

Linear Classification and Support Vector Machine and Stochastic Gradient Descent and Multi-class Classification

Prof. Mingkui Tan

SCUT Machine Intelligence Laboratory (SMIL)



Contents

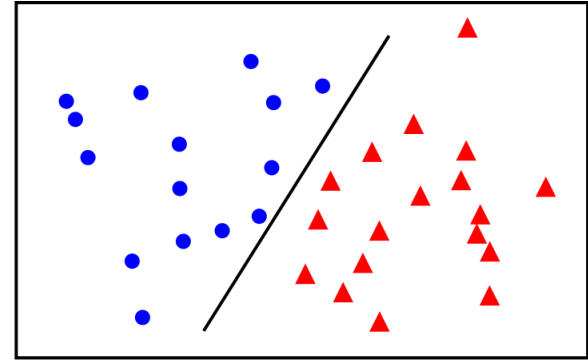
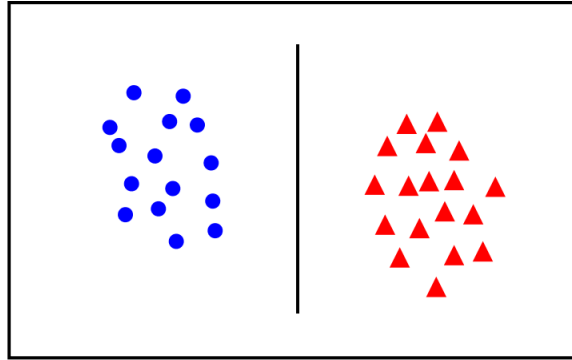
- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Contents

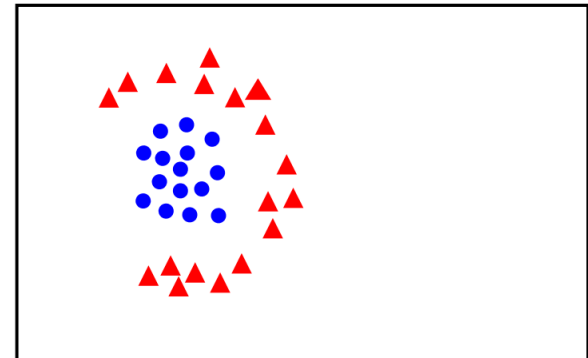
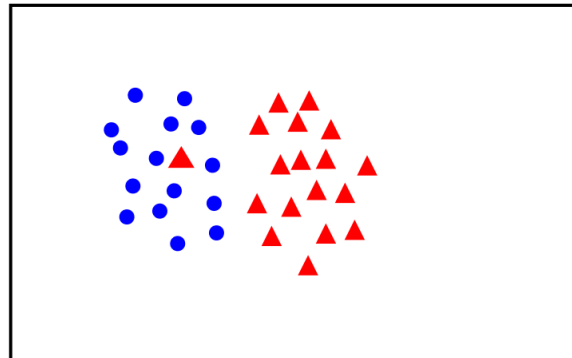
- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Linear Separability

Linearly
separable



Not linearly
separable

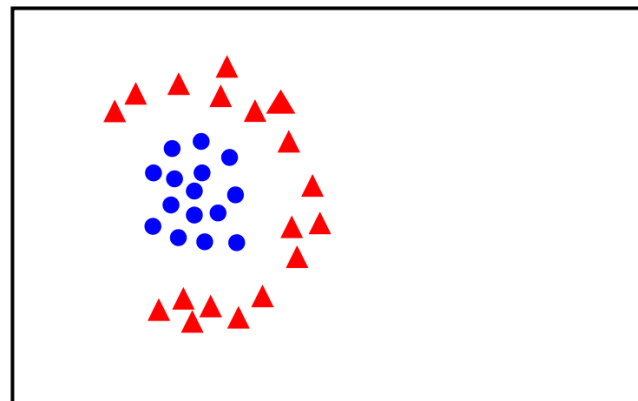
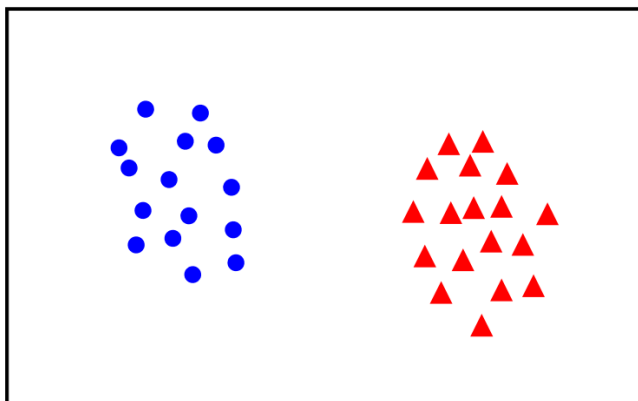


Binary Classification

■ Given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$, learn a classifier $f(x)$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

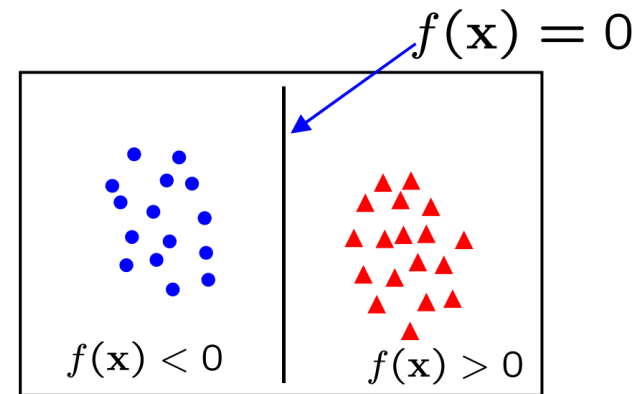
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification



Linear Classifiers: 2D Example

A linear classifier has the form:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

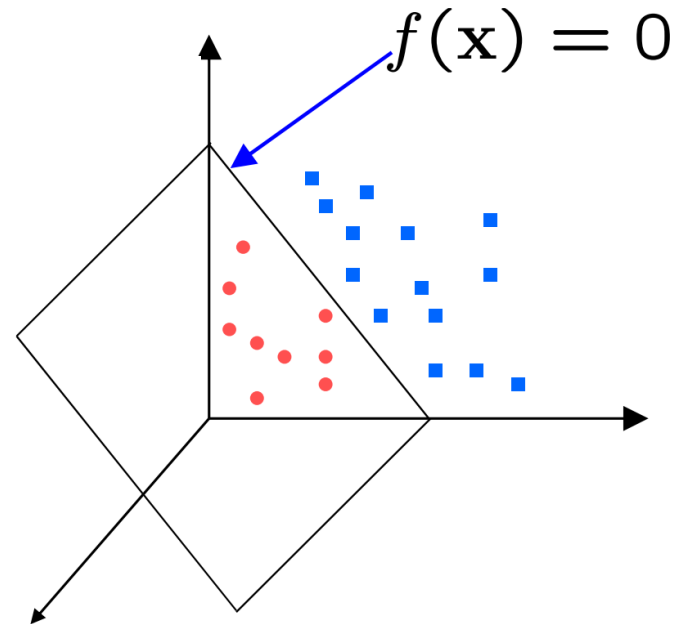


- In 2-D, the discriminant is a line
- \mathbf{w} is the **normal** to the line, and b is the **bias**
- \mathbf{w} is known as the **weight vector**

Linear Classifiers: 3D Example

A linear classifier has the form:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

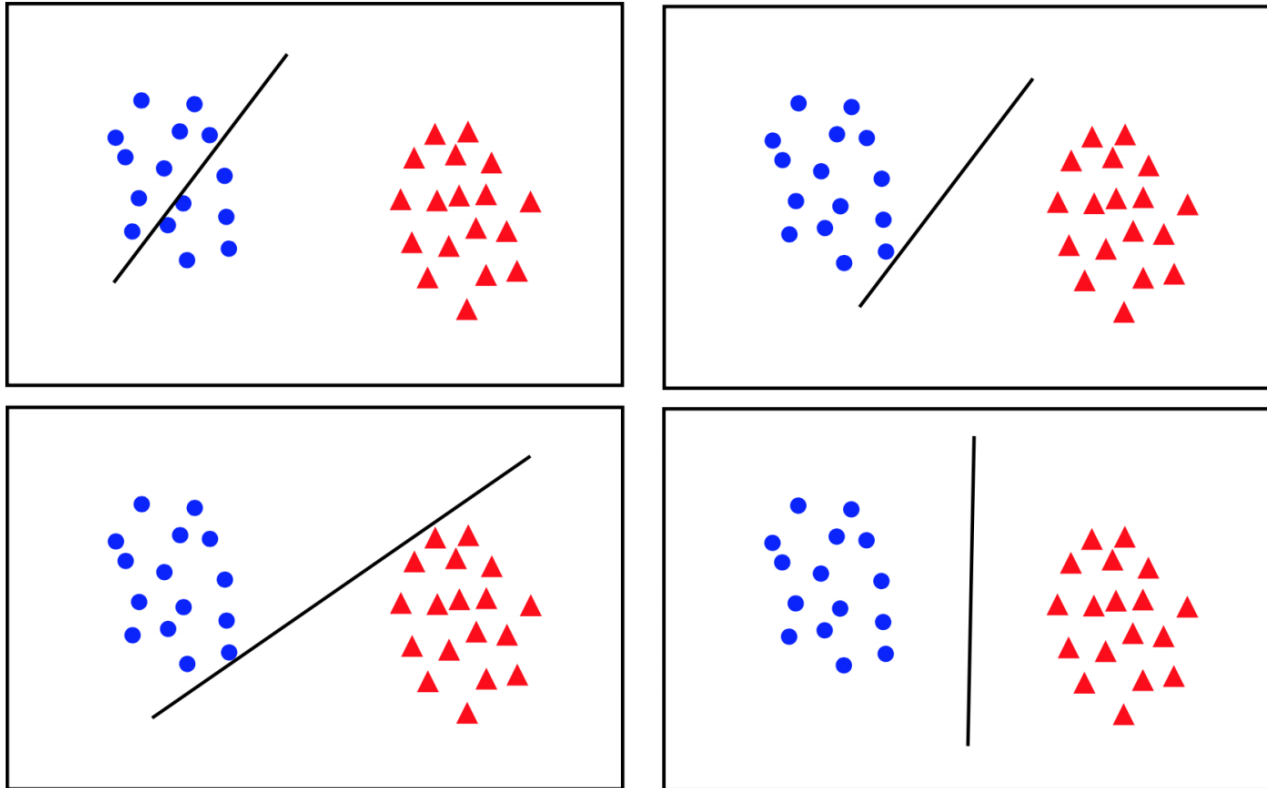


- In 3-D the discriminant is a **plane**, and in m-D it is a **hyperplane**

Contents

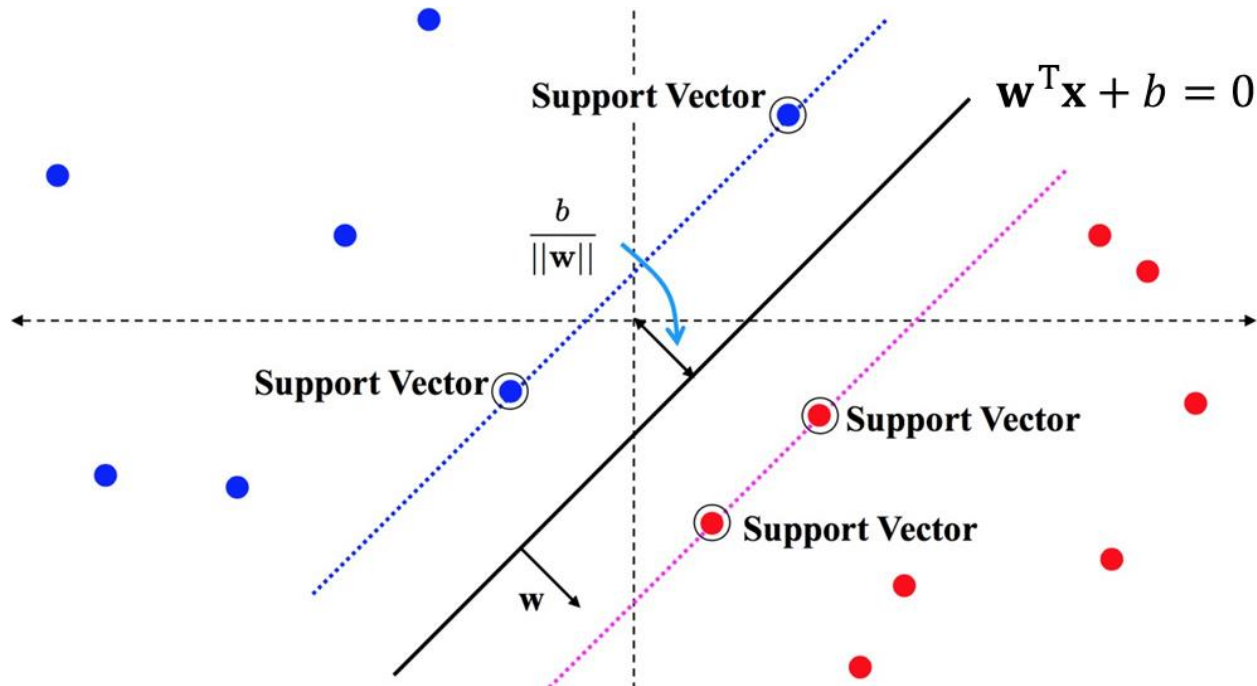
- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

What's Good Decision Boundary?



- **Maximum margin** solution: most stable under perturbations of the input

Max-margin Methods



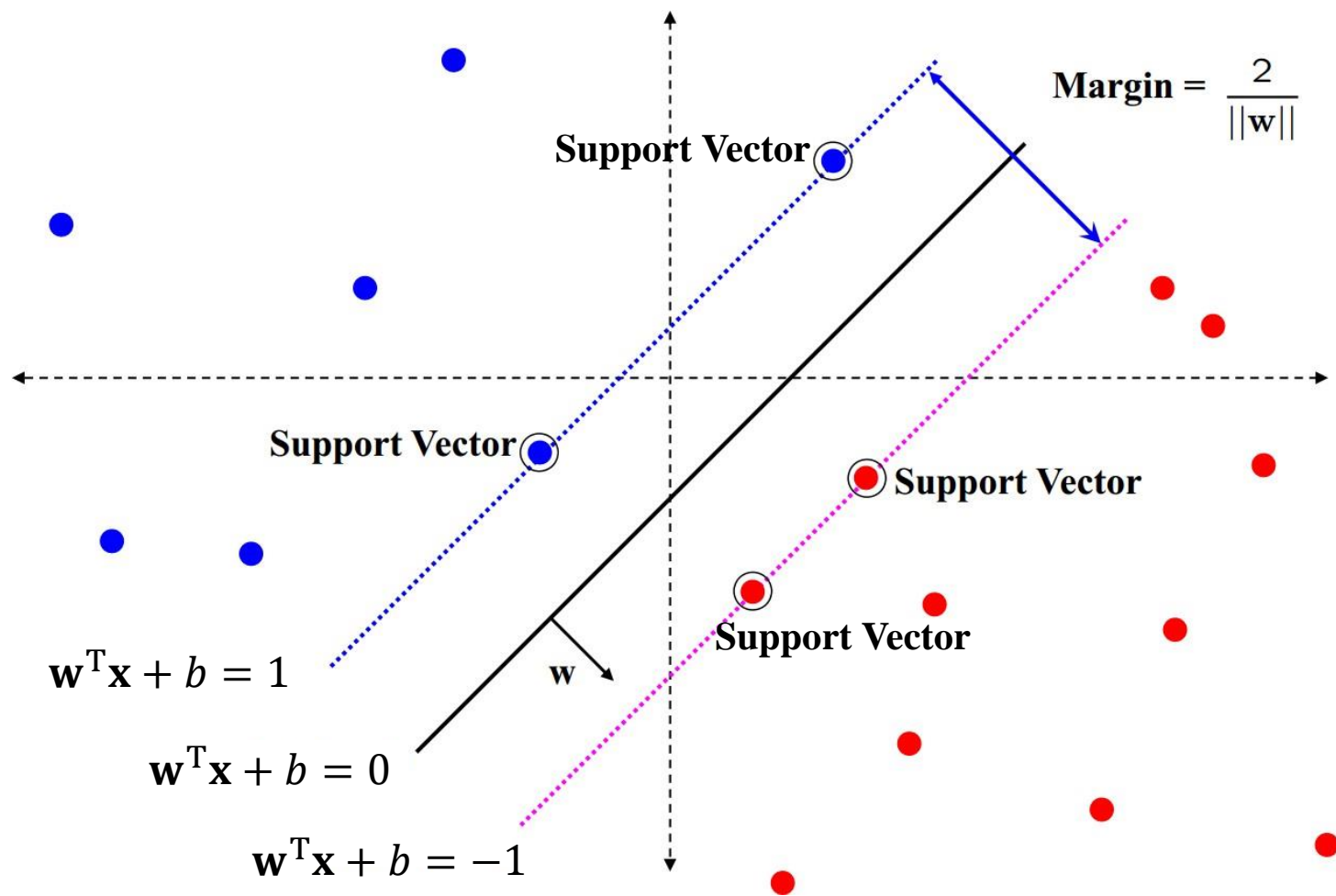
- Select two parallel hyperplanes that separate the two classes of data and let the distance between them as large as possible
- The region bounded by these two hyperplanes is called the “margin”

SVM-sketch Derivation

- Choose normalization such that $\mathbf{w}^T \mathbf{x}_+ + b = +1$ and $\mathbf{w}^T \mathbf{x}_- + b = -1$ for the **positive** and **negative** support vectors respectively
- Then, the **margin** is given by

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^T (\mathbf{x}_+ - \mathbf{x}_-)}{\|\mathbf{w}\|} = \frac{(1 - b) - (-1 - b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Support Vector Machine



Basic Support Vector Machine

- SVM can be formulated as an optimization:

$$\max_{w,b} \frac{2}{||\mathbf{w}'||}$$

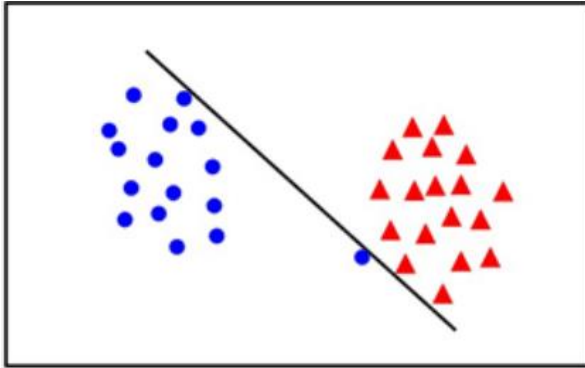
$$s.t. \quad \mathbf{w}'^T \mathbf{x}_i + b \begin{cases} \geq 1 & y_i = +1 \\ \leq -1 & y_i = -1 \end{cases}$$

- Or equivalently:

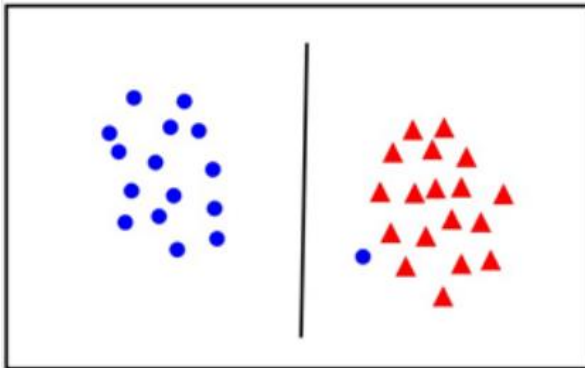
$$\min_{w,b} \frac{||\mathbf{w}'||^2}{2}$$

$$s.t. \quad y_i (\mathbf{w}'^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, 3, \dots, n$$

Linear Separability Again: What is Outlier?



The point can be linearly separated but the margin is very narrow

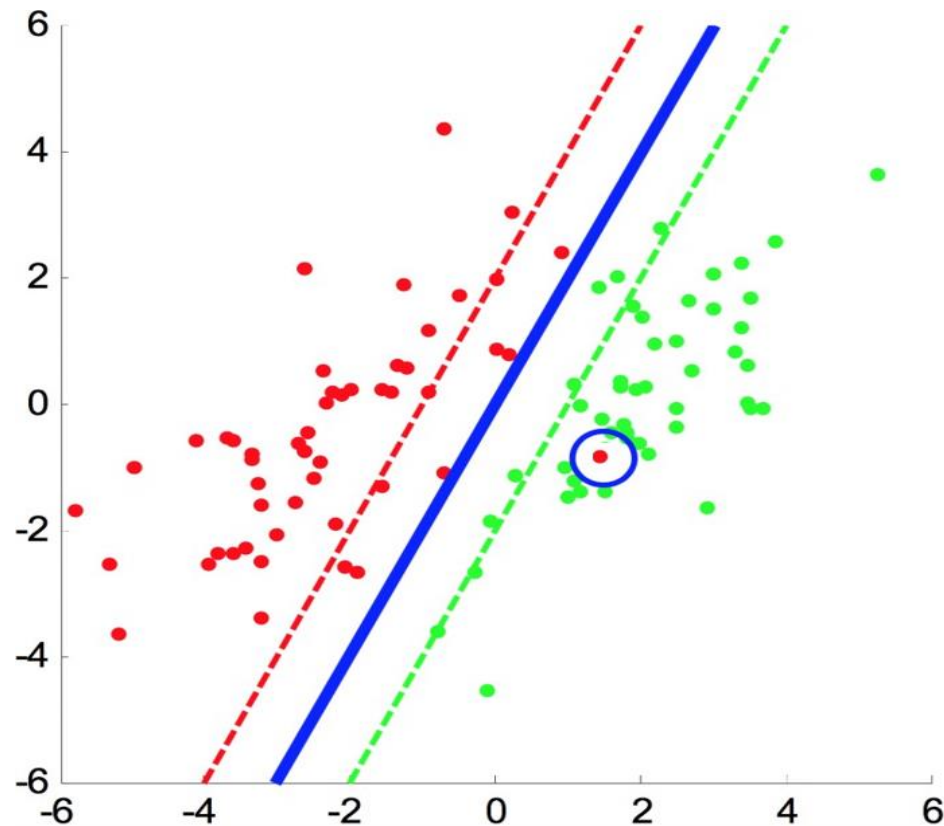


The large margin solution is better, even though one sample point cannot satisfy the constraint

- In general, there exists a **trade-off** between the margin and the number of mistakes on the training data

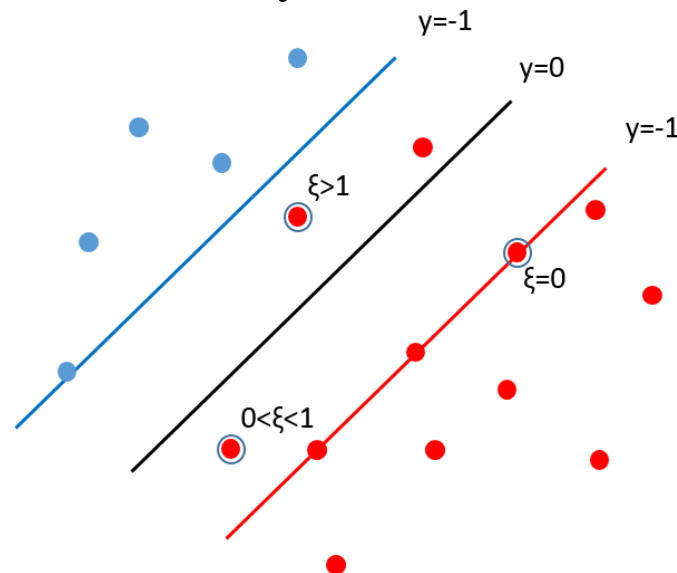
Linear Separability Again: What is Outlier?

- Moreover, training data **may not be linearly separable**



Relaxed Formulation

- Introduce slack variable $\xi_i \geq 0$ for each i , which represents how much example i is on the wrong side of margin boundary
- $\xi_i = 0$ means the basic formulation of SVM works well
- $0 < \xi_i < 1$ means it is correctly classified, but with a smaller margin than $\frac{2}{\|\mathbf{w}\|}$
- $\xi_i \geq 1$ means it is incorrectly classified



Soft Margin Formulation

- The optimization problem becomes:

$$\min_{\mathbf{w}, b} \frac{||\mathbf{w}||^2}{2} + \frac{C}{n} \sum_{i=1}^n \xi_i$$

$$s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall \xi_i \geq 0, \quad i = 1, 2, \dots, n$$

where C is a hyperparameter:

- small C makes constraints easy to be ignored
- large C makes constraints hard to be ignored

Soft Margin Formulation

- As can be seen from above:

$$\xi_i \geq 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b), \quad \forall \xi_i \geq 0, \quad i = 1, 2, \dots, n$$

- What does ξ_i value?

$$\text{if } 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \text{ then } \xi_i = 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)$$

$$\text{if } 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0, \text{ then } \xi_i = 0$$

- Rewrite ξ_i as:

$$\xi_i = \max\left(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)\right)$$

Hinge Loss

- Hinge loss:

$$\xi_i = \max\left(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)\right)$$

- The optimization problem becomes:

$$\min_{\mathbf{w}, b} \frac{||\mathbf{w}||^2}{2} + \frac{C}{n} \sum_{i=1}^n \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

Contents

- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Gradient Descent

- Gradient descent minimum optimization problem:

$$\min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b) = \frac{||\mathbf{w}||^2}{2} + \frac{C}{n} \sum_{i=1}^n \max\left(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)\right)$$

- To minimize a loss function $\mathcal{L}(\mathbf{w}, b)$, use the iterative update:

$$\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b)$$

$$b = b - \eta \nabla_b \mathcal{L}(\mathbf{w}, b)$$

where η is the learning rate

Gradient Descent

■ Let $g_{\mathbf{w}}(\mathbf{x}_i) = \frac{\partial \xi_i}{\partial \mathbf{w}}$, $g_b(\mathbf{x}_i) = \frac{\partial \xi_i}{\partial b}$

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b) = \mathbf{w} + \frac{C}{n} \sum_{i=1}^n g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\nabla_b \mathcal{L}(\mathbf{w}, b) = \frac{C}{n} \sum_{i=1}^n g_b(\mathbf{x}_i)$$

Algorithm 1: GD

```
1 Initialize parameter  $\mathbf{w}$  and learning rate  $\eta$ 
2 while stopping condition is not achieved do
3      $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b)$ 
4      $b = b - \eta \nabla_b \mathcal{L}(\mathbf{w}, b)$ 
5 end
```

Gradient Computation

- Gradient descent minimum optimization problem:

$$\min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b) = \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{n} \sum_{i=1}^n \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

- Gradient computation :

$$\nabla \mathcal{L} = \begin{bmatrix} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b) \\ \nabla_b \mathcal{L}(\mathbf{w}, b) \end{bmatrix}$$

Gradient Computation

- The hinge loss is $\xi_i = \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$
- Let $g_{\mathbf{w}}(\mathbf{x}_i) = \frac{\partial \xi_i}{\partial \mathbf{w}}$
- If $1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0$:

$$\begin{aligned} g_{\mathbf{w}}(\mathbf{x}_i) &= \frac{\partial(-y_i(\mathbf{w}^T \mathbf{x}_i + b))}{\partial \mathbf{w}} \\ &= - \frac{\partial(y_i \mathbf{w}^T \mathbf{x}_i)}{\partial \mathbf{w}} \\ &= -y_i \mathbf{x}_i \end{aligned}$$

- If $1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0$:

$$g_{\mathbf{w}}(\mathbf{x}_i) = 0$$

Gradient Computation

■ So we have:

$$g_{\mathbf{w}}(\mathbf{x}_i) = \begin{cases} -y_i \mathbf{x}_i & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0 \\ 0 & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0 \end{cases}$$

■ Let $g_b(\mathbf{x}_i) = \frac{\partial \xi_i}{\partial b}$

$$g_b(\mathbf{x}_i) = \begin{cases} -y_i & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0 \\ 0 & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0 \end{cases}$$

Gradient Descent

- Gradient descent is a batch algorithm that uses all examples

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b) = \mathbf{w} + \frac{C}{n} \sum_{i=1}^n g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\nabla_b \mathcal{L}(\mathbf{w}, b) = \frac{C}{n} \sum_{i=1}^n g_b(\mathbf{x}_i)$$

Algorithm 1: GD

```
1 Initialize parameter  $\mathbf{w}$  and learning rate  $\eta$ 
2 while stopping condition is not achieved do
3      $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b)$ 
4      $b = b - \eta \mathcal{L}(\mathbf{w}, b)$ 
5 end
```

Gradient Descent

■ The algorithm for calculating the subgradient is as follow:

$$g_{\mathbf{w}}(\mathbf{x}_i) = \begin{cases} -y_i \mathbf{x}_i & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0 \\ 0 & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0 \end{cases}$$

$$g_b(\mathbf{x}_i) = \begin{cases} -y_i & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0 \\ 0 & 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0 \end{cases}$$

Algorithm 2: Subgradient

```
1 Initialize  $g_{\mathbf{w}} = 0, g_b = 0$ 
2 for  $i=1$  to  $n$  do
3   if  $y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq 1$  then
4      $g_{\mathbf{w}} = g_{\mathbf{w}} + (-y_i \mathbf{x}_i)$ 
5      $g_b = g_b + (-y_i)$ 
6   end if
7 end for
```

Contents

- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Stochastic Optimization Motivation

- Information is redundant amongst samples
- Sufficient samples mean we can afford noisy updates
- Never-ending stream means we should not wait for all data
- Tracking non-stationary data means the target is changing

Stochastic Optimization

Keypoint: enough iterations with gradient from a small, current subsamples of dataset help to converge to the global minimum

- Better for large datasets and often faster convergence
- Hard to reach high accuracy
- Classical methods cannot handle stochastic approximation

Stochastic Gradient Descent

- SGD works similar as GD, but more quickly by estimating gradient from an example at a time

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b) = \mathbf{w} + C g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\nabla_b \mathcal{L}(\mathbf{w}, b) = C g_b(\mathbf{x}_i)$$

Algorithm 3: SGD

```
1 Initialize parameter  $\mathbf{w}$  and learning rate  $\eta$ 
2 while stopping condition is not achieved do
3     Randomly select an example  $i$  in the
       training set
4      $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b)$ 
5      $b = b - \eta \nabla_b \mathcal{L}(\mathbf{w}, b)$ 
6 end
```

Benefits of SGD

- Gradient is easy to calculate (instantaneous)
- Less prone to local minima
- Small memory to use
- Get to a reasonable solution quickly
- Can be used for more complex models

Contents

- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Minibatch Stochastic Gradient Descent

- Like the single random sample, the full gradient is approximated via an unbiased noisy estimate
- Rather than use a single point, randomly choose a subset S_k , and optimization problem is as follows:

$$\min_{\mathbf{w}, b} \mathcal{L} := \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{|S_k|} \sum_{i \in S_k} \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

- The size of S_k , denoted by $|S_k|$, is much smaller than the original data size n , and $|S_k|$ can be $2^3, 2^4 \dots$

Minibatch Stochastic Gradient Descent

- MSGD works identically to SGD, except that we use more than one training example to make each estimate of the gradient

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b) = \mathbf{w} + \frac{C}{|S_k|} \sum_{i \in S_k} g_{\mathbf{w}}(\mathbf{x}_i)$$

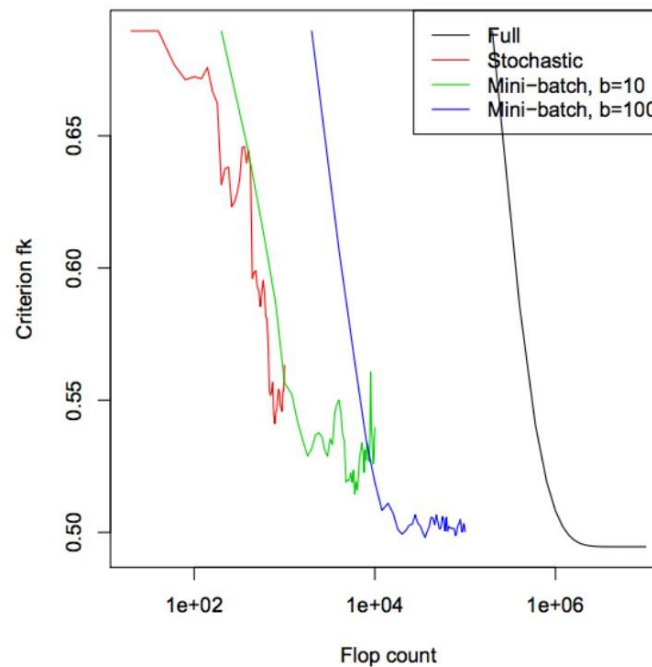
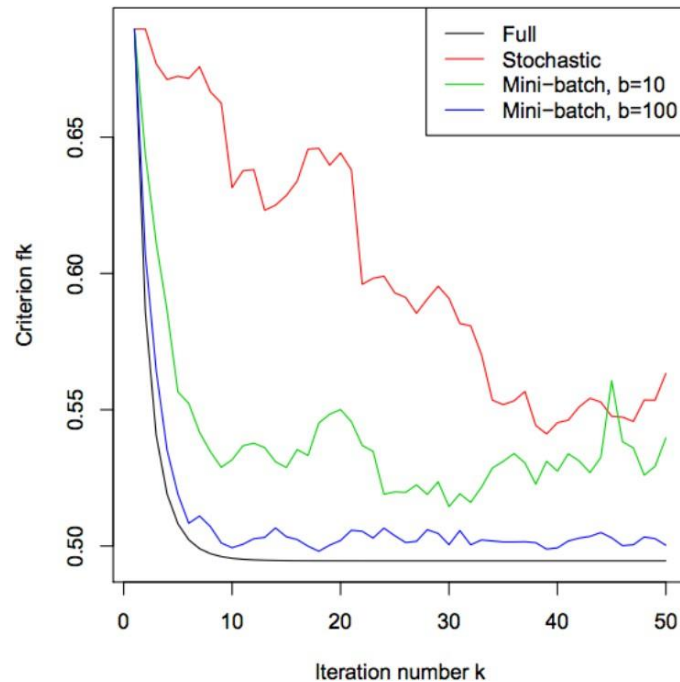
$$\nabla_b \mathcal{L}(\mathbf{w}, b) = \frac{C}{|S_k|} \sum_{i \in S_k} g_b(\mathbf{x}_i)$$

Algorithm 4: MSGD

```
1 Initialize parameter  $\mathbf{w}$  and learning rate  $\eta$ 
2 while stopping condition is not achieved do
3     Randomly select  $|S_k|$  examples in the
       training set
4      $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, b)$ 
5      $b = b - \eta \nabla_b \mathcal{L}(\mathbf{w}, b)$ 
6 end
```

Graphical Comparison of Optimizers

- Larger mini-batch size takes more computation time but less iterations



Importance of Learning Rate

- Learning rate has a large impact on convergence

Too small \rightarrow too slow

Too large \rightarrow oscillate and may even diverge

- Should learning rate be fixed or adaptive?

- Is convergence necessary?

Non-stationary: convergence may not be required

Stationary: learning rate should decrease with time

SGD Recommendations

Randomly shuffle training examples

- Although theory says you should randomly pick examples, it is easier to pass through your training set sequentially
- Shuffling before each iteration eliminates the effect of order

Monitor both training cost and validation errors

- Set aside samples for validation set
- Compute the objective on the training set and validation set
(expensive but better than overfitting or wasting computation)

SGD Recommendations

Check gradient using finite differences

- Incorrect computation can yield erratic and slow algorithm
- Verify your code by slightly perturbing the parameter and inspecting differences between the two gradients

Leverage sparsity of the training examples

- For very high-dimensional vectors with few non-zero coefficients, you only need to update the weight corresponding to non-zero pattern in \mathbf{x}

SGD Recommendations

Experiment with the learning rates using small sample of training set

- SGD convergence learning rates are independent from sample size
- Use traditional optimization algorithms as a reference point

Use learning rates of the form $\eta_t = \eta_{t-1}(1 + \eta_{t-1}\lambda t)^{-1}$

- Allow you to start from reasonable learning rates determined by testing on a small sample
- Work well in most situations if the initial point is slightly smaller than best value observed in training sample

Contents

- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Three Common Classification Problems

- Binary classification
- Multi-class classification
- Multi-label classification

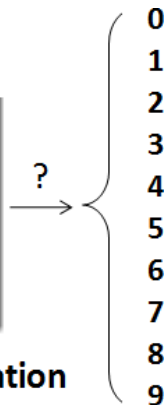
Is it the number 2?(Y/N)



Binary classification



Multi-class classification



Multi-label classification

Multi-class Classification

Multi-class classification is the common classification problem, which classifies instances into **one of the more than two classes**

Dataset

- MNIST
- Cifar-10 and Cifar-100
- ImageNet
- ...

Multi-class Classification

Three general strategies :

- Convert to binary classification
- Extend from binary classification
- Hierarchical classification

Convert to Binary Classification

The strategies reduce the problem of multi-class classification to multiple binary classification problems

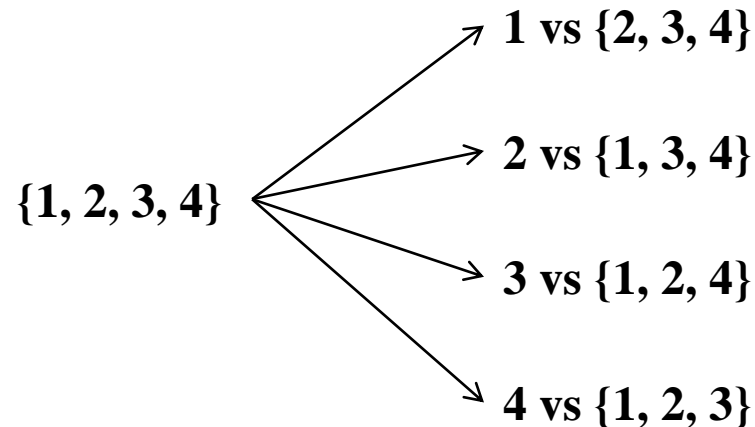
- One-vs.-rest
- One-vs.-one
- Decision Directed Acyclic Graph, DDAG

One-vs.-rest Method

Training:

- For each class

Train a binary classifier with the samples of that class as positive samples and others as negatives



One-vs.-rest

One-vs.-rest Method

Prediction:

- For each binary classifier
 - Produce a real-valued confidence score
- Predict the label with the highest confidence score

$$\hat{y} = \max_{k \in \{1 \dots K\}} f_k(x)$$

One-vs.-rest Method

Advantage vs Disadvantages

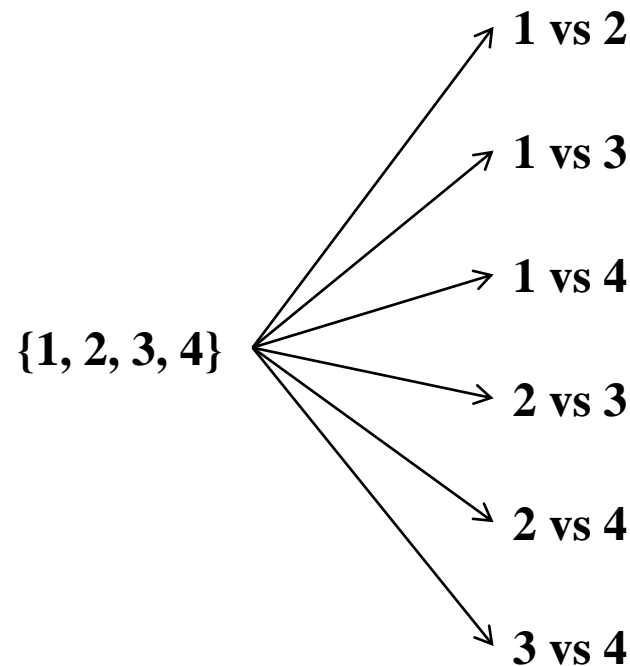
- Advantage
 - Train K binary classifiers for a K -way multi-class problem
- Disadvantages
 - The distributions of the binary classifications are unbalanced
(The set of negatives **is much larger than** the set of positives)
 - The scale of the confidence values may differ between the binary classifiers

One-vs.-one Method

Training:

- For each pair of classes

Train a binary classifier to discriminate between them



One-vs.-one

One-vs.-one Method

Prediction:

- For each binary classifier

- Contrast the two categories and do a voting

$$A = B = C = D = 0$$

A vs B -classifier: if A win, $A = A + 1$; otherwise, $B = B + 1$;

A vs C -classifier: if A win, $A = A + 1$; otherwise, $C = C + 1$;

...

C vs D -classifier: if C win, $C = C + 1$; otherwise, $D = D + 1$;

- Predict the label with the maximum number of votes wins

$$\hat{y} = \max(A, B, C, D)$$

One-vs.-one Method

Advantage vs Disadvantages

■ Advantage

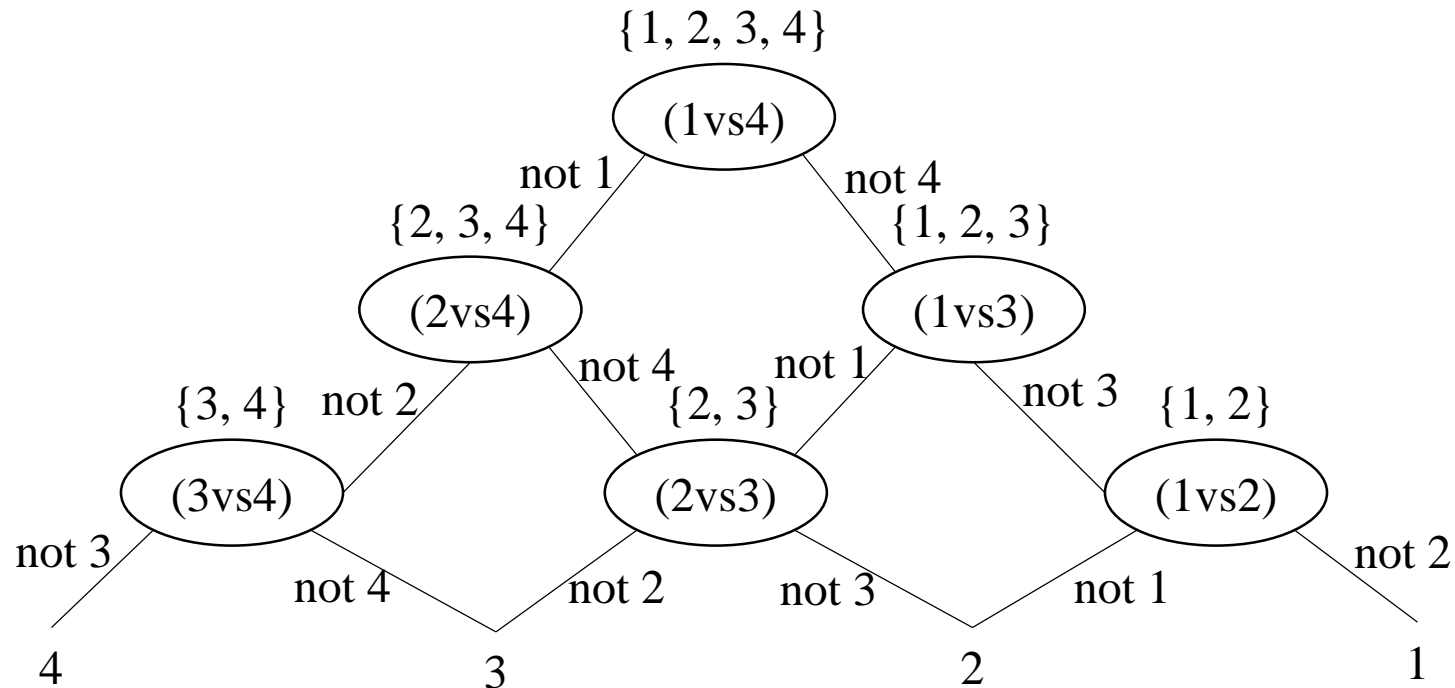
- The distributions of the binary classification are balanced

■ Disadvantages

- Train $K(K - 1)/2$ binary classifiers for a K -way multi-class problem, which has high computed complexity
- Suffer from ambiguities when receive the same number of votes

Decision Directed Acyclic Graph

- Compared to One-vs.-one method, it uses a **rooted binary directed acyclic graph** which has internal nodes and leaves



Decision Directed Acyclic Graph

Decision Directed Acyclic Graph

Predict:

- Start at the root node
- Before reaching a leaf node:
 - Evaluate the binary decision function
 - Move to either left or right depending on the output value

Compared to One-vs.-one method

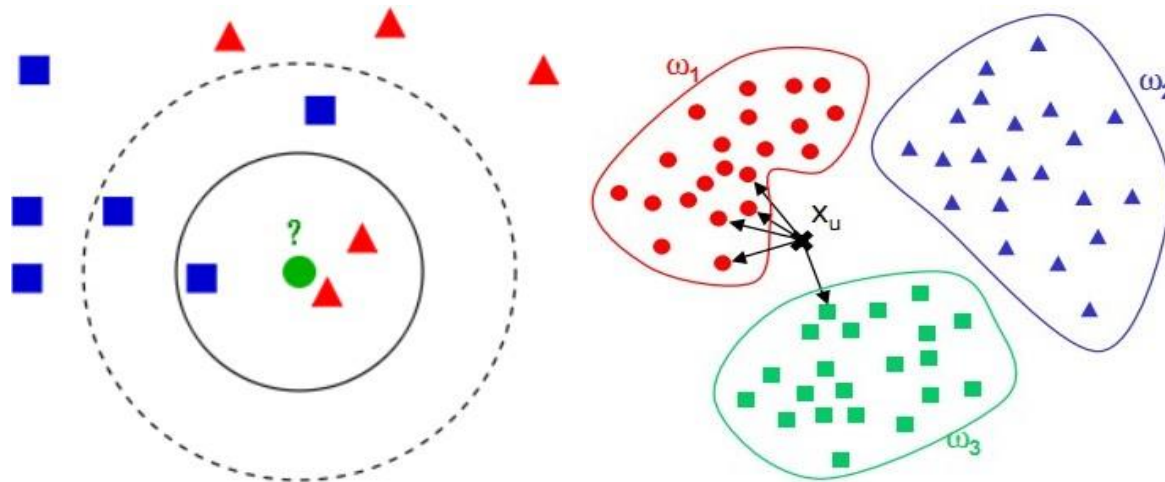
- The correlation between each of the binary classifications brings the cost of the prediction down

Extend from Binary Classification

- The strategy extends the existing binary classifiers to solve multi-class classification problems
 - K-nearest neighbors

K-nearest Neighbors

- Calculate the distances
 - Calculate the distances between the test object and each object in the training set
- Find the neighbors
 - Get the K-nearest training objects as neighbors
- Vote on labels
 - Classify the test object based on the most frequent class of the neighbors



K-nearest Neighbors

Advantages vs Disadvantages

■ Advantages

- The method is a non-parametric classification algorithm
- The algorithm can naturally handle binary and multi-class classification

■ Disadvantages

- The computational and memory requirements are high
- Finding good representations and distance measures between objects is hard

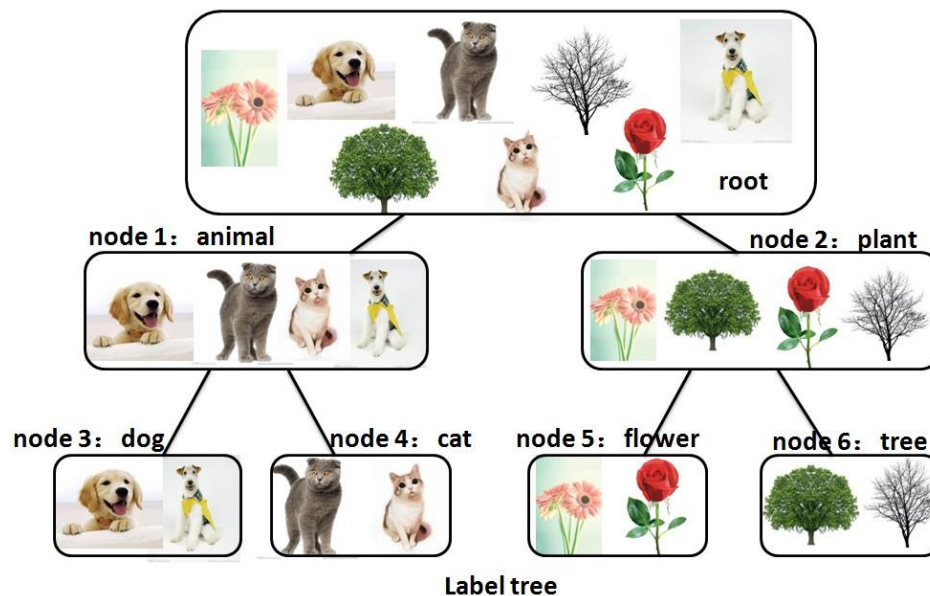
Hierarchical Classification

- The strategy tackles the multi-class classification problem by dividing the output space into a tree
 - Label tree

Label Tree

Train:

- Before the leaf nodes contain only a single class
 - Each parent node is divided into a number of clusters, one for each child node
- At each node, a simple classifier is trained to discriminate between the different child class clusters



Label Tree

Predict:

- Start from the root node
 - Travel to a leaf node which is associated with a label

Advantage vs Disadvantage

- Advantage
 - The tree method brings the cost of the prediction down
- Disadvantage
 - Depended on a good clustering method.

Q & A?

Contents

- 1 Linear Classification
- 2 Support Vector Machine
- 3 Gradient Descent
- 4 Stochastic Gradient Descent
- 5 Mini-Batch Stochastic Gradient Descent
- 6 Multi-class Classification
- 7 Dual Problem For SVM

Dual Problem for SVM

- An optimization problem can be considered in two ways, **primal problem** and **dual problem**

- For primal problem of basic SVM:

$$\min_{\mathbf{w}, b} \frac{||\mathbf{w}'||^2}{2}$$

$$s. t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, n$$

- Its Lagrange function is:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2} ||\mathbf{w}'||^2 + \sum_{i=1}^n \alpha_i \left(1 - y_i(\mathbf{w}^T \mathbf{x}_i + b) \right) \quad (1)$$

Dual Problem for SVM

- Its Lagrange “dual function” is:

$$D(\alpha) = \inf \mathcal{L}(w, b, \alpha)$$

Dual function gives the lower bound of the optimal value of primal problem

- Dual problem: the best lower bound dual function can get

$$\max_{\alpha} D(\alpha)$$

Dual Problem for SVM

- Setting partial derivative of D with respect to \mathbf{w} equal to 0:

$$\begin{aligned}\nabla_{\mathbf{w}} D &= \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0 \\ \rightarrow \mathbf{w} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i\end{aligned}\tag{2}$$

- Setting partial derivative of \mathcal{L} with respect to b equal to 0:

$$\nabla_b D = \sum_{i=1}^n \alpha_i y_i = 0\tag{3}$$

Dual Problem for SVM

- Put (2), (3) into (1), we can get:

$$\begin{aligned}\mathcal{L}(\mathbf{w}, b, \alpha) &= \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^n \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i + b)) \\&= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \mathbf{w}^T \alpha_i y_i \mathbf{x}_i - \sum_{i=1}^n \alpha_i y_i \\&= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^n \alpha_i - \mathbf{w}^T \mathbf{w} - 0 \\&= \sum_{i=1}^n \alpha_i - \frac{1}{2} \left[\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right]^T \left[\sum_{j=1}^n \alpha_j y_j \mathbf{x}_j \right] \\&= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j\end{aligned}$$

Dual Problem for SVM

- Last, we can get the dual problem:

$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$s.t. \quad \sum_{i=1}^n \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, n$$

Thank You