Classification and Regression

Sunday, October 28, 2018 2:42 PM

2017 and 2016:

1.

Point Loss $\mathcal{L}1(\theta,\theta_0;x,y)$	Predictor	Technique	Algorithm*
$\mathcal{L}_S(y-(\theta^\top x+\theta_0))$	$f(x;\theta,\theta_0)=\theta^\top x+\theta_0$	Linear Regression	Exact Solution, Gradient Descent
$\mathcal{L}_S(y-(\theta^\top x+\theta_0))+\frac{\lambda}{2}\lVert\theta\rVert^2$	$f(x;\theta,\theta_0)=\theta^\top x+\theta_0$	Ridge Regression	Exact Solution, Gradient Descent
$\mathcal{L}_H(y(\theta^\top x+\theta_0))$	$h(x;\theta,\theta_0)=\text{sign}(\theta^\top x+\theta_0)$	Linear Classification using Hinge Loss	Gradient Descent
$\mathcal{L}_H(y(\theta^\top x+\theta_0))+\frac{\lambda}{2}\lVert\theta\rVert^2$	$h(x;\theta,\theta_0)=\text{sign}(\theta^\top x+\theta_0)$	Support Vector Machine with Slack Variables	Gradient Descent
$\mathcal{L}_Z(y(\theta^\top x+\theta_0))$	$h(x;\theta,\theta_0)=\text{sign}(\theta^\top x+\theta_0)$	Perceptron (with Offset)	Perceptron Algorithm
$\mathcal{L}_L(y(\theta^\top x+\theta_0))$	$p(y\|x,\theta,\theta_0)=\text{sigmoid}(y(\theta^\top x+\theta_0))$	Logistic Regression	Gradient Descent

Learning Cost
$\mathcal{L}_n(\theta,\theta_0)=\frac{1}{n}\sum_{\text{data }(x,y)}\ \frac{1}{2}(y-(\theta^\top x+\theta_0)$
$\mathcal{L}_n(\theta,\theta_0)=\frac{1}{n}\sum_{\text{data }(x,y)}\ \frac{1}{2}(y-(\theta^\top x+\theta_0))$
$\mathcal{L}_n(\theta,\theta_0)=\frac{1}{n}\sum_{\text{data }(x,y)}\ \max\{1-y(\theta^\top x+$
$\mathcal{L}_n(\theta,\theta_0)=\frac{1}{n}\sum_{\text{data }(x,y)}\ \max\{1-y(\theta^\top$
$\mathcal{L}_n(\theta,\theta_0)=\frac{1}{n}\sum_{\text{data }(x,y)}\ [\![\ y(\theta^\top x+\theta_0)$

2.

Gradient for Linear Regression without the offset θ_0:	$\nabla_\theta\mathcal{L}_n(\theta;\mathcal{S}_n)=\frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n}x(\theta^\top x-y)$
Gradient for Logistic Regression without the	$\nabla_\theta\mathcal{L}_n(\theta;\mathcal{S}_n)=\frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n}x(\text{sigmoid}(\theta^\top x)-[\![y=1]\!])$

2015:

[Ridge Regression]

Ridge regression optimizes what objective function?

=>

mean squared error with squared ℓ_2-norm $\|w\|_2^2 = \sum_{d=1}^{D} w_d^2$ on weights w

[Regression Error]

Why is the following measure no good objective function for measuring the error in a regression problem ? The error is computed between ground truth yi and prediction $f(x_i)$ as given by the function

$$E=\frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i)^3$$

Hint: you can imagine what can happen if this objective is used with a linear model: $f(x_i)=w^\top x_i$.

Ans: The loss function z^3 is negative for negative errors z. Since the objective is to minimize the loss, the trained parameters will end up favoring large negative errors.

[Logistic Regression]

1. We consider here a discriminative approach for solving the classification problem illustrated in Figure 1

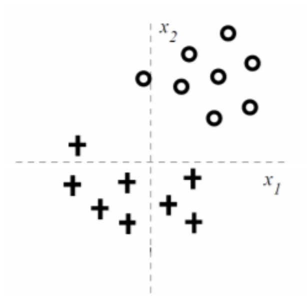

Figure 1: The 2-dimensional labeled training set, where '+' corresponds to class y=1 and 'O' corresponds to class y = 0.

a. We attempt to solve the binary classification task depicted in Figure 1 with the simple linear logistic regression model

Notice that the training data can be separated with zero training error with a linear separator. Consider training regularized linear logistic regression models where we try to maximize

$$\sum_{i=1}^{n} \log \left(P(y_i|x_i, w_0, w_1, w_2)\right) - Cw_j^2$$

for very large C. The regularization penalties used in penalized conditional log-likelihood estimation are $-Cw_j^2$, where $j = \{0, 1, 2\}$. In other words, only one of the parameters is regularized in each case. Given the training data in Figure 1, how does the training error change with regularization of each parameter w_j? State whether the training error increases or stays the same (zero) for each w_j for very large C. Provide a brief justification for each of your answers.

i. By regularizing w_2:

Ans: Increases. When we regularize w_2, the resulting boundary can rely less and less on the value of x_2 and therefore becomes more vertical. For very large C, the training error increases as there is no good linear vertical separator of the training data

ii. By regularizing w_1:

Ans: Remains the same. When we regularize w_1, the resulting boundary can rely less and less on the value of x_1 and therefore becomes more horizontal and the training data can be separated with zero training error with a horizontal linear separator.

iii. By regularizing w_0:

Ans: Increases. When we regularize w_0, then the boundary will eventually go through the origin (bias term set to zero). Based on the figure, we can not find a linear boundary through the origin with zero error. The best we can get is one error.

b.

L1 regularization => Lasso Regression

Lasso Regression (Least Abs

"absolute value of magnitude"

function.

$\sum_{i=1}^{n}($

L2 regularization => Ridge Regression

c.

[**3 pts**] For very large C, with the same L1-norm regularization for w_1 and w_2 as above, which value(s) do you expect w_0 to take? Explain briefly. (Note that the number of points from each class is the same.) (You can give a range of values for w_0 if you deem necessary).

SOLUTION: For very large C, we argued that both w_1 and w_2 will go to zero. Note that when $w_1 = w_2 = 0$, the log-probability of labels becomes a finite value, which is that when $w_1 = w_2 = 0$, the log-probability of labels becomes a finite value, which is equal to n log(0.5), i.e. $w_0 = 0$. In other words, $P(y = 1|\vec{x}, \vec{w})$=$P(y = 0|\vec{x}, \vec{w})$=0.5. We expect so because the number of elements in each class is the same and so we would like to predict each one with the same probability, and w_0=0 makes $P(y = 1|\vec{x}, \vec{w})$=0.5.

d.

[3 pts] Assume that we obtain more data points from the '+' class that corresponds to y=1 so that the class labels become unbalanced. Again for very large C, with the same L1-norm regularization for w_1 and w_2 as above, which value(s) do you expect w_0 to take? Explain briefly. (You can give a range of values for w_0 if you deem necessary).

SOLUTION: For very large C, we argued that both w_1 and w_2 will go to zero. With unbalanced classes where the number of '+' labels are greater than that of 'o' labels, we want to have $P(y = 1|\vec{x}, \vec{w}) > P(y = 0|\vec{x}, \vec{w})$. For that to happen the value of w_0 should be greater than zero which makes $P(y = 1|\vec{x}, \vec{w}) > 0.5$.