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

Prof.Mingkui Tan

South China University of Technology
Southern Artificial Intelligence Laboratory(SAIL)

December 1, 2018

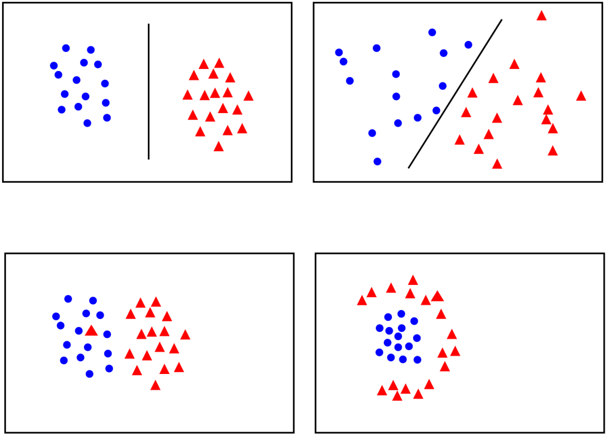
Content

Contents

Linear Separability

linearly separable

not linearly separable

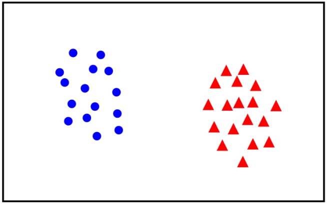


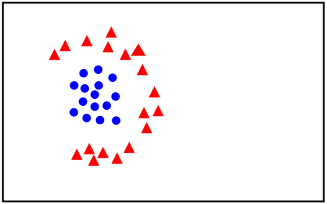
Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \dots n$, with $\mathbf{x}_i \in R^m$ and $y_i \in \{-1, 1\}$, learn a classfier $f(\mathbf{x})$ 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

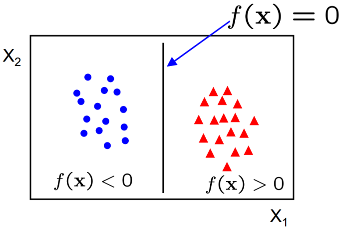




Linear Classifiers: 2D Example

A linear classifier has the form:

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

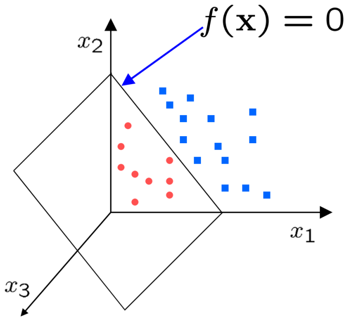


- In 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b is the bias
- $\mathbf{w}$ is known as the weight vector

Linear Classifiers: 3D Example

A linear classifier has the form:

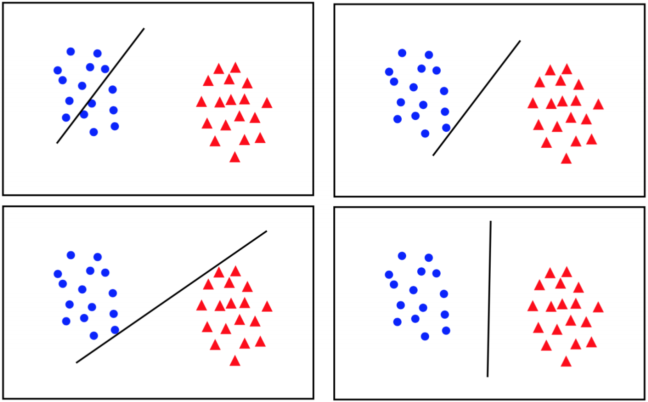
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- In $3D$ the discriminant is a plane, and in mD it is a hyperplane

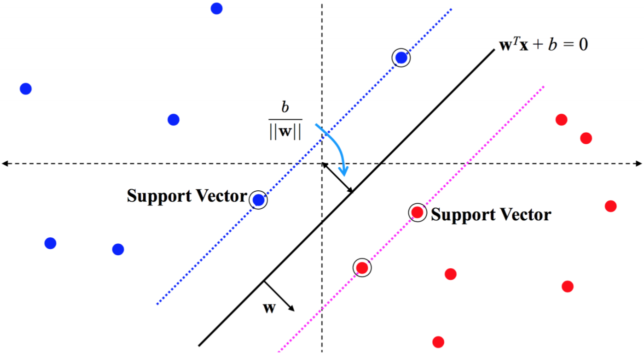
Contents

What's a Good Decision Boundary?



- Maximum margin solution: most stable under perturbations of the inputs

Max-margin Methods



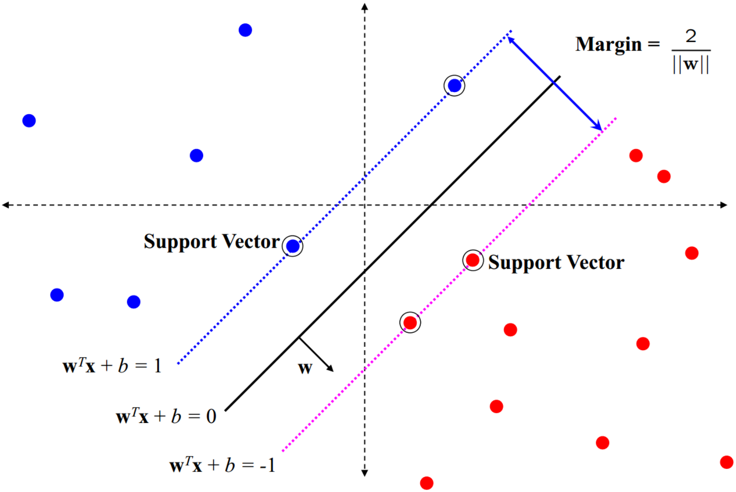
- Select two parallel hyperplanes that separate the two classes of data and let the distance between them as large as possible
- The region bounded by these two hyperplanes is called the "margin"

SVM-sketch Derivation

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

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

Support Vector Machine



Basic Support Vector Machine

- Learning the SVM can be formulated as an optimization:

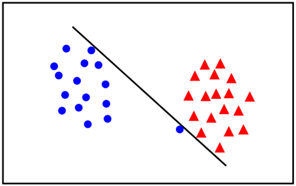
$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|}$$

$$s.t. \quad \mathbf{w}^\top \mathbf{x}_i + b \begin{cases} \geqslant 1 & y_i = +1 \\ \leqslant -1 & y_i = -1 \end{cases}$$

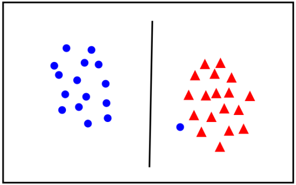
- Or equivalently:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geqslant 1, \qquad i = 1, 2, \ldots, n$$

Linear Separability Again: What is The Outlier?



• the points can be linearly separated but there is a very narrow margin

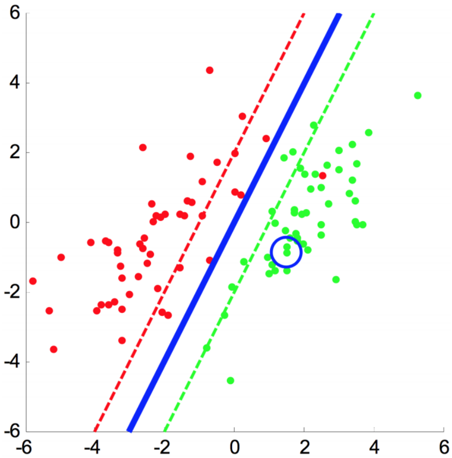


• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

Linear Separability Again: What is The Outlier?

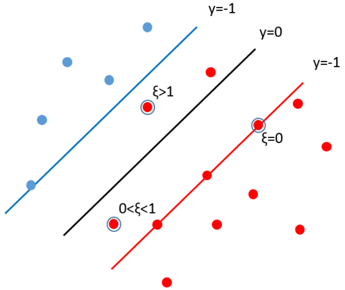
Moreover, training data may not be linearly separable!



A Relaxed Formulation

Introduce variable $\xi_i \geqslant 0$, for each i, which represents how much example i is on wrong side of margin boundary

- If $\xi_i = 0$ then it is ok
- If $0 < \xi_i < 1$ it is correctly classified, but with a smaller margin than $\frac{1}{\|\mathbf{w}\|}$
- If $\xi_i > 1$ then it is incorrectly classified



Soft Margin Formulation

The optimization problem becomes:

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

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

Note:

-small C allows constraints to be easily ignored

-large C makes constraints hard to ignore

Soft Margin Formulation

- As can be seen from above:

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

- What is the optimal value ξ_i as a function of $\mathbf{w}$ and b?

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

- So ξ_i can be written as:

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

Hinge Loss

Hinge loss:

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

The optimization problem becomes:

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

Contents

Gradient Descent

- Gradient descent minimum optimization problem:

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

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

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

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

- where η is the learning rate.

Gradient Descent

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

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

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

Algorithm 1: GD

Initialize parameter $\mathbf{w}$ and learning rate η
while stopping condition is not achieved **do**
  $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}, b)$
  $b = b - \eta \nabla_b L(\mathbf{w}, b)$
end

Gradient Computation

- Gradient descent minimum optimization problem:

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

- Gradient computation:

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

Gradient Computation

$$\mathbf{w} = \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix}^\top$$
$$\|\mathbf{w}\|^2 = \|\mathbf{w}\|_2^2 = w_1^2 + w_2^2 + \dots + w_n^2$$

- Writting in the denominator-layout notation:

$$\begin{aligned} \frac{\partial(\|\mathbf{w}\|^2)}{\partial \mathbf{w}} &= \begin{bmatrix} \frac{\partial(w_1^2+w_2^2+\dots+w_n^2)}{\partial w_1} & \dots & \frac{\partial(w_1^2+w_2^2+\dots+w_n^2)}{\partial w_n} \end{bmatrix}^\top \\ &= \begin{bmatrix} 2w_1 & \dots & 2w_n \end{bmatrix}^\top \\ &= 2\mathbf{w} \end{aligned}$$

- so we have:

$$\frac{1}{2} \cdot \frac{\partial(\|\mathbf{w}\|^2)}{\partial \mathbf{w}} = \mathbf{w}$$

Gradient Computation

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

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

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

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

Gradient Computation

- so we have:

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

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

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

The final algorithm

Gradient descent is a batch algorithm that uses all examples

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

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

Algorithm 2: GD

Initialize parameter $\mathbf{w}$ and learning rate η
while stopping condition is not achieved **do**
    $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}, b)$
    $b = b - \eta \nabla_{b} L(\mathbf{w}, b)$
end

The final algorithm

The algorithm for calculating the subgradient is as follows:

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

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

Algorithm 3: Subgradient

```
Initialize g_w=0,g_b=0
for i=1 to n do
    if  y_i(w^T x_i + b) ≤ 1 then
        g_w=g_w+(-y_i x_i)
        g_b=g_b+(-y_i)
    end if
end for
```

Contents

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 moving

Stochastic Optimization

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

- Better for large datasets and often faster convergence
- Hard to reach high accuracy
- Classical methods can not handle stochastic approximation
- Theoretical definitions for convergence are not as well defined

Stochastic Gradient Descent

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

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

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

Algorithm 4: SGD

Initialize parameter $\mathbf{w}$ and learning rate η

while stopping condition is not achieved **do**

  Randomly select an example i in the training set

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

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

end

The Benefits of SGD

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

Contents

Minibatch Stochastic Gradient Descent

- Like the single random sample, the full gradient is approximated via an unbiased noisy estimate
- Rather than using a single point, randomly choose a subset $\boldsymbol{\mathcal{S}}_k$, and optimization problem:

$$\min_{\mathbf{w},b} \quad L : \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{|\boldsymbol{\mathcal{S}}_k|} \sum_{i \in \boldsymbol{\mathcal{S}}_k} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

- The size of $\boldsymbol{\mathcal{S}}_k$, denoted by $|\boldsymbol{\mathcal{S}}_k|$, is much smaller than the original data size n, and $|\boldsymbol{\mathcal{S}}_k|$ can be 2^3, 2^4 ...

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

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

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

Algorithm 5: MSGD

```
Initialize parameter w and learning rate η
while stopping condition is not achieved do
    Randomly select |S_k| examples in the training set
    w = w − η∇_w L(w, b)
    b = b − η∇_b L(w, b)
end
```

Example

- n=10000,d=20

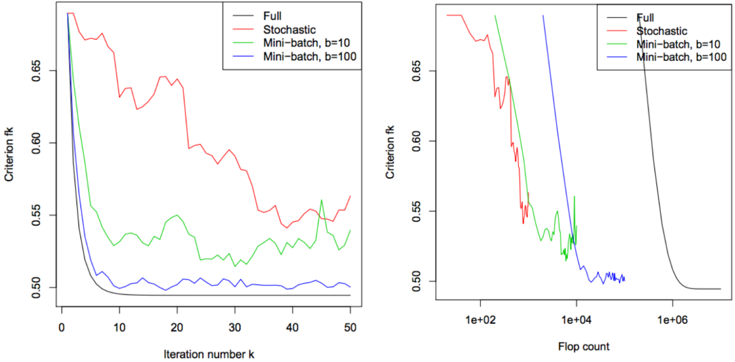


Figure: Larger mini-batch size take more cpmputation time but less iterations

Importance of Learning Rate

- Learning rate has a large impact on convergence
  - Too small $\rightarrow$ too slow
  - Too large $\rightarrow$ oscillatory 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 error

- 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 nonzero 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 $\eta_t = \eta_{t-1}(1 + \eta_{t-1}\lambda t)^{-1}$

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

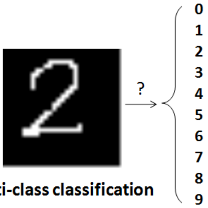
Contents

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

- Transformation to binary classification
- Extension from binary classification
- Hierarchical classification

Transformation to binary classification

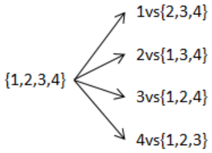
The strategies reduces the problem of multi-class classification to multiple binary classification problems.

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

One-vs.-rest method

Train:

- 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

Predict:

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

$$\hat{y} =_{k \epsilon \{1 \ldots k\}}^{argmax} f_k(x)$$

One-vs.-rest method

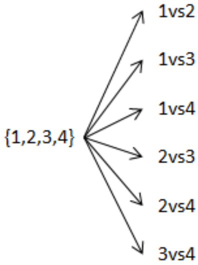
Advantage vs Disadvantage

- Advantage
  - Trains K binary classifiers for a K-way multi-class problem.
- Disadvantage
  - 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

Train:

- For each pair of classes
  - Train a binary classifier to discriminate between them.



One-vs.-one

One-vs.-one method

Predict:

- 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.

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

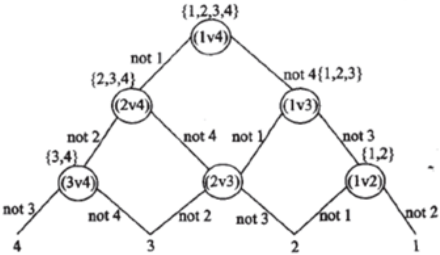
One-vs.-one method

Advantage vs Disadvantage

- Advantage
  - The distributions of the binary classifications are balanced.
- Disadvantage
  - 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.

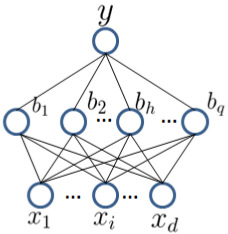
Extension from binary classification

The strategies extends the existing binary classifiers to solve multi-class classification problems.

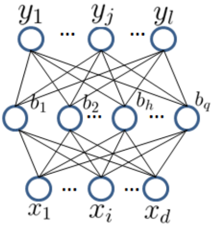
- Neural networks
- Decision trees
- K-nearest neighbours
- Softmax function

Neural networks

Instead of just having one neuron in the output layer, the network could have N binary neurons leading to multi-class classification.



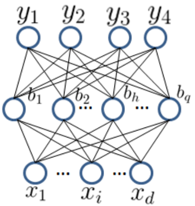
binary classification



multi-class classification

Neural networks

Each output neuron is designated to identify a given class. N=K



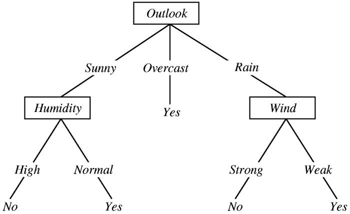
K \ N	y_1	y_2	y_3	y_4
Class 1	1	0	0	0
Class 2	0	1	0	0
Class 3	0	0	1	0
Class 4	0	0	0	1

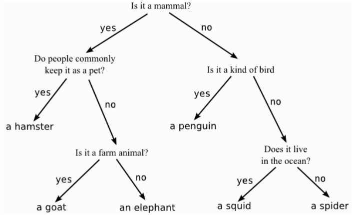
One-per-class coding

Train: The loss function $\mathbf{E} = \sum_{j=1}^{l} \mathbf{E}_j$
Predict: The neuron with the maximum output is considered as the class of the example.

Decision trees

The decision tree tries to split the training data based on the values of the features to produce a good generalization.

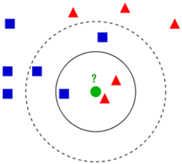


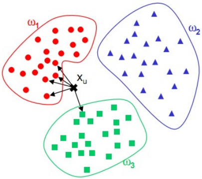


- The algorithm can naturally handle binary and multi-class classification.

K-nearest neighbours

- 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 neighbours

Advantage vs Disadvantage

- Advantage
  - The method is a non-parametric classification algorithms.
  - The algorithm can naturally handle binary and multi-class classification.
- Disadvantage
  - The computational and memory requirements are high.
  - Finding good representations and distance measures between objects is hard.

Hierarchical classification

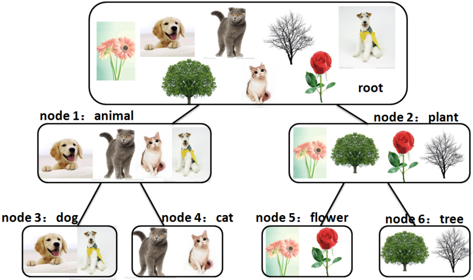
The strategies tackles the multi-class classification problem by dividing the output space i.e. into a tree.

- Label tree

Label tree

Train:

- Before the leaf nodes contain only a single class
  - Each parent node are 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

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
  - Finding good clustering method is important.

Q & A?

Contents

Dual Problem for SVM

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

- for primal problem of basic SVM:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geqslant 1, \qquad i = 1, 2, \ldots, n$$

- its Lagrange function is:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \tag{1}$$

Dual Problem for SVM

- its Lagrange "dual function" is:

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

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

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

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

Dual Problem for SVM

- setting partial derivative of $\mathbf{D}$ with respect to $\mathbf{w}$ to 0:

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

- setting partial derivative of $\mathcal{L}$ with respect to b to 0:

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

Dual Problem for SVM

- adding (2), (3) to (1):

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

Dual Problem for SVM

- finally, we can get the dual problem:

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i &- \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \\
\alpha_i \geqslant 0, \quad & i = 1, 2, \ldots, n
\end{aligned}
$$

THANK YOU!