Introduction to Logistic Regression and Support Vector Machine

guest lecturer: Ming-Wei Chang
CS 446

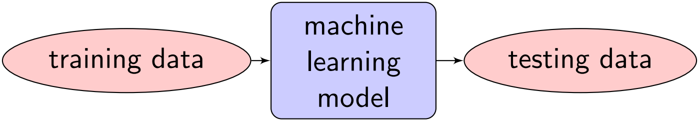
Fall, 2009

Before we start

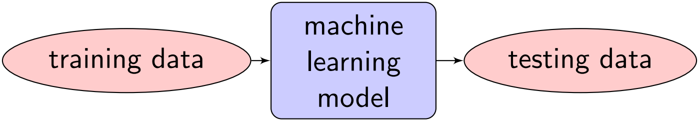
Before we start

- Feel free to ask questions anytime
- The slides are newly made. Please tell me if you find any mistake.

Today: supervised learning algorithms



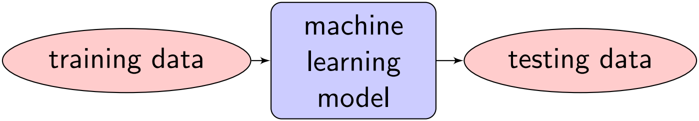
Today: supervised learning algorithms



Supervised learning algorithms we have mentioned

- Decision Tree
- Online Learning: Perceptron, Winnow, ...
- Generative Model: Naive Bayes

Today: supervised learning algorithms



Supervised learning algorithms we have mentioned

- Decision Tree
- Online Learning: Perceptron, Winnow, . . .
- Generative Model: Naive Bayes

What are we going to talk about today?

- "Modern" supervised learning algorithms
- Specifically, logistic regression and support vector machine

Motivation

- Logistic regression and support vector machine are both very popular!

Motivation

- Logistic regression and support vector machine are both very popular!

- Batch learning algorithms
  - Using optimization algorithms as training algorithms
  - An important technique we need to be familiar with.
  - Learn not to be afraid of these algorithms

Motivation

- Logistic regression and support vector machine are both very popular!

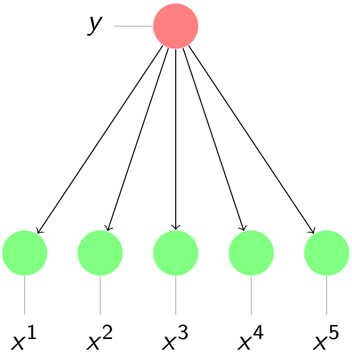
- Batch learning algorithms
  - Using optimization algorithms as training algorithms
  - An important technique we need to be familiar with.
  - Learn not to be afraid of these algorithms

Understand the relationships
between these algorithms and the algorithms we have learned

Review: Naive Bayes

Notations

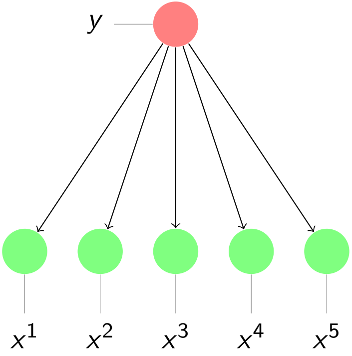
- **Input**: x, **Output** $y \in \{+1, -1\}$
- Assume each x has m features.
  - We use x^j to represent the j-th features of x



Review: Naive Bayes

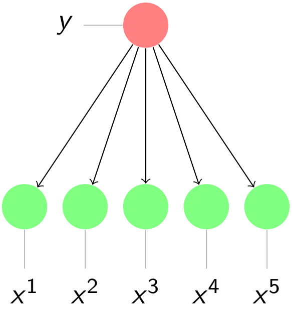
Notations

- **Input**: x, **Output** $y \in \{+1, -1\}$
- Assume each x has m features.
  - We use x^j to represent the j-th features of x



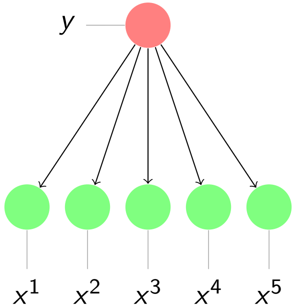
Conditional Independence

Review: Naive Bayes



$$P(y, x) = P(y) \prod_{j=1}^{m} P(x^j | y)$$

Review: Naive Bayes

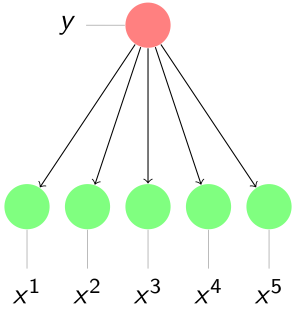


$$P(y, x) = P(y) \prod_{j=1}^{m} P(x^j | y)$$

Training

- Maximize the likelihood of $P(D) = P(Y, X) = \prod_i^l p(y_i, x_i)$
- Algorithm
  - Estimate $P(y = -1)$ and $P(y = 1)$ by counting
  - Estimate $P(x^j | y)$ by counting

Review: Naive Bayes



$$P(y, x) = P(y) \prod_{j=1}^{m} P(x^j | y)$$

Training

- Maximize the likelihood of $P(D) = P(Y, X) = \prod_i^l p(y_i, x_i)$
- Algorithm
  - Estimate $P(y = -1)$ and $P(y = 1)$ by counting
  - Estimate $P(x^j | y)$ by counting

Testing

- $\frac{P(y=+1|x)}{P(y=-1|x)} = \frac{P(y=+1,x)}{P(y=-1,x)} \geq 1?$

Review: Naive Bayes

The prediction function of a Naive Bayes model is a linear function

- In previous lectures, we have shown that
  $$log \frac{P(y=+1|x)}{P(y=-1|x)} \geq 0 \Rightarrow w^T x + b \geq 0$$
- The counting results can be re-expressed as a linear function

Review: Naive Bayes

The prediction function of a Naive Bayes model is a linear function

- In previous lectures, we have shown that

$$log \frac{P(y=+1|x)}{P(y=-1|x)} \geq 0 \Rightarrow w^T x + b \geq 0$$

- The counting results can be re-expressed as a linear function

- Key observation: Naive Bayes cannot express all possible linear functions
  - Intuition: conditional independence assumption

Review: Naive Bayes

The prediction function of a Naive Bayes model is a linear function

- In previous lectures, we have shown that

$$log \frac{P(y=+1|x)}{P(y=-1|x)} \geq 0 \Rightarrow w^T x + b \geq 0$$

- The counting results can be re-expressed as a linear function

- Key observation: Naive Bayes cannot express all possible linear functions
  - Intuition: conditional independence assumption
- We will propose a model (logistic regression) that can express all possible linear functions in the next few slides.

Modeling conditional probability using a linear function

Starting point: the predicting function of Naive Bayes

$$log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x + b \Leftrightarrow$$

Modeling conditional probability using a linear function

Starting point: the predicting function of Naive Bayes

$$log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x + b \Leftrightarrow$$

$$\frac{P(y=+1|x)}{1-P(y=+1|x)} = \frac{e^{w^T x+b}}{1} \Leftrightarrow$$

Modeling conditional probability using a linear function

Starting point: the predicting function of Naive Bayes

$$log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x + b \Leftrightarrow$$

$$\frac{P(y=+1|x)}{1-P(y=+1|x)} = \frac{e^{w^T x+b}}{1} \Leftrightarrow$$

$$P(y=+1|x) = \frac{e^{w^T x+b}}{1+e^{w^T x+b}} = \frac{1}{1+e^{-1(w^T x+b)}}$$

Modeling conditional probability using a linear function

Starting point: the predicting function of Naive Bayes

$$log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x + b \Leftrightarrow$$

$$\frac{P(y=+1|x)}{1-P(y=+1|x)} = \frac{e^{w^T x+b}}{1} \Leftrightarrow$$

$$P(y=+1|x) = \frac{e^{w^T x+b}}{1+e^{w^T x+b}} = \frac{1}{1+e^{-1(w^T x+b)}}$$

The conditional probability $P(y|x) = \frac{1}{1+e^{-y(w^T x+b)}}$

Modeling conditional probability using a linear function

Starting point: the predicting function of Naive Bayes

$$log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x + b \Leftrightarrow$$

$$\frac{P(y=+1|x)}{1-P(y=+1|x)} = \frac{e^{w^T x+b}}{1} \Leftrightarrow$$

$$P(y=+1|x) = \frac{e^{w^T x+b}}{1+e^{w^T x+b}} = \frac{1}{1+e^{-1(w^T x+b)}}$$

The conditional probability $P(y|x) = \frac{1}{1+e^{-y(w^T x+b)}}$

- In order to simplify the notation,
  - $w^T \leftarrow \begin{bmatrix} w^T & b \end{bmatrix}$
  - $x^T \leftarrow \begin{bmatrix} x^T & 1 \end{bmatrix}$

Modeling conditional probability using a linear function

Starting point: the predicting function of Naive Bayes

$$log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x + b \Leftrightarrow$$

$$\frac{P(y=+1|x)}{1-P(y=+1|x)} = \frac{e^{w^T x+b}}{1} \Leftrightarrow$$

$$P(y=+1|x) = \frac{e^{w^T x+b}}{1+e^{w^T x+b}} = \frac{1}{1+e^{-1(w^T x+b)}}$$

The conditional probability $P(y|x) = \frac{1}{1+e^{-y(w^T x+b)}}$

- In order to simplify the notation,
  - $w^T \leftarrow \begin{bmatrix} w^T & b \end{bmatrix}$
  - $x^T \leftarrow \begin{bmatrix} x^T & 1 \end{bmatrix}$

Using the bias trick, $P(y|x) = \frac{1}{1+e^{-y(w^T x)}}$

Logistic regression: introduction

Logistic regression: introduction

- Naive Bayes: model $P(y, x)$
  - conditional independence assumption: training = counting
  - not all w are possible

Logistic regression: introduction

- Naive Bayes: model $P(y, x)$
  - conditional independence assumption: training = counting
  - not all w are possible
- In the testing phase, we just showed that the conditional probability can be expressed

$$P(y|x, w) = \frac{1}{1 + e^{-y(w^T x)}} \tag{1}$$

Logistic regression: introduction

- Naive Bayes: model $P(y,x)$
  - conditional independence assumption: training = counting
  - not all w are possible
- In the testing phase, we just showed that the conditional probability can be expressed

$$P(y|x,w) = \frac{1}{1+e^{-y(w^T x)}} \tag{1}$$

Logistic Regression

- Maximizes conditional likelihood $P(y|x)$ directly in the training phase

Logistic regression: introduction

- Naive Bayes: model $P(y, x)$
  - conditional independence assumption: training = counting
  - not all w are possible
- In the testing phase, we just showed that the conditional probability can be expressed

$$P(y|x, w) = \frac{1}{1 + e^{-y(w^T x)}} \tag{1}$$

Logistic Regression

- Maximizes conditional likelihood $P(y|x)$ directly in the training phase
- How to find w?
  - $w = \text{argmax}_w P(Y|X, w) = \text{argmax}_w \prod_{i=1}^{l} P(y_i|x_i, w)$

Logistic regression: introduction

- Naive Bayes: model $P(y, x)$
  - conditional independence assumption: training = counting
  - not all w are possible
- In the testing phase, we just showed that the conditional probability can be expressed

$$P(y|x, w) = \frac{1}{1 + e^{-y(w^T x)}} \tag{1}$$

Logistic Regression

- Maximizes conditional likelihood $P(y|x)$ directly in the training phase
- How to find w?
  - $w = \text{argmax}_w P(Y|X, w) = \text{argmax}_w \prod_{i=1}^{l} P(y_i|x_i, w)$
  - For *all possible* w, find the one that maximizes the conditional likelihood
    - drop the conditional independence assumption!

Logistic regression: the final objective function

- $w = \text{argmax}_w P(Y|X, w) = \text{argmax}_w \prod_{i=1}^{l} P(y|x, w)$

Logistic regression: the final objective function

- $w = \text{argmax}_w P(Y|X, w) = \text{argmax}_w \prod_{i=1}^{l} P(y|x, w)$

Finding w as an optimization problem

$$\begin{aligned} w &= \underset{w}{\text{argmax}} \log P(Y|X, w) = \underset{w}{\text{argmin}} - \log P(Y|X, w) \\ &= \underset{w}{\text{argmin}} - \sum_{i=1}^{l} \log \frac{1}{1 + e^{-y_i(w^T x_i)}} \\ &= \underset{w}{\text{argmin}} \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)}) \end{aligned}$$

Logistic regression: the final objective function

- $w = \text{argmax}_w P(Y|X, w) = \text{argmax}_w \prod_{i=1}^{l} P(y|x, w)$

Finding w as an optimization problem

$$
\begin{aligned}
w = \underset{w}{\text{argmax}} \log P(Y|X, w) &= \underset{w}{\text{argmin}} - \log P(Y|X, w) \\
&= \underset{w}{\text{argmin}} - \sum_{i=1}^{l} \log \frac{1}{1 + e^{-y_i(w^T x_i)}} \\
&= \underset{w}{\text{argmin}} \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})
\end{aligned}
$$

Properties of this optimization problem

- A convex optimization problem

Adding regularization

Explanation

- Empirical loss : $\log(1 + e^{-y_i(w^T x_i)})$

Adding regularization

Explanation

- Empirical loss : $\log(1 + e^{-y_i(w^T x_i)})$
  - $y_i(w^T x_i)$ increases $\rightarrow \log(1 + e^{-y_i(w^T x_i)})$ decreases
  - In order to minimize the empirical loss, w will tend to be large
- Therefore, to prevent over-fitting, we add a regularization term

- Regularization Term

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})$$

Adding regularization

Explanation

- Empirical loss : $\log(1 + e^{-y_i(w^T x_i)})$
  - $y_i(w^T x_i)$ increases $\rightarrow \log(1 + e^{-y_i(w^T x_i)})$ decreases
  - In order to minimize the empirical loss, w will tend to be large
- Therefore, to prevent over-fitting, we add a regularization term

- Regularization Term

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})$$

- Empirical Loss

Adding regularization

Explanation

- Empirical loss : $\log(1 + e^{-y_i(w^T x_i)})$
  - $y_i(w^T x_i)$ increases $\rightarrow$ $\log(1 + e^{-y_i(w^T x_i)})$ decreases
  - In order to minimize the empirical loss, w will tend to be large
- Therefore, to prevent over-fitting, we add a regularization term

- Regularization Term

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})$$

- Empirical Loss
- balance parameter

Optimization

- An unconstrained problem. We can use the gradient descent algorithm!

Optimization

- An unconstrained problem. We can use the gradient descent algorithm! However, it is quite slow.
- Many other methods

Optimization

- An unconstrained problem. We can use the gradient descent algorithm! However, it is quite slow.
- Many other methods
  Iterative scaling; non-linear conjugate gradient; quasi-Newton methods; truncated Newton methods; trust-region newton method.
- All methods are iterative methods, that generate a sequence w_k
  Converging to the optimal solution of the optimization problem above.

Optimization

- An unconstrained problem. We can use the gradient descent algorithm! However, it is quite slow.
- Many other methods
  Iterative scaling; non-linear conjugate gradient; quasi-Newton methods; truncated Newton methods; trust-region newton method.
- All methods are iterative methods, that generate a sequence w_k
  Converging to the optimal solution of the optimization problem above.

Choice of optimization techniques

Low cost per iteration	–	High cost per iteration
(slow convergence)		(fast convergence)
Iterative scaling		Newton Methods
(each w component at a time)		

Optimization

- An unconstrained problem. We can use the gradient descent algorithm! However, it is quite slow.
- Many other methods
  Iterative scaling; non-linear conjugate gradient; quasi-Newton methods; truncated Newton methods; trust-region newton method.
- All methods are iterative methods, that generate a sequence w_k
  Converging to the optimal solution of the optimization problem above.

Choice of optimization techniques

Low cost per iteration	–	High cost per iteration
(slow convergence)		(fast convergence)
Iterative scaling		Newton Methods
(each w component at a time)		

Currently: Limited memory BFGS is very popular in NLP community

Logistic regression versus Naive Bayes

	Logistic regression	Naive Bayes
Training	maximize $P(Y\|X)$	maximize $P(Y,X)$
Training Algorithm	optimization algorithms	counting
Testing	$P(y\|x) \geq 0.5$?	$P(y\|x) \geq 0.5$?

Table: Comparison between Naive Bayes and logistic regression

Logistic regression versus Naive Bayes

	Logistic regression	Naive Bayes
Training	maximize $P(Y\|X)$	maximize $P(Y, X)$
Training Algorithm	optimization algorithms	counting
Testing	$P(y\|x) \geq 0.5$?	$P(y\|x) \geq 0.5$?

Table: Comparison between Naive Bayes and logistic regression

- LR and NB are both linear functions in the testing phase
- However, their training agendas are very different

Support Vector Machine: another loss function

- No magic. Just another loss function

Support Vector Machine: another loss function

- No magic. Just another loss function
- Logistic Regression

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})$$

Support Vector Machine: another loss function

- No magic. Just another loss function
- Logistic Regression

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})$$

- L1-loss SVM

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$$

Support Vector Machine: another loss function

- No magic. Just another loss function
- Logistic Regression

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i(w^T x_i)})$$

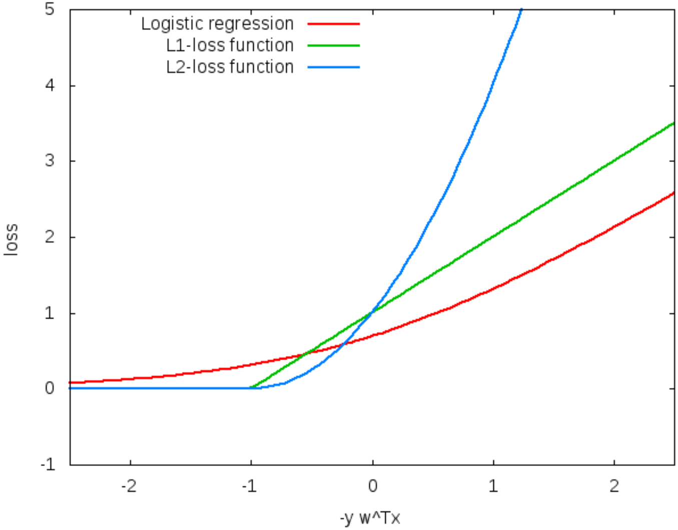
- L1-loss SVM

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$$

- L2-loss SVM

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)^2$$

Compare these loss functions



The regularization term: maximize margin

- The L1-loss SVM: $\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$

The regularization term: maximize margin

- The L1-loss SVM: $\min_w \frac{1}{2}w^T w + C\sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$
- Rewrite it using slack variables (why are they the same?)

$$\begin{aligned} \min_w \quad & \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \\ s.t. \quad & 1 - y_i w^T x_i \leq \xi_i, \xi_i \geq 0 \end{aligned}$$

The regularization term: maximize margin

- The L1-loss SVM: $\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$
- Rewrite it using slack variables (why are they the same?)

$$
\begin{aligned}
\min_w \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\
s.t. \quad & 1 - y_i w^T x_i \leq \xi_i, \xi_i \geq 0
\end{aligned}
$$

- If there is no training error, what is the margin of w?

The regularization term: maximize margin

- The L1-loss SVM: $\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$
- Rewrite it using slack variables (why are they the same?)

$$\begin{aligned} \min_w \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ s.t. \quad & 1 - y_i w^T x_i \leq \xi_i, \xi_i \geq 0 \end{aligned}$$

- If there is no training error, what is the margin of w? $\frac{1}{\|w\|}$

The regularization term: maximize margin

- The L1-loss SVM: $\min_w \frac{1}{2}w^T w + C\sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$
- Rewrite it using slack variables (why are they the same?)

$$\min_w \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$$
$$s.t. \quad 1 - y_i w^T x_i \leq \xi_i, \xi_i \geq 0$$

- If there is no training error, what is the margin of w? $\frac{1}{\|w\|}$
- Maximizing $\frac{1}{\|w\|} \Leftrightarrow$ minimizing $w^T w$

SVM regularization: find the linear line that maximizes the margin

The regularization term: maximize margin

- The L1-loss SVM: $\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)$
- Rewrite it using slack variables (why are they the same?)

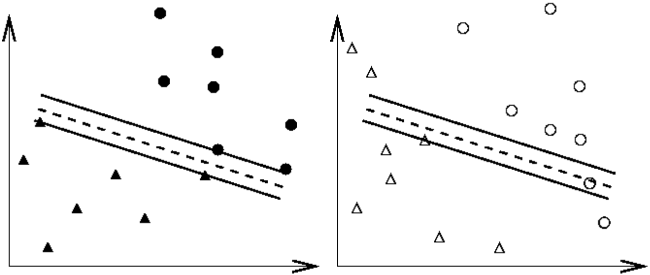
$$\begin{aligned} \min_w \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ s.t. \quad & 1 - y_i w^T x_i \leq \xi_i, \xi_i \geq 0 \end{aligned}$$

- If there is no training error, what is the margin of w? $\frac{1}{\|w\|}$
- Maximizing $\frac{1}{\|w\|} \Leftrightarrow$ minimizing $w^T w$

SVM regularization: find the linear line that maximizes the margin

Learning theory: Link to SVM theory notes

Balance between regularization and empirical loss



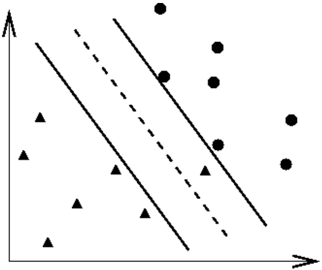
(a) Training data and an over-fitting classifier

(b) Testing data and an over-fitting classifier

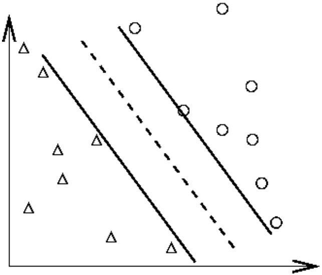
The maximal margin line with 0 training error

Best?

Balance between regularization and empirical loss



(c) Training data and a better classifier



(d) Testing data and a better classifier

If we allow some training error, we can find a better line

We need to balance the regularization term and the empirically loss term

Problem of model selection. Select balance parameter with cross validation

Primal and Dual Formulations

Explaining the primal-dual relationship

- Link to the lecture notes: 07-LecSvm-opt.pdf

Why primal-dual relationship is useful

- Link to a talk by Professor Chih-Jen Lin in 2005.
  - Optimization, Support Vector Machines, and Machine Learning. Talk in DIS, University of Rome and IASI, CNR, Italy. September 1-2, 2005.
- We will only use the slides from page 11-20.
- Link to notes

Nonlinear SVM

- SVM tries to find a linear line that maximizes the margin.
- Why people mention about non-linear SVM?

Nonlinear SVM

- SVM tries to find a linear line that maximizes the margin.
- Why people mention about non-linear SVM?
  - Usually this means: $x \rightarrow \phi(x)$
    - Find a linear function of $\phi(x)$
    - This can be a non linear function for x

Nonlinear SVM

- SVM tries to find a linear line that maximizes the margin.
- Why people mention about non-linear SVM?
  - Usually this means: $x \rightarrow \phi(x)$
    - Find a linear function of $\phi(x)$
    - This can be a non linear function for x

Primal

$$\min_{w,\xi_i} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$$

$$s.t. \quad 1 - y_i w^T \phi(x)_i \leq \xi_i$$

$$\xi_i \geq 0, \forall i = 1 \dots l$$

Nonlinear SVM

- SVM tries to find a linear line that maximizes the margin.
- Why people mention about non-linear SVM?
  - Usually this means: $x \rightarrow \phi(x)$
    - ★ Find a linear function of $\phi(x)$
    - ★ This can be a non linear function for x

Primal

$$\min_{w,\xi_i} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$$

$$s.t. \quad 1 - y_i w^T \phi(x)_i \leq \xi_i$$

$$\xi_i \geq 0, \forall i = 1 \ldots l$$

Dual

$$\min_{\alpha} \quad \frac{1}{2}\alpha^T Q\alpha - eT\alpha$$

$$s.t. \quad \forall i, 0 \leq \alpha_i \leq C$$

where Q is a l-by-l matrix with $Q_{ij} = y_i y_j K(x_i, x_j)$

Nonlinear SVM

- SVM tries to find a linear line that maximizes the margin.
- Why people mention about non-linear SVM?
  - Usually this means: $x \rightarrow \phi(x)$
    - Find a linear function of $\phi(x)$
    - This can be a non linear function for x

Primal

$$\begin{aligned} \min_{w,\xi_i} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ s.t. \quad & 1 - y_i w^T \phi(x)_i \leq \xi_i \\ & \xi_i \geq 0, \forall i = 1 \dots l \end{aligned}$$

Dual

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q \alpha - eT\alpha \\ s.t. \quad & \forall i, 0 \leq \alpha_i \leq C \end{aligned}$$

where Q is a l-by-l matrix
with $Q_{ij} = y_i y_j K(x_i, x_j)$

Same for Kernel perceptron: find a linear function on $\phi(x)$

Demo

Solving SVM

- Both primal and dual problems have constraints
  - We can not use the gradient descent algorithm
- For linear dual SVM, there is a simple optimization algorithm
  - Coordinate descent method!

Solving SVM

- Both primal and dual problems have constraints
  - We can not use the gradient descent algorithm
- For linear dual SVM, there is a simple optimization algorithm
  - Coordinate descent method!

$$\min_\alpha \quad \frac{1}{2}\alpha^T Q\alpha - eT\alpha$$
$$s.t. \quad \forall i, 0 \leq \alpha_i \leq C$$

  - # of α_i = # of training example
  - The idea: pick one example i. Optimize α_i only

Coordinate Descent Algorithm

Algorithm

- Run through the training data multiple times

Coordinate Descent Algorithm

Algorithm

- Run through the training data multiple times
  - Pick a random example (i) among the training data.

Coordinate Descent Algorithm

Algorithm

- Run through the training data multiple times
  - Pick a random example (i) among the training data.
  - Fix $\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_l$, only change α_i

$$\alpha_i' = \alpha_i + s$$

Coordinate Descent Algorithm

Algorithm

- Run through the training data multiple times
  - Pick a random example (i) among the training data.
  - Fix $\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_l$, only change α_i

$$\alpha_i' = \alpha_i + s$$

  - Solve the problem

$$\begin{aligned} \min_{s} \quad & \frac{1}{2}(\alpha + sd)^T Q(\alpha + sd) - eT(\alpha + sd) \\ s.t. \quad & 0 \leq \alpha_i + s \leq C \Leftarrow \text{only one constraint}, \end{aligned}$$

where d is a vector of $l - 1$ zeros. The i-th component of d is 1.

Coordinate Descent Algorithm

Algorithm

- Run through the training data multiple times
  - Pick a random example (i) among the training data.
  - Fix $\alpha_1, \alpha_2, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_l$, only change α_i

$$\alpha_i' = \alpha_i + s$$

  - Solve the problem

$$\begin{aligned} \min_s \quad & \frac{1}{2}(\alpha + sd)^T Q(\alpha + sd) - eT(\alpha + sd) \\ s.t. \quad & 0 \le \alpha_i + s \le C \Leftarrow \text{only one constraint}, \end{aligned}$$

    where d is a vector of $l - 1$ zeros. The i-th component of d is 1.
  - It is a single variable problem. We know how to solve this.

Coordinate Descent Algorithm

- Assume that the optimal s is s^*. We can update α_i using:

$$\alpha_i' = \alpha_i + s^*$$

- Given that $w = \sum_i^l \alpha_i y_i x_i$, this is equivalent to is equivalent to

$$w \leftarrow w + (\alpha_i' - \alpha_i) y_i x_i$$

- Isn't this familiar?

Coordinate Descent Algorithm

- Assume that the optimal s is s^*. We can update α_i using:

$$\alpha_i' = \alpha_i + s^* \Leftarrow \text{Similar to dual perceptron}$$

- Given that $w = \sum_i^l \alpha_i y_i x_i$, this is equivalent to is equivalent to

$$w \leftarrow w + (\alpha_i' - \alpha_i) y_i x_i$$

- Isn't this familiar?

Coordinate Descent Algorithm

- Assume that the optimal s is s^*. We can update α_i using:

$$\alpha_i' = \alpha_i + s^* \Leftarrow \text{Similar to dual perceptron}$$

- Given that $w = \sum_i^l \alpha_i y_i x_i$, this is equivalent to is equivalent to

$$w \leftarrow w + (\alpha_i' - \alpha_i) y_i x_i \Leftarrow \text{Similar to primal perceptron}$$

- Isn't this familiar?

Relationships between linear classifiers

- NB, LR, Perceptron and SVM are all linear classifiers
- NB and LR have the same interpretation for conditional probability

$$P(y|x, w) = \frac{1}{1 + e^{-y(w^T x)}} \quad (2)$$

- The difference between LR and SVM are their loss functions
  - But they are quite similar!
- Perceptron algorithm and the coordinate descent algorithm for SVM are very similar

Summary

Logistic regression

- Maximizes $P(Y|X)$ while Naive Bayes maximizes the joint probability $P(Y, X)$
- Model the conditional probability using a linear line. Drop the conditional independence assumption
- Many available methods of optimizing the objective function

Summary

Logistic regression

- Maximizes $P(Y|X)$ while Naive Bayes maximizes the joint probability $P(Y, X)$
- Model the conditional probability using a linear line. Drop the conditional independence assumption
- Many available methods of optimizing the objective function

Support Vector Machine

- Similar to Logistic Regression; Different Loss function
- Maximizes Margin; Has many nice theoretical properties
- Interesting Primal-Dual relationship
  - Allows us to choose the easier one to solve
- Many available methods of optimizing the objective function
  - The linear dual coordinate descent method turns out to be similar to Perceptron