; \boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

Common RBF Kernels

- How to act like soft K-NN?
- Gaussian RBF kernel:

$$k(\boldsymbol{x}^{(i)},\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}^{(i)} - \boldsymbol{x}; \boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

- Or simply

$$k(\boldsymbol{x}^{(i)},\boldsymbol{x}) = \exp\left(-\gamma \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|^2\right)$$

- $\gamma \geq 0$ (or σ^2) is a hyperparameter controlling the smoothness of f

Outline

1. **Non-Parametric Methods**
   - K-NN
   - Parzen Windows
   - Local Models
2. **Support Vector Machines**
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick

Locally Weighted Linear Regression

- In addition to the majority voting and average, we can define ***local models*** for lazy predictions

找出要的小範圍data後
可以不只用投票或平均

甚至可以套用model來做predict

Locally Weighted Linear Regression

- In addition to the majority voting and average, we can define ***local models*** for lazy predictions
- E.g., in (eager) linear regression, we find $\boldsymbol{w} \in \mathbb{R}^{D+1}$ that minimizes SSE:

$$\arg\min_{\boldsymbol{w}} \sum_i (y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)})^2$$

- Local model: to find $\boldsymbol{w}$ minimizing ***SSE local to the point $\boldsymbol{x}$ we want to predict***:

$$\arg\min_{\boldsymbol{w}} \sum_i k(\boldsymbol{x}^{(i)}, \boldsymbol{x})(y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)})^2$$

  - $k(\cdot,\cdot) \in \mathbb{R}$ is an RBF kernel

Outline

Kernel Machines

- ***Kernel machines***:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + c_0$$

- For example:
  - Parzen windows: $c_i = y^{(i)}$ and $c_0 = 0$
  - Locally weighted linear regression: $c_i = (y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)})^2$ and $c_0 = 0$

Kernel Machines

- *Kernel machines*:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + c_0$$

- For example:
  - Parzen windows: $c_i = y^{(i)}$ and $c_0 = 0$
  - Locally weighted linear regression: $c_i = (y^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)})^2$ and $c_0 = 0$
- The variable $\boldsymbol{c} \in \mathbb{R}^N$ can be learned in either an eager or lazy manner
- Pros: complex, but highly accurate if regularized well

Sparse Kernel Machines

- To make a prediction, we need to store ***all*** examples
- May be infeasible due to
  - Large dataset (N)
  - Time limit
  - Space limit

Sparse Kernel Machines

- To make a prediction, we need to store ***all*** examples
- May be infeasible due to
  - Large dataset (N)
  - Time limit
  - Space limit
- Can we make $\boldsymbol{c}$ ***sparse***?
  - I.e., to make $c_i \neq 0$ for only a small fraction of examples called ***support vectors***
- How?

Outline

Separating Hyperplane I

- Model: $\mathbb{F} = \{f : f(\boldsymbol{x};\boldsymbol{w},b) = \boldsymbol{w}^\top \boldsymbol{x} + b\}$
  - A collection of hyperplanes
- Prediction: $\hat{y} = \mathrm{sign}(f(\boldsymbol{x}))$

Separating Hyperplane I

- Model: $\mathbb{F} = \{f : f(\boldsymbol{x};\boldsymbol{w},b) = \boldsymbol{w}^\top \boldsymbol{x} + b\}$
  - A collection of hyperplanes
- Prediction: $\hat{y} = \text{sign}(f(\boldsymbol{x}))$
- Training: to find $\boldsymbol{w}$ and b such that

$$\begin{aligned} \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b \geq 0, &\quad \text{if } y^{(i)} = 1 \\ \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b \leq 0, &\quad \text{if } y^{(i)} = -1 \end{aligned}$$

Separating Hyperplane I

- Model: $\mathbb{F} = \{f : f(\boldsymbol{x};\boldsymbol{w},b) = \boldsymbol{w}^\top \boldsymbol{x} + b\}$
  - A collection of hyperplanes
- Prediction: $\hat{y} = \mathrm{sign}(f(\boldsymbol{x}))$
- Training: to find $\boldsymbol{w}$ and b such that

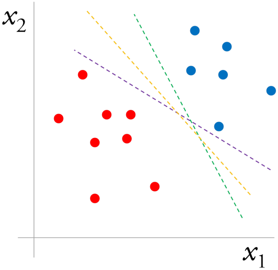
$$\begin{array}{l} \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b \geq 0, \quad \text{if } y^{(i)} = 1 \\ \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b \leq 0, \quad \text{if } y^{(i)} = -1 \end{array}$$

or simply

$$y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 0$$

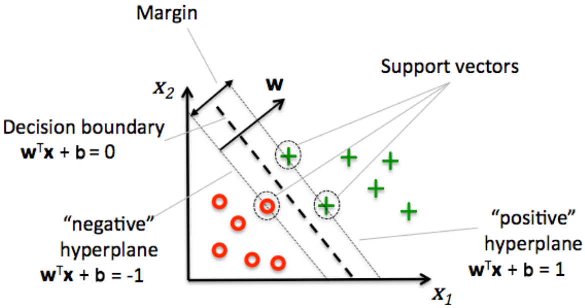
Separating Hyperplane II

- There are many feasible $\boldsymbol{w}$'s and b's when the classes are linearly separable
- Which hyperplane is the best?



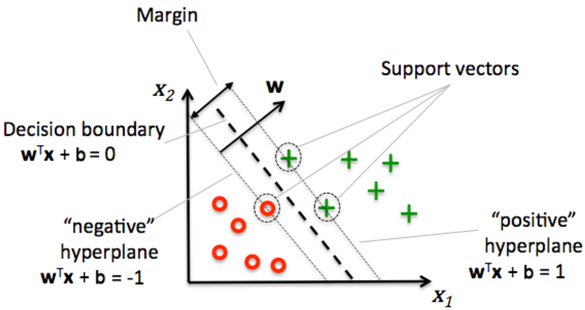
Support Vector Classification

- ***Support vector classifier*** (SVC) picks one with ***largest margin***:
  - $y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq a$ for all i
  - Margin: $2a/\|\boldsymbol{w}\|$ [Homework]



Support Vector Classification

- ***Support vector classifier*** (SVC) picks one with ***largest margin***:
  - $y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq a$ for all i
  - Margin: $2a/\|\boldsymbol{w}\|$ [Homework]



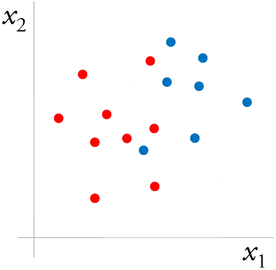
- With loss of generality, we let $a = 1$ and solve the problem:

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 1, \forall i$$

Outline

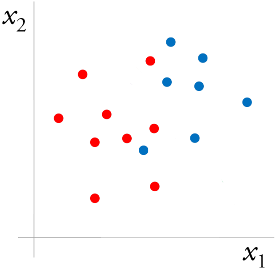
Overlapping Classes

- In practice, classes may be overlapping
  - Due to, e.g., noises or outliers



Overlapping Classes

- In practice, classes may be overlapping
  - Due to, e.g., noises or outliers



- The problem

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 1, \forall i$$

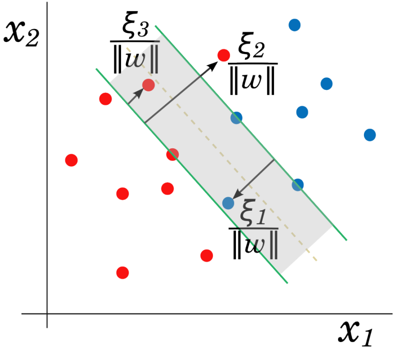
has no solution in this case. How to fix this?

Slacks

- SVC tolerates ***slacks*** that fall outside of the regions they ought to be
- Problem:

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i$$

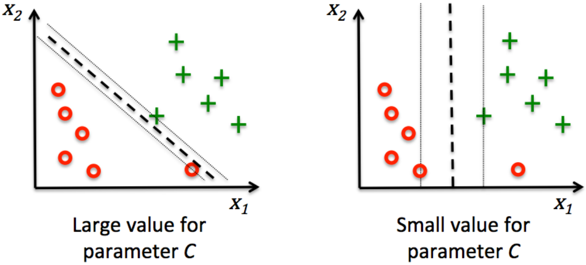
  - Favors large margin but also fewer slacks



Hyperparameter C

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

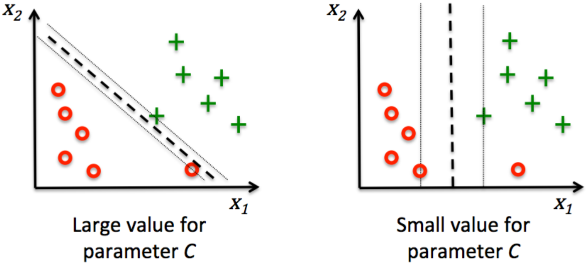
- The hyperparameter C controls the tradeoff between
  - Maximizing margin
  - Minimizing number of slacks



Hyperparameter C

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

- The hyperparameter C controls the tradeoff between
  - Maximizing margin
  - Minimizing number of slacks



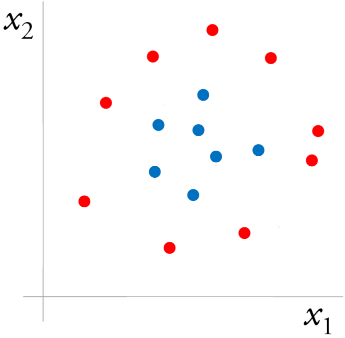
- Provides a geometric explanation to the weight decay

Outline

1. Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models
2. **Support Vector Machines**
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - Kernel Trick

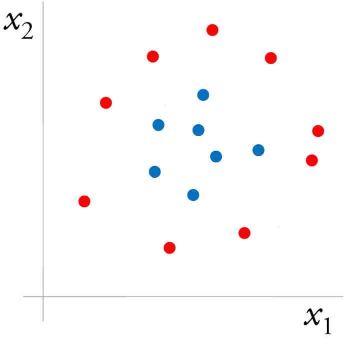
Nonlinearly Separable Classes

- In practice, classes may be nonlinearly separable



Nonlinearly Separable Classes

- In practice, classes may be nonlinearly separable



- SVC (with slacks) gives "bad" hyperplanes due to underfitting
- How to make it nonlinear?

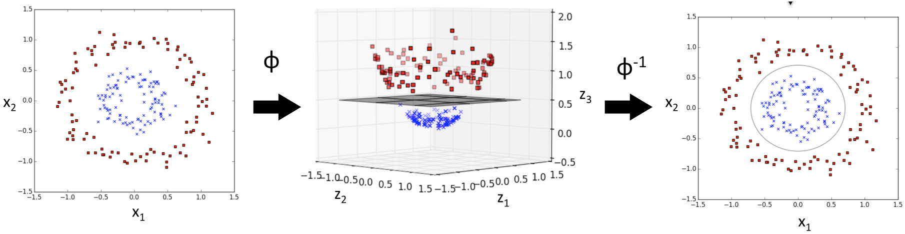
Feature Augmentation

- Recall that in polynomial regression, we augment data features to make a linear regressor nonlinear

Feature Augmentation

- Recall that in polynomial regression, we augment data features to make a linear regressor nonlinear
- We can can define a function $\Phi(\cdot)$ that maps each data point to a high dimensional space:

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i$$



Outline

Time Complexity

- Nonlinear SVC:

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i$$

- The higher augmented feature dimension, the more variables in $\boldsymbol{w}$ to solve

Time Complexity

- Nonlinear SVC:

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i$$

- The higher augmented feature dimension, the more variables in $\boldsymbol{w}$ to solve
- Can we solve $\boldsymbol{w}$ in time complexity that is independent with the mapped dimension?

Dual Problem

- Primal problem:

$$\begin{gathered}\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \\ \text{sibject to } y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i\end{gathered}$$

- Dual problem:

$$\begin{gathered}\arg\max_{\alpha,\beta} \min_{\boldsymbol{w},b,\xi} L(\boldsymbol{w},b,\xi,\alpha,\beta) \\ \text{subject to } \alpha \geq \mathbf{0}, \beta \geq \mathbf{0}\end{gathered}$$

where $L(\boldsymbol{w},b,\xi,\alpha,\beta) =$
$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$

Dual Problem

- Primal problem:

$$\arg\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$
$$\text{sibject to } y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \forall i$$

- Dual problem:

$$\arg\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} L(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})$$
$$\text{subject to } \boldsymbol{\alpha} \geq \boldsymbol{0}, \boldsymbol{\beta} \geq \boldsymbol{0}$$

where $L(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) =$
$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$

- Primal problem is convex, so ***strong duality*** holds

Solving Dual Problem I

- $L(\boldsymbol{w}, b, \xi, \alpha, \beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$
- The inner problem
  $$\min_{\boldsymbol{w}, b, \xi} L(\boldsymbol{w}, b, \xi, \alpha, \beta)$$
  is convex in terms of $\boldsymbol{w}$, b, and ξ
- Let's solve it analytically:

Solving Dual Problem I

- $L(\boldsymbol{w},b,\xi,\alpha,\beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1-y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)-\xi_i) + \sum_i \beta_i(-\xi_i)$
- The inner problem
  $$\min_{\boldsymbol{w},b,\xi} L(\boldsymbol{w},b,\xi,\alpha,\beta)$$
  is convex in terms of $\boldsymbol{w}$, b, and ξ
- Let's solve it analytically:
- $\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y^{(i)}\Phi(\boldsymbol{x}^{(i)}) = \boldsymbol{0} \Rightarrow \boldsymbol{w} = \sum_i \alpha_i y^{(i)}\Phi(\boldsymbol{x}^{(i)})$

Solving Dual Problem I

- $L(\boldsymbol{w}, b, \xi, \alpha, \beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$
- The inner problem
  $$\min_{\boldsymbol{w}, b, \xi} L(\boldsymbol{w}, b, \xi, \alpha, \beta)$$
  is convex in terms of $\boldsymbol{w}$, b, and ξ
- Let's solve it analytically:
- $\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)}) = \boldsymbol{0} \Rightarrow \boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$
- $\frac{\partial L}{\partial b} = \sum_i \alpha_i y^{(i)} = 0$

Solving Dual Problem I

- $L(\boldsymbol{w}, b, \xi, \alpha, \beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$
- The inner problem
$$\min_{\boldsymbol{w}, b, \xi} L(\boldsymbol{w}, b, \xi, \alpha, \beta)$$
  is convex in terms of $\boldsymbol{w}$, b, and ξ
- Let's solve it analytically:
- $\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)}) = \boldsymbol{0} \Rightarrow \boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$
- $\frac{\partial L}{\partial b} = \sum_i \alpha_i y^{(i)} = 0$
- $\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \Rightarrow \beta_i = C - \alpha_i$

Solving Dual Problem II

- $L(\boldsymbol{w},b,\xi,\alpha,\beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$
- Substituting $\boldsymbol{w} = \sum_i \alpha_i y^{(i)}\Phi(\boldsymbol{x}^{(i)})$ and $\beta_i = C - \alpha_i$ in $L(\boldsymbol{w},b,\xi,\alpha,\beta)$:

$$L(\boldsymbol{w},b,\xi,\alpha,\beta) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}\Phi(\boldsymbol{x}^{(i)})^\top\Phi(\boldsymbol{x}^{(j)}) - b\sum_i \alpha_i y^{(i)},$$

Solving Dual Problem II

- $L(\boldsymbol{w}, b, \xi, \alpha, \beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$
- Substituting $\boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$ and $\beta_i = C - \alpha_i$ in $L(\boldsymbol{w}, b, \xi, \alpha, \beta)$:

$$L(\boldsymbol{w}, b, \xi, \alpha, \beta) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)}) - b\sum_i \alpha_i y^{(i)},$$

$$\min_{\boldsymbol{w}, b, \xi} L(\boldsymbol{w}, b, \xi, \alpha, \beta) = \begin{cases} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)}) \ , & \text{if } \sum_i \alpha_i y^{(i)} = 0, \\ -\infty, & \text{otherwise} \end{cases}$$

Solving Dual Problem II

- $L(\boldsymbol{w},b,\xi,\alpha,\beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_i \xi_t+\sum_i \alpha_i(1-y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)-\xi_i)+\sum_i \beta_i(-\xi_i)$
- Substituting $\boldsymbol{w}=\sum_i \alpha_i y^{(i)}\Phi(\boldsymbol{x}^{(i)})$ and $\beta_i = C-\alpha_i$ in $L(\boldsymbol{w},b,\xi,\alpha,\beta)$:

$$L(\boldsymbol{w},b,\xi,\alpha,\beta)=\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}\Phi(\boldsymbol{x}^{(i)})^\top\Phi(\boldsymbol{x}^{(j)}) - b\sum_i \alpha_i y^{(i)},$$

$$\min_{\boldsymbol{w},b,\xi} L(\boldsymbol{w},b,\xi,\alpha,\beta)=\begin{cases}\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^{(i)}y^{(j)}\Phi(\boldsymbol{x}^{(i)})^\top\Phi(\boldsymbol{x}^{(j)}) \;, & \text{if } \sum_i \alpha_i y^{(i)}=0,\\ -\infty, & \text{otherwise}\end{cases}$$

- Outer maximization problem:

$$\arg\max_\alpha \mathbf{1}^\top\alpha - \tfrac{1}{2}\alpha^\top \boldsymbol{K}\alpha$$
$$\text{subject to } \mathbf{0}\le\alpha\le C\mathbf{1} \text{ and } \boldsymbol{y}^\top\alpha=0$$

  - $K_{i,j}=y^{(i)}y^{(j)}\Phi(\boldsymbol{x}^{(i)})^\top\Phi(\boldsymbol{x}^{(j)})$

Solving Dual Problem II

- $L(\boldsymbol{w},b,\xi,\alpha,\beta) =$
  $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_t + \sum_i \alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) + \sum_i \beta_i(-\xi_i)$
- Substituting $\boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$ and $\beta_i = C - \alpha_i$ in $L(\boldsymbol{w},b,\xi,\alpha,\beta)$:

$$L(\boldsymbol{w},b,\xi,\alpha,\beta) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)}) - b\sum_i \alpha_i y^{(i)},$$

$$\min_{\boldsymbol{w},b,\xi} L(\boldsymbol{w},b,\xi,\alpha,\beta) = \begin{cases} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)}) \ , & \text{if } \sum_i \alpha_i y^{(i)} = 0, \\ -\infty, & \text{otherwise} \end{cases}$$

- Outer maximization problem:

$$\arg\max_\alpha \mathbf{1}^\top \alpha - \tfrac{1}{2}\alpha^\top \boldsymbol{K} \alpha$$
$$\text{subject to } \mathbf{0} \leq \alpha \leq C\mathbf{1} \text{ and } \boldsymbol{y}^\top \alpha = 0$$

  - $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - $\beta_i = C - \alpha_i \geq 0$ implies $\alpha_i \leq C$

Solving Dual Problem II

- Dual minimization problem of SVC:

$$\arg\min_{\alpha} \frac{1}{2}\alpha^\top \boldsymbol{K}\alpha - \mathbf{1}^\top \alpha$$
$$\text{subject to } \mathbf{0} \leq \alpha \leq C\mathbf{1} \text{ and } \boldsymbol{y}^\top \alpha = 0$$

- Number of variables to solve?

Solving Dual Problem II

- Dual minimization problem of SVC:

$$\arg\min_{\alpha} \frac{1}{2}\alpha^\top \boldsymbol{K}\alpha - \mathbf{1}^\top \alpha$$
$$\text{subject to } \mathbf{0} \leq \alpha \leq C\mathbf{1} \text{ and } \boldsymbol{y}^\top \alpha = 0$$

- Number of variables to solve? N instead of augmented feature dimension

Solving Dual Problem II

- Dual minimization problem of SVC:

$$\arg\min_{\alpha} \frac{1}{2}\alpha^\top \boldsymbol{K}\alpha - \mathbf{1}^\top \alpha$$
$$\text{subject to } \mathbf{0} \leq \alpha \leq C\mathbf{1} \text{ and } \boldsymbol{y}^\top \alpha = 0$$

- Number of variables to solve? N instead of augmented feature dimension
- In practice, this problem is solved by specialized solvers such as the sequential minimal optimization (SMO) [3]
  - As $\boldsymbol{K}$ is usually ill-conditioned

Making Predictions

- Prediction: $\hat{y} = \text{sign}(f(\boldsymbol{x})) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$

Making Predictions

- Prediction: $\hat{y} = \mathrm{sign}(f(\boldsymbol{x})) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- We have $\boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$
- How to obtain b?

Making Predictions

- Prediction: $\hat{y} = \mathrm{sign}(f(\boldsymbol{x})) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- We have $\boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$
- How to obtain b?
- By the complementary slackness of KKT conditions, we have:

$$\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0 \text{ and } \beta_i(-\xi_i) = 0$$

Making Predictions

- Prediction: $\hat{y} = \mathrm{sign}(f(\boldsymbol{x})) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- We have $\boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$
- How to obtain b?
- By the complementary slackness of KKT conditions, we have:

$$\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0 \text{ and } \beta_i(-\xi_i) = 0$$

- For any $\boldsymbol{x}^{(i)}$ having $0 < \alpha_i < C$, we have

$$\beta_i = C - \alpha_i > 0 \Rightarrow \xi_i = 0,$$

$$(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0 \Rightarrow b = y^{(i)} - \boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)})$$

Making Predictions

- Prediction: $\hat{y} = \text{sign}(f(\boldsymbol{x})) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- We have $\boldsymbol{w} = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})$
- How to obtain b?
- By the complementary slackness of KKT conditions, we have:

$$\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0 \text{ and } \beta_i(-\xi_i) = 0$$

- For any $\boldsymbol{x}^{(i)}$ having $0 < \alpha_i < C$, we have

$$\beta_i = C - \alpha_i > 0 \Rightarrow \xi_i = 0,$$

$$(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0 \Rightarrow b = y^{(i)} - \boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)})$$

- In practice, we usually take the average over ***all*** $\boldsymbol{x}^{(i)}$'s having $0 < \alpha_i < C$ to avoid numeric error

Outline

1. Non-Parametric Methods
   - K-NN
   - Parzen Windows
   - Local Models
2. **Support Vector Machines**
   - SVC
   - Slacks
   - Nonlinear SVC
   - Dual Problem
   - **Kernel Trick**

Kernel as Inner Product

- We need to evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - Making a prediction, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) + b$

Kernel as Inner Product

- We need to evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - Making a prediction, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) + b$
- Time complexity?

Kernel as Inner Product

- We need to evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - Making a prediction, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) + b$
- Time complexity?
- If we choose Φ carefully, we can can evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) = k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ efficiently

Kernel as Inner Product

- We need to evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - Making a prediction, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) + b$
- Time complexity?
- If we choose Φ carefully, we can can evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) = k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ efficiently
- Polynomial kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}^\top \boldsymbol{b}/\alpha + \beta)^\gamma$
  - E.g., let $\alpha = 1$, $\beta = 1$, $\gamma = 2$ and $\boldsymbol{a} \in \mathbb{R}^2$, then $\Phi(\boldsymbol{a}) = [1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1 a_2]^\top \in \mathbb{R}^6$

Kernel as Inner Product

- We need to evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - Making a prediction, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) + b$
- Time complexity?
- If we choose Φ carefully, we can can evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) = k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ efficiently
- Polynomial kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}^\top \boldsymbol{b}/\alpha + \beta)^\gamma$
  - E.g., let $\alpha = 1$, $\beta = 1$, $\gamma = 2$ and $\boldsymbol{a} \in \mathbb{R}^2$, then $\Phi(\boldsymbol{a}) = [1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1 a_2]^\top \in \mathbb{R}^6$
- Gaussian RBF kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \|\boldsymbol{a} - \boldsymbol{b}\|^2)$, $\gamma \geq 0$
  - $k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \|\boldsymbol{a}\|^2 + 2\gamma \boldsymbol{a}^\top \boldsymbol{b} - \gamma \|\boldsymbol{b}\|^2) = \exp(-\gamma \|\boldsymbol{a}\|^2 - \gamma \|\boldsymbol{b}\|^2)(1 + \frac{2\gamma \boldsymbol{a}^\top \boldsymbol{b}}{1!} + \frac{(2\gamma \boldsymbol{a}^\top \boldsymbol{b})^2}{2!} + \cdots)$
  - Let $\boldsymbol{a} \in \mathbb{R}^2$, then $\Phi(\boldsymbol{a}) = \exp(-\gamma \|\boldsymbol{a}\|^2)[1, \sqrt{\frac{2\gamma}{1!}} a_1, \sqrt{\frac{2\gamma}{1!}} a_2, \sqrt{\frac{2\gamma}{2!}} a_1^2, \sqrt{\frac{2\gamma}{2!}} a_2^2, 2\sqrt{\frac{\gamma}{2!}} a_1 a_2, \cdots]^\top \in \mathbb{R}^\infty$

Kernel as Inner Product

- We need to evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$ when
  - Solving dual problem of SVC, where $K_{i,j} = y^{(i)} y^{(j)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}^{(j)})$
  - Making a prediction, where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_i \alpha_i y^{(i)} \Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) + b$
- Time complexity?
- If we choose Φ carefully, we can can evaluate $\Phi(\boldsymbol{x}^{(i)})^\top \Phi(\boldsymbol{x}) = k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ efficiently
- Polynomial kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}^\top \boldsymbol{b}/\alpha + \beta)^\gamma$
  - E.g., let $\alpha = 1$, $\beta = 1$, $\gamma = 2$ and $\boldsymbol{a} \in \mathbb{R}^2$, then $\Phi(\boldsymbol{a}) = [1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1 a_2]^\top \in \mathbb{R}^6$
- Gaussian RBF kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \|\boldsymbol{a} - \boldsymbol{b}\|^2)$, $\gamma \geq 0$
  - $k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \|\boldsymbol{a}\|^2 + 2\gamma \boldsymbol{a}^\top \boldsymbol{b} - \gamma \|\boldsymbol{b}\|^2) = \exp(-\gamma \|\boldsymbol{a}\|^2 - \gamma \|\boldsymbol{b}\|^2)(1 + \frac{2\gamma \boldsymbol{a}^\top \boldsymbol{b}}{1!} + \frac{(2\gamma \boldsymbol{a}^\top \boldsymbol{b})^2}{2!} + \cdots)$
  - Let $\boldsymbol{a} \in \mathbb{R}^2$, then $\Phi(\boldsymbol{a}) = \exp(-\gamma \|\boldsymbol{a}\|^2)[1, \sqrt{\frac{2\gamma}{1!}}a_1, \sqrt{\frac{2\gamma}{1!}}a_2, \sqrt{\frac{2\gamma}{2!}}a_1^2, \sqrt{\frac{2\gamma}{2!}}a_2^2, 2\sqrt{\frac{\gamma}{2!}}a_1 a_2, \cdots]^\top \in \mathbb{R}^\infty$

Kernel Trick

- If we choose Φ induced by Polynomial or Gaussian RBF kernel, then

$$K_{i,j} = y^{(i)} y^{(j)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$$

takes only $O(D)$ time to evaluate, and

$$f(\boldsymbol{x}) = \sum_i \alpha_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b$$

takes $O(ND)$ time

Kernel Trick

- If we choose Φ induced by Polynomial or Gaussian RBF kernel, then

$$K_{i,j} = y^{(i)} y^{(j)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$$

  takes only $O(D)$ time to evaluate, and

$$f(\boldsymbol{x}) = \sum_i \alpha_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b$$

  takes $O(ND)$ time
- Independent with the augmented feature dimension

Kernel Trick

- If we choose Φ induced by Polynomial or Gaussian RBF kernel, then

$$K_{i,j} = y^{(i)} y^{(j)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x})$$

  takes only $O(D)$ time to evaluate, and

$$f(\boldsymbol{x}) = \sum_i \alpha_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b$$

  takes $O(ND)$ time
- Independent with the augmented feature dimension
- α, β, and γ are new hyperparameters

Sparse Kernel Machines

- SVC is a kernel machine:

$$f(\boldsymbol{x}) = \sum_i \alpha_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b$$

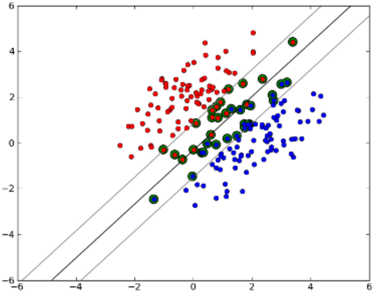
- It is surprising that SVC works like K-NN in some sense

Sparse Kernel Machines

- SVC is a kernel machine:

$$f(\boldsymbol{x}) = \sum_i \alpha_i y^{(i)} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b$$

- It is surprising that SVC works like K-NN in some sense
- However, SVC is a ***sparse*** kernel machine
- Only the ***slacks*** become the support vectors ($\alpha_i > 0$)



KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$
  - Complementary slackness: $\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0$ and $\beta_i(-\xi_i) = 0$

KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$
  - Complementary slackness: $\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0$ and $\beta_i(-\xi_i) = 0$
- Depending on the value of α_i, each example $\boldsymbol{x}^{(i)}$ can be:

KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)\geq 1-\xi_i$ and $\xi_i\geq 0$
  - Complementary slackness: $\alpha_i(1-y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)-\xi_i)=0$ and $\beta_i(-\xi_i)=0$
- Depending on the value of α_i, each example $\boldsymbol{x}^{(i)}$ can be:

KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$
  - Complementary slackness: $\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0$ and $\beta_i(-\xi_i) = 0$
- Depending on the value of α_i, each example $\boldsymbol{x}^{(i)}$ can be:
- Non SVs ($\alpha_i = 0$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1$ (usually strict)
- ***Free SVs*** ($0 < \alpha_i < C$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) = 1$
- ***Bounded SVs*** ($\alpha_i = C$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \leq 1$ (usually strict)

KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$
  - Complementary slackness: $\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0$ and $\beta_i(-\xi_i) = 0$
- Depending on the value of α_i, each example $\boldsymbol{x}^{(i)}$ can be:
- Non SVs ($\alpha_i = 0$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1$ (usually strict)
  - $1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i \leq 0$
  - Since $\beta_i = C - \alpha_i \neq 0$, we have $\xi_i = 0$
- ***Free SVs*** ($0 < \alpha_i < C$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) = 1$
- ***Bounded SVs*** ($\alpha_i = C$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \leq 1$ (usually strict)

KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b) \geq 1-\xi_i$ and $\xi_i \geq 0$
  - Complementary slackness: $\alpha_i(1-y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)-\xi_i) = 0$ and $\beta_i(-\xi_i) = 0$
- Depending on the value of α_i, each example $\boldsymbol{x}^{(i)}$ can be:
- Non SVs ($\alpha_i = 0$): $y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b) \geq 1$ (usually strict)
  - $1-y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)-\xi_i \leq 0$
  - Since $\beta_i = C-\alpha_i \neq 0$, we have $\xi_i = 0$
- ***Free SVs*** ($0 < \alpha_i < C$): $y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b) = 1$
  - $1-y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b)-\xi_i = 0$
  - Since $\beta_i = C-\alpha_i \neq 0$, we have $\xi_i = 0$
- ***Bounded SVs*** ($\alpha_i = C$): $y^{(i)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(i)})+b) \leq 1$ (usually strict)

KKT Conditions and Types of SVs

- By KKT conditions, we have:
  - Primal feasibility: $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$
  - Complementary slackness: $\alpha_i(1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i) = 0$ and $\beta_i(-\xi_i) = 0$
- Depending on the value of α_i, each example $\boldsymbol{x}^{(i)}$ can be:
- Non SVs ($\alpha_i = 0$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \geq 1$ (usually strict)
  - $1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i \leq 0$
  - Since $\beta_i = C - \alpha_i \neq 0$, we have $\xi_i = 0$
- ***Free SVs*** ($0 < \alpha_i < C$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) = 1$
  - $1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i = 0$
  - Since $\beta_i = C - \alpha_i \neq 0$, we have $\xi_i = 0$
- ***Bounded SVs*** ($\alpha_i = C$): $y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) \leq 1$ (usually strict)
  - $1 - y^{(i)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(i)}) + b) - \xi_i = 0$
  - Since $\beta_i = 0$, we have $\xi_i \geq 0$

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set
- Cons:
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set
- Cons:
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $\boldsymbol{K}$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $\boldsymbol{K}$ in memory

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set
- Cons:
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $\boldsymbol{K}$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $\boldsymbol{K}$ in memory
  - Sensitive to irrelevant data features (vs. decision trees)

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set
- Cons:
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $\boldsymbol{K}$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $\boldsymbol{K}$ in memory
  - Sensitive to irrelevant data features (vs. decision trees)
  - Non-trivial hyperparameter tuning
    - The effect of a (C, γ) combination is unknown in advance
    - Usually done by ***grid search***

Remarks I

- Pros of SVC:
  - Global optimality (convex problem)
  - Works with different kernels (linear, Polynomial, Gaussian RBF, etc.)
  - Works well with small training set
- Cons:
  - Nonlinear SVC not scalable to large tasks
    - Takes $O(N^2) \sim O(N^3)$ time to train using SMO in LIBSVM [1]
    - On the other hand, linear SVC takes $O(ND)$ time
  - Kernel matrix $\boldsymbol{K}$ requires $O(N^2)$ space
    - In practice, we cache only a small portion of $\boldsymbol{K}$ in memory
  - Sensitive to irrelevant data features (vs. decision trees)
  - Non-trivial hyperparameter tuning
    - The effect of a (C, γ) combination is unknown in advance
    - Usually done by ***grid search***
  - Separate only 2 classes
    - Usually wrapped by the 1-vs-1 technique for multi-class classification

Remarks II

- Does nonlinear SVC always perform better than linear SVC?

Remarks II

- Does nonlinear SVC always perform better than linear SVC? ***No***
- Choose linear SVC (e.g., LIBLINEAR [2]) when
  - N is large (since nonlinear SVC does not scale), or
  - D is large (since classes may already be linearly separable)

Reference I

[1] Chih-Chung Chang and Chih-Jen Lin.
Libsvm: a library for support vector machines.
ACM Transactions on Intelligent Systems and Technology (TIST), 2(3):27, 2011.

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin.
Liblinear: A library for large linear classification.
Journal of machine learning research, 9(Aug):1871–1874, 2008.

[3] John Platt et al.
Sequential minimal optimization: A fast algorithm for training support vector machines.
1998.