WestlakeNLP

Natural Language Processing

Yue Zhang
Westlake University

西湖大學
WESTLAKE UNIVERSITY

WestlakeNLP

Chapter 4

Discriminative Linear Classifiers

Contents

Contents

Review linear models

- Input $x = w_1, w_2, \ldots, w_n$ output class c

Feature vector	Naïve Bayes Parameter Vector	SVM Parameter Vector	Perceptron Parameter Vector
$\phi(x,c) =$ $<c_1, w_1c_1, w_2c_2, \ldots, w_{\lvert V\rvert}c_1,$ $c_2, w_1c_2, w_2c_2, \ldots, w_{\lvert V\rvert}c_2,$ $\ldots$ $c_{\lvert C\rvert}, w_1c_{\lvert C\rvert}, w_2c_{\lvert C\rvert}, \ldots, w_{\lvert V\rvert}c_{\lvert C\rvert}>$	$\theta=$ $<log\,P(c_1), log\,P(w_1\lvert c_1), log\,P(w_2\lvert c_1), \ldots, log\,P(w_{\lvert V\rvert}\lvert c_1),$ $log\,P(c_2), log\,P(w_1\lvert c_2), log\,P(w_2\lvert c_2), \ldots, log\,P(w_{\lvert V\rvert}\lvert c_2),$ $\cdots$ $log\,P(c_{\lvert C\rvert}), log\,P(w_1\lvert c_{\lvert C\rvert}), log\,P(w_2\lvert c_{\lvert C\rvert}), \ldots, log\,P(w_{\lvert V\rvert}\lvert c_{\lvert C\rvert}) >$ Each instance trained separately using MLE.	Trained together with a large-margin objective	Trained together online.

Discriminative linear classifiers

WestlakeNLP

- The general form:

$$score(x, y) = \vec{\theta} \cdot \vec{\phi}(x, y)$$

- Instances :
  - SVMs
  - Perceptrons

- Advantages over generative models (e.g. Naïve Bayes):
  - flexibility in feature definition
  - a direct training goal of minimizing prediction errors.

- Disadvantage:
  - No probabilistic interpretation

Log-linear Models

WestlakeNLP

- Probabilistic discriminative linear model
- Motivation: the Naive Bayes classifier:

$$P(c|d) \propto \prod_{i=1}^{n} P(w_i|c)P(c)$$

The log form of $P(c \mid d)$ is a linear model:

$$logP(c|d) \propto \sum_{i=1}^{n} logP(w_i|c) + logP(c)$$

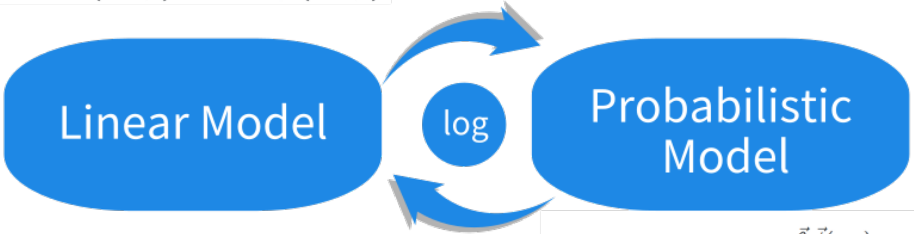
which is similar to a discriminative linear model.

- Design: $P(y|x) \propto exp\left(\vec{\theta} \cdot \vec{\phi}(x, y)\right)$

Log-linear model for multi-class classification

- Inputs : $x \in x$
- Outputs : $y \in C$

$$score(x, y) = \vec{\theta} \cdot \vec{\phi}(x, y)$$



$$P(y|x) = \frac{e^{\vec{\theta} \cdot \vec{\phi}(x,y)}}{\sum_{y' \in C} e^{\vec{\theta} \cdot \vec{\phi}(x,y')}}$$

Which can also be described as:

$$P(y|x) = softmax_C \left(\vec{\theta} \cdot \vec{\phi}(x, y) \right)$$

Log-linear model for binary classification

WestlakeNLP

- **Sigmoid function** is an exponential function that maps a number in [-∞, ∞] to [0,1].

$$\sigma(x) = \frac{e^{x}}{1 + e^{x}}$$

- a binary classifier *score(y=+1)=* $\vec{\theta} \cdot \vec{\phi}(x) \in [-\infty, +\infty]$ can be mapped into a probabilistic classifier

$$P(y = +1|x) = \sigma\left(\vec{\theta} \cdot \vec{\phi}(x)\right) = \frac{e^{\vec{\theta} \cdot \vec{\phi}(x)}}{1 + e^{\vec{\theta} \cdot \vec{\phi}(x)}}$$

$$P(y = -1|x) = 1 - \sigma\left(\vec{\theta} \cdot \vec{\phi}(x)\right) = \frac{1}{1 + e^{\vec{\theta} \cdot \vec{\phi}(x)}}$$

Contents

Training log-linear models

WestlakeNLP

- We want to train the parameters $\vec{\theta}$ so that the scores $P(\cdot)$ truly represent probabilities.

- Training examples : $D = \{(x_i, y_i)\}|_{i=1}^{N}$

- Using maximum likelihood estimation (MLE) :
  The training objective is

$$P(Y|X) = \prod_i P(y_i|x_i)$$

  which is maximizing the conditional likelihood of training data.

Training binary log-linear models

WestlakeNLP

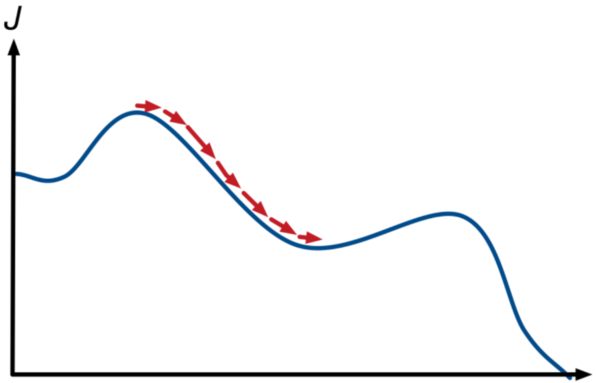
- Given $P(y=+1|x) = \frac{e^{\vec{\theta}\cdot\vec{\phi}(x)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x)}}$, our MLE training objective is

$$P(Y|X) = \prod_i P(y_i|x_i) = \prod_{i^+} P(y=+1|x_i) \prod_{i^-} P(y=-1|x_i)$$

- Maximizing $P(Y \mid X)$ can be achieved by maximizing

$$\begin{aligned}
&\log P(Y \mid X) \\
&= \textstyle\sum_i \log P\,(y_i|x_i) \\
&= \textstyle\sum_{i^+} \log P(y=+1|x_i) + \sum_{i^-} \log P(y=-1|x_i) \\
&= \textstyle\sum_{i^+} \log \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}} + \sum_{i^-} \log \frac{1}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}} \\
&= \textstyle\sum_{i^+} \left(\vec{\theta}\cdot\vec{\phi}(x_i^+) - \log\left(1+e^{\vec{\theta}\cdot\vec{\phi}(x_i^+)}\right)\right) - \sum_{i^-} \log\left(1+e^{\vec{\theta}\cdot\vec{\phi}(x_i^-)}\right)
\end{aligned}$$

Gradient descent



- For log-linear model, the gradient of the objective is :

$$\vec{g} = \frac{\partial logP(Y|X)}{\vec{\theta}}$$

$$= \sum_{i^+}\left(\vec{\phi}(x_i) - \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\vec{\phi}(x_i)\right) - \sum_{i^-}\left(\frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\vec{\phi}(x_i)\right)$$

$$= \sum_{i^+}\left(1 - \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\right)\vec{\phi}(x_i) - \sum_{i^-}\left(\frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\right)\vec{\phi}(x_i)$$

$$= \sum_{i^+}(1 - P(y = +1|x_i))\vec{\phi}(x_i) - \sum_{i^-} P(y = +1|x_i)\vec{\phi}(x_i)$$

Gradient descent

WestlakeNLP

- A simple numerical solution to the minimization of convex functions.

Gradient Descent

Inputs: An objective function F;
Initialization: $\vec{\theta}_0 \leftarrow random()$, $t \leftarrow 0$;
repeat

$\quad \vec{g}_t \leftarrow \nabla_{\vec{\theta}_t} F(\vec{\theta}_t)$;

$\quad \vec{\theta}_{t+1} \leftarrow \vec{\theta}_t - \alpha \vec{g}_t$;

$\quad t \leftarrow t + 1$;

until $||\vec{\theta}_t$ - $\vec{\theta}_{t-1}|| < \epsilon$;
Outputs: $\vec{\theta}_t$;

- Here α is the learning rate (hyper-parameter)
- Finding $\vec{g}$ at each iteration can be computationally inefficient.

Gradient descent

WestlakeNLP

- General numerical optimization method.
  SVM can be optimized with gradient descent too.

- Converges to a local minimum dependent on the initialization.

- Can be slow when the number of training instances N is large.

Stochastic gradient descent (SGD)

WestlakeNLP

Stochastic Gradient Descent

Inputs: An objective function $F(x, y, \vec{\theta})$, and $D = \{(x_i, y_i)\}|_{i=1}^{N}$;

Initialisation: $\vec{\theta}_0 \leftarrow random()$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;

repeat

$\quad \vec{\theta}_{t+1} \leftarrow \vec{\theta}_t$;

$\quad$ **for** $i \in [1, \ldots, N]$ **do**

$$\vec{g}_{t,i} \leftarrow \frac{\partial F(x_i, y_i, \vec{\theta}_{t+1})}{\partial \vec{\theta}_{t+1}};$$

$$\vec{\theta}_{t+1} \leftarrow \vec{\theta}_{t+1} - \alpha \vec{g}_{t,i} ;$$

$\quad t \leftarrow t + 1$;

until $t = T$;

Outputs: $\vec{\theta}_t$

- SGD updates model parameters more frequently, and converge much faster than gradient descents

Stochastic gradient descent (SGD)

WestlakeNLP

SGD for binary classification log-linear model

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialization $\vec{\theta} \leftarrow \vec{0}$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;
repeat
  for $i \in [1 \dots N]$ **do**
    $P(y = +1|x_i) \leftarrow \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}$;
    if $y_i = +1$ **then**
      $\vec{\theta} \leftarrow \vec{\theta} - \alpha(P(y = +1|x_i) - 1)\vec{\phi}(x_i)$;
    else
      $\vec{\theta} \leftarrow \vec{\theta} - \alpha P(y = +1|x_i)\vec{\phi}(x_i)$;
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$

$$\vec{g} = \frac{\partial logP(Y|X)}{\vec{\theta}}$$
$$= \sum_{i^+}\left(\vec{\phi}(x_i) - \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\vec{\phi}(x_i)\right) - \sum_{i^-}\left(\frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\vec{\phi}(x_i)\right)$$
$$= \sum_{i^+}\left(1 - \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\right)\vec{\phi}(x_i) - \sum_{i^-}\left(\frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}\right)\vec{\phi}(x_i)$$
$$= \sum_{i^+}(1 - P(y = +1|x_i))\vec{\phi}(x_i) - \sum_{i^-} P(y = +1|x_i)\vec{\phi}(x_i)$$

Negated for positive samples $(P(y = +1|x_i) - 1)\vec{\phi}(x_i)$
Negated For negative samples $P(y = +1|x_i)\ \vec{\phi}(x_i)$

Contents

WestlakeNLP

Multi-class log-linear models

WestlakeNLP

- For training pairs $\vec{\phi}(x_i, y_i)$, where $y_i \in C, |C| \geq 2$.
  The probability of $y_i = c, c \in C$ is:
  $$P(y_i = c|x_i) = softmax\left(\vec{\theta} \cdot \vec{\phi}(x_i, c)\right) = \frac{e^{\vec{\theta} \cdot \vec{\phi}(x_i, c)}}{\sum_{c' \in C} e^{\vec{\theta} \cdot \vec{\phi}(x_i, c')}}$$
- The log-likelihood of D is
  $$logP(Y|X) = \sum_i logP(y_i|x_i) = \sum_i \left(\vec{\theta} \cdot \vec{\phi}(x_i, y_i) - log\left(\sum_{c \in C} e^{\vec{\theta} \cdot \vec{\phi}(x_i, c)}\right)\right)$$
- For each training example,
  $$logP(y_i|x_i) = \vec{\theta} \cdot \vec{\phi}(x_i, y_i) - log\left(\sum_{c \in C} e^{\vec{\theta} \cdot \vec{\phi}(x_i, c)}\right)$$
- The local gradient is:
  $$\vec{g} = \frac{\partial logP(y_i|x_i)}{\partial \vec{\theta}} = \sum_{c \in C}\left(\vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, c)\right) P(y = c|x_i)$$

Multi-class log-linear models

SGD training for multi-class log-linear models

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialisation $\vec{\theta} \leftarrow \vec{0}$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\vec{g} \leftarrow \vec{0}$;
    for $c \in C$ **do**
      $P(y = c|x_i) \leftarrow \frac{e^{\vec{\theta} \cdot \vec{\phi}(x_i, c)}}{\sum_{c'} e^{\vec{\theta} \cdot \vec{\phi}(x_i, c')}}$;
      $\vec{g} \leftarrow \vec{g} + \left(\vec{\phi}(x_i, c) - \vec{\phi}(x_i, y_i)\right) P(y = c|x_i)$;
    $\vec{\theta} \leftarrow \vec{\theta} - \alpha \vec{g}$;
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$

Mini-batch SGD

A compromise between gradient descent and SGD training.

Split the set of training examples D into several equal-sized subsets $D_1, D_2, \ldots, D_M$, each containing N/M training examples.

Gradient Descend	Mini-batch SGD	SGD
$M=1$	$1<M<N$	$M=N$

The mini-batch size N/M controls the tradeoff between efficiency and accuracy of approximation.

Mini-batch SGD

WestlakeNLP

Mini-batch gradient descent for binary classification

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;

Initialisation: $\vec{\theta} \leftarrow random()$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;

for $i \in [1, \ldots, M]$ **do**

  $D_i \leftarrow \{(x_j, y_j)\}|_{j=1+(i-1)*\frac{N}{M}}^{i*\frac{N}{M}}$;

repeat

  for $i \in [1, \ldots, M]$ **do**

    $\vec{g} \leftarrow \vec{0}$;

    for $j \in [1, \ldots, |D_i|]$ **do**

      $P(y = +1|x_j^i) \leftarrow \frac{e^{\vec{\theta}_t \cdot \vec{\phi}(x_j^i)}}{1+e^{\vec{\theta}_t \cdot \vec{\phi}(x_j^i)}}$;

      if $y_i = +1$ **then**

        $\vec{g} \leftarrow \vec{g} + \left(1 - P(y = +1|x_i^j)\right)\vec{\phi}(x_i^j)$;

      else

        $\vec{g} \leftarrow \vec{g} + P(y = +1|x_i^j)\vec{\phi}(x_i^j)$;

    $\vec{\theta} \leftarrow \vec{\theta} - \alpha\vec{g}$;

  $t \leftarrow t + 1$;

until $t = T$;

Outputs: $\vec{\theta}$

Contents

Review log-linear model

WestlakeNLP

- Target: $P(y|x)$

- Parameterization

$$\begin{cases} P(y|x) = \sigma(\vec{\theta} \cdot \vec{\phi}(x)) \\ P(y|x) = softmax\ (\vec{\theta} \cdot \vec{\phi}(x, y)) \end{cases}$$

- Testing

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y|x)$$

- Training

  MLE using SGD

Test scenario of log-linear models

WestlakeNLP

- Given a test input x, find $\hat{y} = \arg\max_{y \in C} P(y|x)$, which is equal to

  $\arg\max_{y \in C} \vec{\theta} \cdot \vec{\phi}(x, y)$

- The test scenario of log-linear models are identical to those of SVMs and perceptron models

- Binary log-linear models are also called logistic regression

Comparing with Perceptron Binary Classification

WestlakeNLP

binary log-linear models

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialisation: $\vec{\theta} \leftarrow \vec{0}$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\boxed{P(y = +1|x_i) \leftarrow \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}};}$
    if $y_i = +1$ **then**
      $\vec{\theta} \leftarrow \vec{\theta} - \alpha\Big(P(y = +1|x_i) - 1\Big)\vec{\phi}(x_i)$;
    else
      $\vec{\theta} \leftarrow \vec{\theta} - \alpha P(y = +1|x_i)\vec{\phi}(x_i)$;
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$

binary perceptron models

Input: $D = \{(x_i, y_i)\}|_{i=1}^{N}, y_i \in \{-1, +1\}$
Initialization: $\vec{\theta} \leftarrow \vec{0}$; $t \leftarrow 0$
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\boxed{z_i \leftarrow \text{SIGN}(\vec{\theta}\cdot\vec{\phi}(x_i));}$
    if $z_i \neq y_i$ **then**
      $\boxed{\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(x_i) \times y_i;}$
  $t \leftarrow t + 1$;
until $t = T$;

Comparing with Perceptron Multi-class Classification

WestlakeNLP

multi-class log-linear models

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialisation $\vec{\theta} \leftarrow \vec{0}$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\vec{g} \leftarrow \vec{0}$;
    for $c \in C$ **do**
      $\boxed{P(y = c|x_i) \leftarrow \frac{e^{\vec{\theta} \cdot \vec{\phi}(x_i, c)}}{\sum_{c'} e^{\vec{\theta} \cdot \vec{\phi}(x_i, c')}};}$
      $\boxed{\vec{g} \leftarrow \vec{g} + \left(\vec{\phi}(x_i, c) - \vec{\phi}(x_i, y_i)\right) P(y = c|x_i);}$
    $\vec{\theta} \leftarrow \vec{\theta} - \alpha \vec{g}$;
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$

multi-class perceptron models

Input: $D = \{(x_i, c_i)\}|_{i=1}^{N}$, $c_i \in C$
Initialization: $\vec{\theta} \leftarrow \vec{0}$; $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\boxed{z_i \leftarrow \arg\max_{z} \vec{\theta} \cdot \vec{\phi}(x_i, z);}$
    if $z_i \neq c_i$ **then**
      $\boxed{\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(x_i, c_i) - \vec{\phi}(x_i, z_i);}$
  $t \leftarrow t + 1$;
until $t = T$;

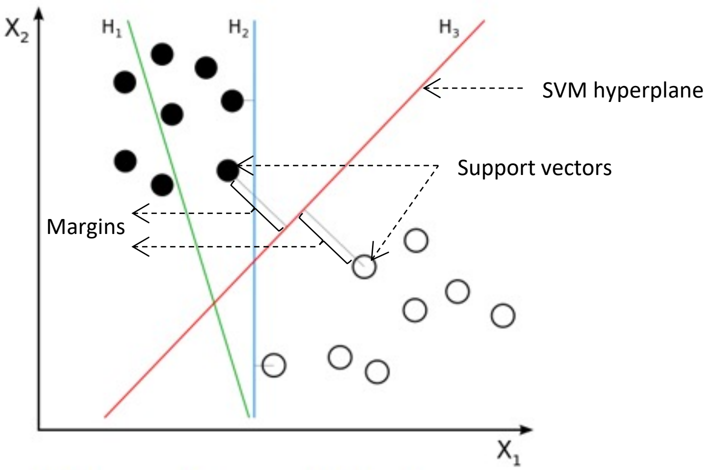
Contents

WestlakeNLP

SVM recap

A linear model for binary classification in vector space

- To maximize **margin** (distances form support vectors to hyperplane)
- To minimize **violation**, i.e., ensuring all data residing to the right side



H1 does not separate the classes.
H2 does, but only with a small margin.
H3 separates them with the maximum margin.

Binary classification SVM

WestlakeNLP

- The training objective of binary classification SVM :

  Minimizing $\frac{1}{2}\left|\left|\vec{\theta}\right|\right|^2 + C\sum_i \max\left(0,1 - y_i\left(\vec{\theta}\cdot\vec{\phi}(x_i)\right)\right)$ given $D = \{(x_i, y_i)\}|_{i=1}^{N}$

- Equivalent to minimizing

  $\sum_i \max\left(0,1 - y_i\left(\vec{\theta}\cdot\vec{\phi}(x_i)\right)\right) + \frac{1}{2}\lambda|\vec{\theta}|^2$

  hyper-parameters of the model $\lambda = \frac{1}{c}$

- Optimization :
  - stochastic gradient descent. SGD
  - derive local training objective for each train example

Binary classification SVM

WestlakeNLP

- The training objective of binary classification SVM :

  Minimizing $\frac{1}{2}\left|\left|\vec{\theta}\right|\right|^2 + C\sum_i \max\left(0, 1 - y_i\left(\vec{\theta}\cdot\vec{\phi}(x_i)\right)\right)$ given $D = \{(x_i, y_i)\}|_{i=1}^{N}$

- Equivalent to minimizing

  $\sum_i \max\left(0, 1 - y_i\left(\vec{\theta}\cdot\vec{\phi}(x_i)\right)\right) + \frac{1}{2}\lambda|\vec{\theta}|^2$

  hyper-parameters of the model $\lambda = \frac{1}{c}$

- Optimization :
  - stochastic gradient descent. SGD
  - derive local training objective for each train example
- Sub-gradient :

$$\begin{cases} \lambda\vec{\theta} & \text{if } 1 - y_i\vec{\theta}\cdot\vec{\phi}(x_i) \leq 0 \\ \lambda\vec{\theta} - y_i\vec{\phi}(x_i) & \text{otherwise} \end{cases}$$

Binary classification SVM

SGD training for binary classification SVM

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialisation: $\vec{\theta} \leftarrow \vec{0}$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    if $y_i \vec{\theta} \cdot \vec{\phi}(x_i) < 1$ **then**
      $\vec{\theta} \leftarrow \vec{\theta} - \alpha\Big(\lambda\vec{\theta} - y_i\vec{\phi}(x_i)\Big)$;
    else
      $\vec{\theta} \leftarrow \vec{\theta} - \alpha\lambda\vec{\theta}$;
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$

Multi-class classification

WestlakeNLP

- The training objective of multi-class SVM is to minimize

$$\frac{1}{2}\left|\vec{\theta}\right|^2 + C\sum_i \max\left(0,1-\vec{\theta}\cdot\vec{\phi}(x_i,y_i)+\max_{c\neq y_i}\vec{\theta}\cdot\vec{\phi}(x_i,c)\right)$$

- Equivalent to minimizing

$$\sum_i \max\left(0,1-\vec{\theta}\cdot\vec{\phi}(x_i,y_i)+\max_{c\neq y_i}\vec{\theta}\cdot\vec{\phi}(x_i,c)\right)+\frac{1}{2}\lambda|\vec{\theta}|^2$$

where $(x_i,y_i)\in D, \lambda=\frac{1}{C}$.

- Sub-gradient for each training example:

$$\begin{cases} \lambda\vec{\theta} & \text{if } 1-\vec{\theta}\cdot\vec{\phi}(x_i,y_i)+\quad\vec{\theta}\cdot\vec{\phi}\,(x_i,z_i)\leq 0 \\ \lambda\vec{\theta}-\left(\vec{\phi}\,(x_i,y_i)-\vec{\phi}\,(x_i,z_i)\right) & \text{otherwise} \end{cases}$$

where $\mathrm{z_i}=\arg\max_{c\neq y_i}\vec{\theta}\cdot\vec{\phi}(x_i,c)$.

Multi-class classification SVM

WestlakeNLP

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}, y_i \in C$;

Initialisation $\vec{\theta} \leftarrow 0, t \leftarrow 0$;

repeat

  for $i \in [1, \ldots, N]$ **do**

    $\vec{g} \leftarrow \vec{0}$;

    $z_i \leftarrow \arg\max_{c \neq y_i} \vec{\theta} \cdot \vec{\phi}(x_i, c)$;

    if $\vec{\theta} \cdot \vec{\phi}(x_i, y_i) - \vec{\theta} \cdot \vec{\phi}(x_i, z_i) < 1$ **then**

      $\vec{g} \leftarrow \vec{g} - \left(\vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, z_i)\right)$;

    $\vec{\theta} \leftarrow \vec{\theta} - \alpha(\vec{g} + \lambda\vec{\theta})$;

  $t \leftarrow t + 1$;

until $t = T$;

Outputs: $\vec{\theta}$;

Contents

WestlakeNLP

Comparing with Perceptron Binary Classification

binary SVM models

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialisation: $\vec{\theta} \leftarrow \vec{0}$, $\alpha \leftarrow \alpha_0$, $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\boxed{\begin{array}{l} \textbf{if } y_i\vec{\theta} \cdot \vec{\phi}(x_i) < 1 \textbf{ then} \\ \quad \vec{\theta} \leftarrow \vec{\theta} - \alpha\left(\lambda\vec{\theta} - y_i\vec{\phi}(x_i)\right); \\ \textbf{else} \\ \quad \vec{\theta} \leftarrow \vec{\theta} - \alpha\lambda\vec{\theta}; \end{array}}$
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$

binary perceptron models

Input: $D = \{(x_i, y_i)\}|_{i=1}^{N}, y_i \in \{-1, +1\}$
Initialization: $\vec{\theta} \leftarrow \vec{0}$; $t \leftarrow 0$
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\boxed{z_i \leftarrow \textsc{Sign}(\vec{\theta} \cdot \vec{\phi}(x_i));}$
    if $z_i \neq y_i$ **then**
      $\boxed{\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(x_i) \times y_i;}$
  $t \leftarrow t + 1$;
until $t = T$;

Comparing with Perceptron Multi-class Classification

WestlakeNLP

multi-class SVM models

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}, y_i \in C$;
Initialisation $\vec{\theta} \leftarrow 0$, $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\vec{g} \leftarrow \vec{0}$;
    $\boxed{z_i \leftarrow \arg\max_{c \neq y_i} \vec{\theta} \cdot \vec{\phi}(x_i, c);}$
    if $\vec{\theta} \cdot \vec{\phi}(x_i, y_i) - \vec{\theta} \cdot \vec{\phi}(x_i, z_i) < 1$ **then**
      $\boxed{\vec{g} \leftarrow \vec{g} - (\vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, z_i));}$
    $\vec{\theta} \leftarrow \vec{\theta} - \alpha(\vec{g} + \lambda\vec{\theta})$;
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$;

multi-class perceptron models

Input: $D = \{(x_i, c_i)\}|_{i=1}^{N}, c_i \in C$
Initialization: $\vec{\theta} \leftarrow \vec{0}$; $t \leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\boxed{z_i \leftarrow \arg\max_{z} \vec{\theta} \cdot \vec{\phi}(x_i, z)\ ;}$
    if $z_i \neq c_i$ **then**
      $\boxed{\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(x_i, c_i) - \vec{\phi}(x_i, z_i);}$
  $t \leftarrow t + 1$;
until $t = T$;

- Comparison with perceptron models
  - Checks if $y_i\left(\vec{\theta} \cdot \vec{\phi}(x_i)\right) \leq 1$ instead of $y_i\left(\vec{\theta} \cdot \vec{\phi}(x_i)\right) \leq 0$
  - Additional regularization term $\lambda\vec{\theta}$
  - A learning rate α to weight the parameter update

A perceptron training objective function

WestlakeNLP

- Perceptron updates can also be viewed as SGD training of a certain objective function.

- The training objective is to minimize

$$\sum_{i=1}^{N} \max\left(0, -\vec{\theta} \cdot \vec{\phi}(x_i, y_i) + \max_{c} \vec{\theta} \cdot \vec{\phi}(x_i, c)\right)$$

Contents

Generalized linear classification model

- Discriminative models
  - Perceptron
  - SVM
  - Log-linear models

- Model form identical

$$score(y|x) \propto f(\vec{\theta} \cdot \vec{\phi}(x, y))$$

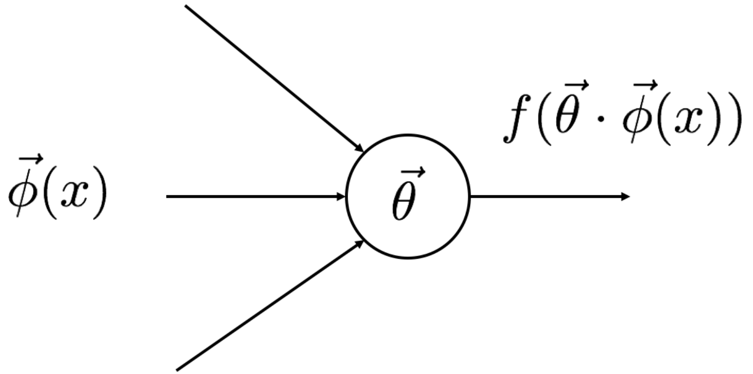
- Testing scenario, criteria identical

$$y = \arg\max_{y'} \; score(y'|x) = \arg\max_{y'} \vec{\theta} \cdot \vec{\phi}(x, y')$$

- Training different

Generalized linear classification model

Model and Testing (binary classification case)



- parameter vector $\vec{\theta}$
- feature vector $\vec{\phi}$
- output class label y using the dot product $\vec{\theta} \cdot \vec{\phi}$
- f – activation function

Contents

Unified Online Training

WestlakeNLP

- All by SGD training

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialization: $\vec{\theta} \leftarrow \vec{0}$, $t \leftarrow 0$;
repeat
  for $i \in [1 \ldots N]$ **do**
    ParameterUpdate(x_i, y_i);
  $t \leftarrow t + 1$;
until $t = T$;
Outputs: $\vec{\theta}$;

- Given a set of training data $D = \{(x_i, y_i)\}|_{i=1}^{N}$, the algorithm goes over D for T iterations, processing each training example (x_i, y_i), and update model parameters when necessary.

Unified Online Training

WestlakeNLP

Parameter Update (x_i, y_i) for perceptrons, SVMs and log-linear models

Feature	Model	Update rule
Binary classification	Perceptron	$y_i\vec{\phi}(x_i)$ **if** $y_i\left(\vec{\theta}\cdot\vec{\phi}(x_i)\right) < 0$
	SVM	$\begin{cases} \alpha y_i\vec{\phi}(x_i) - \alpha\lambda\vec{\theta} \ \textbf{if} \ y_i\left(\vec{\theta}\cdot\vec{\phi}(x_i)\right) \leqslant 1 \\ -\alpha\lambda\vec{\theta} \quad \text{otherwise} \end{cases}$
	Log-linear models	$\begin{cases} \alpha(1 - P(y=+1\|x_i))\vec{\phi}(x_i) \ \textbf{if} \ y_i = +1 \\ \alpha(-P(y=+1\|x_i))\vec{\phi}(x_i) \ \text{otherwise} \end{cases}$

Unified Online Training

WestlakeNLP

Multi-class classification	Perceptron	$\vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, z_i)$ **if** $z_i \neq y_i$ $z_i = \arg\max_{\mathbf{c}} \vec{\theta} \cdot \vec{\phi}(x_i, \mathbf{c})$
	SVM	$\begin{cases} \alpha\Big(\vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, \mathbf{c})\Big) - \alpha\lambda\vec{\theta} \\ \quad \textbf{if } \vec{\theta} \cdot \vec{\phi}(x_i, y_i) - \vec{\theta} \cdot \vec{\phi}(x_i, z_i) \leqslant 1 \\ -\alpha\lambda\vec{\theta} \quad \text{otherwise} \end{cases}$ $z_i = \arg\max_{\mathbf{c} \neq y_i} \vec{\theta} \cdot \vec{\phi}(x_i, \mathbf{c})$
	Log-linear models	$\alpha \sum_{\mathbf{c}} \left(\vec{\phi}(x_i, y_i) - \vec{\phi}(x_i, \mathbf{c})\right) P(y = \mathbf{c} \mid x_i)$

Contents

Loss Functions

WestlakeNLP

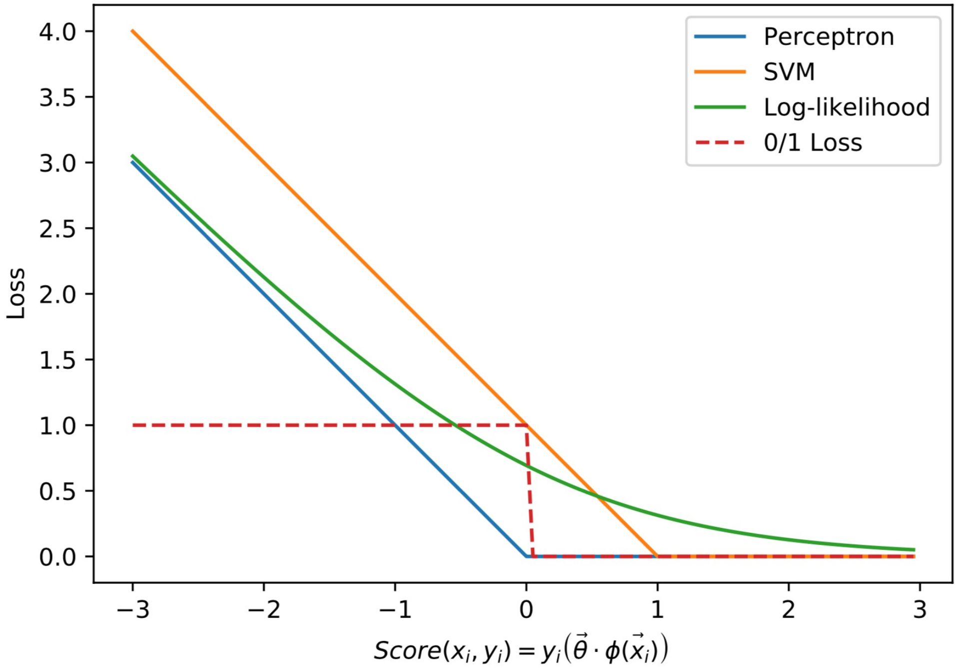
- The training objectives for linear models can be regarded as to minimize different **loss functions** of a model over a training set.

Feature	Model	Loss function
Binary classification	Perceptron	$\sum_{i=1}^{N} \max\left(0, -y_i \vec{\theta} \cdot \vec{\phi}(x_i)\right)$
	SVM	$\sum_{i=1}^{N} \max\left(0, 1 - y_i \vec{\theta} \cdot \vec{\phi}(x_i)\right) + \frac{1}{2}\lambda \vert\vert\vec{\theta}\vert\vert^2$
	Log-linear models	$\sum_{i=1}^{N} \log(1 + e^{-y_i \vec{\theta} \cdot \vec{\phi}(x_i)})$
Multi-class classification	Perceptron	$\sum_{i=1}^{N} \max\Big(0, -\vec{\theta} \cdot \vec{\phi}(x_i, y_i) + \vec{\theta} \cdot \vec{\phi}(x_i, \arg\max_{\mathbf{c}} \vec{\theta} \cdot \vec{\phi}(x_i, \mathbf{c}))\Big)$
	SVM	$\sum_{i=1}^{N} \max\Big(0, 1 - \vec{\theta} \cdot \vec{\phi}(x_i, y_i) + \max_{\mathbf{c} \neq y_i} \vec{\theta} \cdot \vec{\phi}(x_i, \mathbf{c})\Big) + \frac{1}{2}\lambda \vert\vert\vec{\theta}\vert\vert^2$
	Log-linear models	$\sum_{i=1}^{N} \Big(\log\big(\sum_{\mathbf{c}} e^{\vec{\theta} \cdot \vec{\phi}(x_i, \mathbf{c})}\big) - \vec{\theta} \cdot \vec{\phi}(x_i, y_i)\Big)$

Different types of loss functions

WestlakeNLP

- **Hinge loss**: the loss functions of SVMs and perceptrons
- **Log-likelihood loss**: the loss functions for log-linear models
- ***0/1* loss**: loss is 1 for an incorrect output and 0 for a correct output.



Risks

WestlakeNLP

- The true **expected risk** of a linear model with parameter can be formulated as

$$risk(\vec{\theta}) = \sum_{x,y} loss\left(\vec{\theta} \cdot \phi(x,y)\right) P(x,y),$$

which cannot be calculated, we use **empirical risk** as a proxy

$$\widetilde{risk}(\vec{\theta}) = \frac{1}{N} \sum_{i=1}^{N} loss\left(\vec{\theta} \cdot \phi(x_i, y_i)\right), (x_i, y_i) \in D$$

Contents

Regularization

- SVM training objective

$$L = \sum_{i=1}^{N} \max\left(0, 1 - y_i \vec{\theta} \cdot \vec{\phi}(x_i)\right) + \frac{1}{2}\lambda ||\vec{\theta}||^2$$

- A large element in the parameter vector $\vec{\theta}$ implies higher reliance of the model to its corresponding feature, sometimes unnecessarily much.

  L2 regularization : $\frac{1}{2}\lambda \left|\left|\vec{\theta}\right|\right|^2$

  and **L1 regularization** : $\lambda \left|\left|\vec{\theta}\right|\right|_1$

  minimize a polynomial of $\vec{\theta}$ in loss functions,
  reduce over-fitting of models on given training data.

Contents

WestlakeNLP

Comparing model performances

WestlakeNLP

- Different models results from:
  - different training objectives (large margin or log-likelihood)
  - different feature definitions
  - different hyperparameters (number of training iterations, learning rate)
- Need to compare accuracies on testset
- Can make combination to exploit complementary strengths

Significance test

WestlakeNLP

- Model A 93% Model B 92%

  Model A is better?

- Null hypothesis

  The probability of null hypothesis – significance level

  $p < 0.01$ / $p<0.05$ / $p < 0.001$ / $p < 0.000001$

- Use the set of test results to evaluate the probability of null hypothesis.

- t-test

Hands on

WestlakeNLP

T test using python and numpy

import the packages

```
import numpy as np
from scipy import stats
```

Define 2 random distributions with size N

```
N = 10
a = np.random.randn(N) + 2
b = np.random.randn(N)
```

```
print(a,b)

[1.36929026 0.45171207 1.980284   2.69894659 3.58459313
1.02503278 1.5443674  3.75178741 0.79744317 1.48595977]

[ 1.97056194 -0.44926516 -0.1049745  -1.02613361 -2.1329443
0.7991724 0.60720041  1.22176851 -1.22300267 -0.21052362]
```

Hands on

WestlakeNLP

Calculate the standard derivation

```
var_a = a.var(ddof=1)
var_b = b.var(ddof=1)
s = np.sqrt((var_a + var_b)/2)
```

Calculate the t-statistics and compare with the critical t-value

```
t = (a.mean() - b.mean())/(s*np.sqrt(2/N))

df = 2*N -2
p = 1 - stats.t.cdf(t, df=df)
```

Results:

```
print("t = " + str(t))
print("p = " + str(2*p))

t = 4.26411975457851
p = 0.0004668051407590301
```

Hands on

WestlakeNLP

Cross checking with the internal scipy function

```
t2, p2 = stats.ttest_ind(a, b)
```

Results:

```
print("t = " + str(t2))
print("p = " + str(p2))

t = 4.264119754578509
p = 0.0004668051407589875
```

Contents

Ensemble models

WestlakeNLP

- **Ensemble approach**: a combination of multiple models for better accuracies.
- **Voting**: a simple method to ensemble different models.

  Given a set of models $M = (m_1, m_2, \ldots m_{|M|})$ and output classes $C = (c_1, c_2, \ldots, c_{|C|})$, the output class y for a given input x can be decided by counting the vote (*hard 0/1votes*):

  $v_i = \sum_{j=1}^{|M|} \mathrm{b1}(y(m_j), c_i)$

  - *majority voting* chooses the class label that receives more than half the total votes
  - *plurality voting* chooses the class with the most votes.
  - More fine-grained voting methods are soft voting and weighted voting.

Ensemble models

- **Stacking**:

  use the outputs of one model as features to inform another model.

  - **Training for stacking**

    the stacking method trains A after B is trained.

  - We use **K-fold jackknifing** to make model B output accuracies on the training data as close to the test scenario as possible.

Training set	**Model**	**Test set**
$D_2, \ldots, D_k$	B_1	D_1
$D_1, D_3, \ldots, D_k$	B_2	D_2
$\ldots$	$\ldots$	$\ldots$
$D_1, D_2, \ldots, D_{k-1}$	B_k	D_k

Ensemble models

WestlakeNLP

- **Bagging** use different subsets of D to obtain different models and then ensemble them. Voting is then performed between models given a test case. Bagging can outperform a single model for many tasks.

Model	**Feature Type**	**Features**
B	Bag-of-words	w_1 =Ronaldo, w_2 =donated, w_3 =€, w_4 =600,000, w_5 =to, w_6 =charity
A	Bag-of-words + B's output label	w_1 =Ronaldo, w_2 =donated, w_3 =€, w_4 =600,000, w_5 =to, w_6 =charity, y_B =sports
A	Bag-of-words + B's probability outputs	w_1 =Ronaldo, w_2 =donated, w_3 =€, w_4 =600,000, w_5 =to, w_6 =charity, $P(y_B = sports) \in [0.6, 0.7]$, $P(y_B = finance) \in [0.2, 0.3]$

Co-training and self-training

WestlakeNLP

- *Data augmentation.*
- *Semi-supervised* learning use different models trained on *D* to predict the labels on a set of unlabeled data *U*, augmenting *D* with the outputs that most models agree on.
- the more accurate the baseline models are on *U*, the more likely that the new data form *U* can be correct and useful.

Co-training

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$, $U = \{x'_i\}|_{i=1}^{M}$, models A and B;
Initialization: t $\leftarrow$ 0;
repeat
  t $\leftarrow$ t+1;
  $\textsc{Train}(A,\ D)$;
  $\textsc{Train}(B,\ D)$;
  for $x'_i \in U$ **do**
    $z'_A \leftarrow \textsc{Predict}(A,\ x'_i)$;
    $z'_B \leftarrow \textsc{Predict}(B,\ x'_i)$;
    if $z'_A = z'_B = z'_i$ **and** $\textsc{Confident}(A, x'_i, z'_i)$ **and** $\textsc{Confident}(B, x'_i, z'_i)$ **then**
      $\textsc{Add}(D,\ (x'_i, z'_i))$;
      $\textsc{Remove}(U,\ (x'_i))$;
until $t = T$;

Self-training

WestlakeNLP

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$, $U = \{x'_i\}|_{i=1}^{M}$, model A;
Initialisation: t $\leftarrow$ 0;
repeat
  t $\leftarrow$ t+1;
  $\textsc{Train}(A,\ D)$;
  for $x'_i \in U$ **do**
    $z'_i \leftarrow \textsc{Predict}(A,\ x'_i)$;
    if $\textsc{Confident}(A, x'_i, z'_i)$ **then**
      $\textsc{Add}(D,\ (x'_i, z'_i))$;
      $\textsc{Remove}(U,\ (x'_i))$;
until $t = T$;

Summary

- Log-linear models for binary and multi-class classification
- Stochastic Gradient Descent (SGD) training of log-linear models and SVMs
- A generalized linear discriminative model for text classification
- The correlation between SVMs, perceptrons and log-linear models in terms of training objective (loss) functions and regularization terms
- Significance testing
- Ensemble methods for integrating different models