L3 wtr

TianruiWang

May 2019

1 Logistic Regression

1.1 Mathematical Foundation

Bayes Formula
$P(B_i|A) = P(B_i)P(A|B_i)/P(A)$
$P(A) = \sum_{j=1}^{n} P(B_j)P(A|B_j)$
sometimes P(B$_i|A)$ *is too difficult to figure*
so we use Bayes Formula to simplify it
Sigmoid Function
$Sigmoid\ Function: \ f(x) = 1/(1+e^{-x})$
We can use Sigmoid Function to judge the probabilities of classification
and take derivative for any times

The Exponential Family
seems like
p(y;$\eta) = b(y)exp(\eta^T T(y) - a(\eta))$
$b(y)$ *is a given function*
η*is a Natural parameter*
$T(y)$ *is called Sufficient Statistic usually* $T(y) = y$
$a(\eta)$ *is to ensure the sum equal to* 1
for example
Bernoulli distribution
$f(y;p) = p^y(1-p)^{1-y}$
$= exp(yln(p/(1-p)) + ln(1-p))$

1.2 Logistic Regression

Symbol
let X=(x$_1, x_2....)$ *and* Y *is boolen*
Assume $P(x_i|Y = y_k)$ *as* $N(\mu_{ik}, \sigma_i)$
From log to linear
$P(Y = 0|X) = exp(w_0 + \sum iw_ix_i)$
$ln(P(Y = 0|x)/P(Y = 1|X)) = w_0 + \sum iw_ix_i$

$Our\ target\ is\ to\ select\ a\ set\ of\ W = (w_1, w_2...)$
$which\ maximizes\ \sum_i P(y_i|x_i; W)$
$We\ can\ understand\ it\ by\ maximizing\ P(output = y_i)\ when\ input = x_i$
log-likelihood
$define\ l(W) = \sum_l y_l lnP(y_l = 1|x_l; W) + (1 - y_l)lnP(y_l = 0|x_l; W)$
$only\ one\ of\ both\ can\ be\ non-zero\ the\ bigger\ the\ expression\ is\ the\ better\ our\ function\ is$
$l(W) = \sum_l y_l(w_0 + \sum_i^n w_i x_{il}) - ln(1 + exp(w_0 + \sum_i^n w_i x_{il}))$
$However\ the\ function\ has\ no\ closed\ form\ (which\ means\ we\ can't\ figure\ it\ out\ easily)$
Gradient Descent Method
$this\ function\ is\ a\ concave\ function$
$the\ gradient\ descent\ method\ can\ be\ applied\ to\ find\ the\ answer$
$Note:$
$We\ should\ select\ proper\ step\ length\ in\ case\ of\ low\ efficient\ or\ away\ from\ minimum$
$Algorithm$
$Iterate\ until\ change\ <\ \epsilon$
$For\ all\ i\ repeat\ w_i < -W_i + \eta \sum_i x_{il}(y_l - P(y_l = 1|x_i; W))$
$End\ for$
Regularization
$W < -argmax \sum_l lnP(y_l|x_l; W) - \lambda/2||W||^2$
$We\ add\ \lambda/2||W||^2\ because\ if\ ||W||\ is\ too\ big,\ the\ model\ will\ overfit$
Discrete Values
$change\ the\ samples\ in\ [0, 1]$
$for\ every\ k$
$P(y = y_R|X) = 1/(1 + \sum_{j=1}^{R-1} exp(w_{j0} + \sum w_{ji}x_i))$

2 Boosting

Comprehension
A method to train a ensembling module with many individual learners

2.1 Common Method

Bagging
f(x)=1/B $\sum_{b=1}^{B} f_b(x)$
$1 - 1/e \approx 63.2\%$
$Algorithm$
$Input:$
$Training\ Set:\ \{(x1, y1), (x2, y2)...\}$
$Learning\ algorithm:\ l$
$Training\ times:\ T$
$Procedure:$
$for\ t = 1, 2..., T$
$h_t = l(D, D_b s)$

$We\ divide\ the\ set\ into\ two\ parts,\ D_b s\ is\ the\ remaining\ part$
$P(H(x)! = f(x)) = P(x <= T/2) = ...$
$using\ Hoeffding\ inequality$
$<= exp(-1/2T(1-2\epsilon)^2)$
$However,\ those\ methods\ are\ not\ independant,\ so\ the\ inequality\ has\ no\ siginificance$

2.2 Adaboost Algorithm

Input: The training set D
A weak base learner h=h(x;θ)
$Initialize:\ w_i = 1/N$
$Iterate\ for\ t = 1, 2, ...T$
1. $train\ base\ learner\ according\ to\ weighted\ example\ set\ and\ obtain\ hypothesis$
2. $compute\ error\ \epsilon_t = \sum_{i=1}^{n} w_{mi} I(G_m(x_i)! = (y_i))$
3. $compute\ weight\ \alpha_t = 1/2\ ln\epsilon_t/1 - \epsilon_t$
4. $update\ example\ weights\ for\ next\ iterate$
$How\ to\ update?$
$\epsilon_k = \sum_{i=1}^{n} w_i^{k-1} I(y_i! = h(x_i;\theta_k))/\sum_{i=1}^{n} w_i^{k-1}$
$\alpha_t = 1/2\ ln\epsilon_t/1 - \epsilon_t$
$w_i^k = w_i^{k-1} exp(-y_i \alpha_k h(x_i;\theta))$

$w_i^k\ means\ in\ the\ i_{th}\ iteration\ in\ k_{th}\ sample$

2.3 Expotential Loss

L(h)=$\sum_{i=1}^{n} exp(-y_i h_m(x_i))$
$h(x) = \sum_{i=1}^{m} \alpha_i h(x;\theta_i)/\sum_{i=1}^{m} \alpha_i$