Lecture 2 gradient descent and loss functions

wbg231

December 2022

1 introduction

- recall that in practice we are working with constrained erm where we are trying to minimize the risk given some loss function over a hypothesis space

gradient descent

- here we assume that our gradient function is differentiable and our goal is to learn the parameter x such that

$$x = argmin_{x \in \mathbb{R}^d} f(x)$$

Gradient Descent

- Initialize $x \leftarrow 0$.
- Repeat:
  - $x \leftarrow x - \eta \nabla f(x)$
- until the stopping criterion is satisfied.

-
- where $\eta \in \mathbb{R}$ is the step size hyper parameter
- for a fixed step this will eventually converge if the step size is small enough (but may take a long time)
- given the function is convex differentiable and Lipschitz continuous we can actually bound how long this converge will take.
- when to stop gradient descent is an open question, one choice is to wait until the magnitude of the gradient is below a certain threshold,

- early stopping is another method which after each iteration evaluates the validation error and stops when the validation error starts to increase
- doing gradient descent for constrained erm at every iteration we need to compute the sum of the gradients of our loss function with respect to our weights over the whole training set this can take a long time for large training sets so at every step we make n computations so if we do i iterations then full batch gradient descent is $O(ni)$ which does not scale

stochastic gradient descent

- mini batch gradient descent given a random subsample of the training data (called a mini batch) $((x_1, y_1) \cdots (x_n, y_n))$ the mini batch gradient is

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla_w \ell(f_w(x_{m_i}, y_{m_i})$$

  that is at every step we just take a random subsample of the dataset and use the gradient of that as an estimate of the true gradient
- stochastic methods work well far away from an optima (they get close to an optima quickly) but they have trouble converging (full batch gradient descent has the opposite short comings)
- minibatch gradient is an unbiased and consistent estimate of the gradient
- can get better performance in minibatch gradient descent with a diminishing step size
- stochastic gradient descent is just gradient descent with a batch size of 1, it is super common
- witch works best depends on the problem and how much error you are ok with in your estimator

regression loss functions

- in a regression problem we are predicting a continuous output output value y which for the sake of this we can assume to be a scaler
- the residual $r = w^t x_i - y = \hat{y} - y$ ie the difference between our prediction and the observed outcome
- a loss function is distance based if
  1. it only depends on the residual ie $\ell(r)$
  2. it is zero when the residual is zero ie $r = 0 \iff \ell(r) = 0$

- distance based loss functions are translation invariant
- square or ℓ_2 loss is fiven by $\ell(r) = r^2$
- absolute loss is given by $\ell(r) = |r|$

y	$\hat{y}$	$\lvert r\rvert = \lvert y - \hat{y}\rvert$	$r^2 = (y - \hat{y})^2$
1	0	1	1
5	0	5	25
10	0	10	100
50	0	50	2500

-
- note that the absolute loss is translation invariant that is it weights errors the same regardless of where they are made
- the square loss is weights errors by how far they are away from being correct
- so we call the absolute loss robust to outliers since a learned parameter will shift more due to an outlier in l2 loss than l1 loss
- note an issue with absolute loss is that it is not differentiable

classification loss functions

- in classification we are predicting one from a set of discrete quantities for now we can just consider binary classification
- we have the score function $f(x) = w^t x$ which represents how much confidence we have in our classification prediction
- the margin is given by $m = y\hat{y} = yf(x)$ if $y \in \{0, 1\}$
- the margin is a measure of how correct we are so we want to maximize the margin of our estimator
- zero one loss is given by

$$\ell(f(x_i), y) = \mathbb{I}(f(x) \neq y)$$

note that this problem is non convex and non differentiable

- hinge loss is given by

$$\ell_{hinge} = max(1 - m, 0)$$

so the loss is linear if our margin is less than one (ie we made the right classification) and zero otherwise

- logistic loss is given by

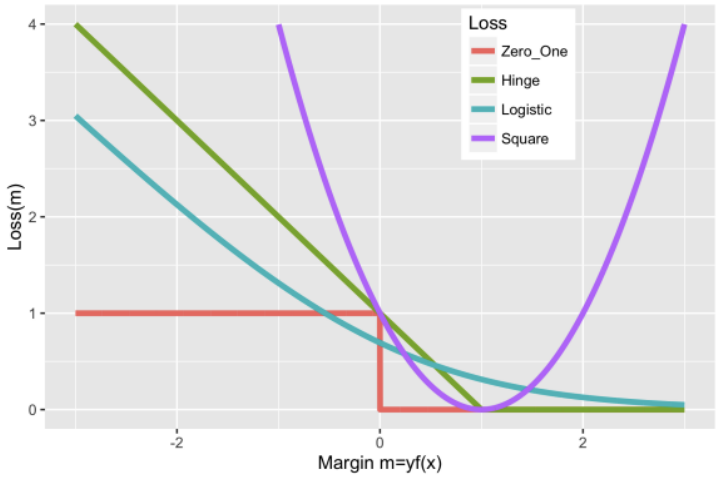
$$\ell_{logistic} = log(1 + e^{-m})$$

itt exponentially decays after 0

- classification square loss is given by

$$\ell(f(x), y) = (1 - m)^2$$

this heavily punishes misclassified examples



-