Lecture #32: Stochastic Gradient Descent & Logistic Regression

Today: what can you do if the function you want to minimize is too big?

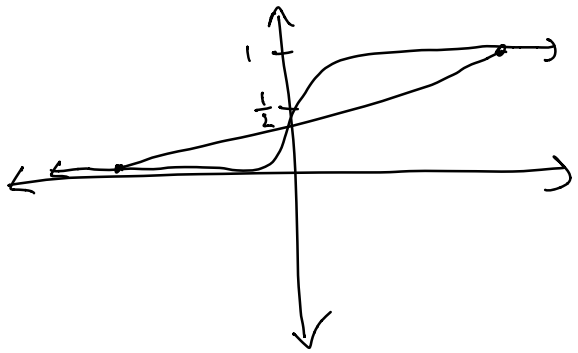
e.g. if it depends on tons of examples, too many to fit in memory

First, let's talk about logistic regression

def: The logistic function is

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

Graphically:



Facts: (1) For any z, $0 < \sigma(z) < 1$

(2) $\lim_{z \to -\infty} \sigma(z) = 0$

(3) $\lim_{z \to +\infty} \sigma(z) = 1$

Q: Why would we need such a function?

When you are predicting a probability

e.g. probability of having a heart attack over the next year?

Features:

- age
- gender
- weight
- HDL
- LDL
- ⋮

Let's encode these as a vector x

Logistic Regression: Find a weight vector w so that $\sigma(w^T x)$ is a good predictor (using labelled data)

Q2: Is the logistic function convex?

No, but when we combine it with the right way to measure loss, it will be

Let's build up to it slowly

Q3: If we have a data point (x, y) ($y \in \pm 1$), what is the probability the model assigns to the true outcome?

Case #1: $y = +1$

Given the weight vector w, the model thinks the example x has probability

$$\frac{1}{1 + e^{-w^T x}}$$

of being assigned a label $+1$

Case #2: $y = -1$

$$\frac{e^{-w^T x}}{1 + e^{-w^T x}} = \frac{1}{1 + e^{w^T x}}$$

Or to put it more compactly

$$\frac{1}{1 + e^{-y w^T x}}$$

Q4: If we have a collection of data

$$(x_1, y_1), (x_2, y_2) \ldots (x_N, y_N)$$

what is the probability of all these outcomes?

It is:

$$L(w) \triangleq \prod_{i=1}^{N} \frac{1}{1 + e^{-y_i w^T x_i}}$$

"likelihood"

Claim: Maximizing $L(w)$ is the same as minimizing $-\frac{\ln L(w)}{N} \triangleq f(w)$

$$f(w) = \frac{-\ln L(w)}{N} = \frac{-1}{N}\sum_{i=1}^{N} \ln\left(1 + e^{-y_i w^T x_i}\right)^{-1}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \ln\left(1 + e^{-y_i w^T x_i}\right)$$

Key Fact: $\ln\left(1 + e^{-y_i w^T x_i}\right)$ is a convex function of the weights w

Can check this using rules from Lecture #28

Let's take a step back:

Key Takeaway: For logistic regression, there isn't a closed form expression for optimal solution

In contrast, for least squares there is

This is why convex optimization is a powerful perspective — can solve richer problems

Q5: What are we still missing in order to get an algorithm for logistic regression?

Compute the gradient

Fact: $\nabla_w \ln\left(1 + e^{-y w^T x}\right) = \frac{-y x}{1 + e^{y w^T x}}$

Let's check this on a coordinate-by-coordinate basis: $y w^T x = y w_1 x_1 + y w_2 x_2 + \ldots + y w_i x_i + \ldots$

$$\frac{\partial}{\partial w_i} \ln\left(1 + e^{-y w^T x}\right) = \frac{-y x_i e^{-y w^T x}}{1 + e^{-y w^T x}}$$

$$= \frac{-y x_i}{1 + e^{y w^T x_i}}$$

Thus we get our algorithm

Gradient Descent (for Logistic Regression)

Initialize $w_0 = 0$

For $k = 1$ to T

update $w_k = w_{k-1} + \frac{\gamma}{N} \sum_{i=1}^{N} \frac{y_i x_i}{1 + e^{y_i w_{k-1}^T x_i}}$

End

Problem: At every step we have to look at all our data

What if it doesn't fit in memory?

This is very common in large-scale problems

Solution: Consider the error on just one (random) data point at time

Stochastic Gradient Descent

Initialize $w_0 = 0$

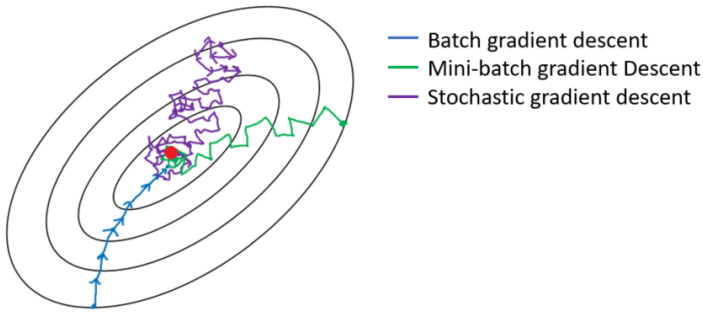
For $k = 1$ to T

Choose i at random

Update $w_k = w_{k-1} + \frac{\gamma y_i x_i}{1 + e^{y_i w_{k-1}^T x_i}}$

End

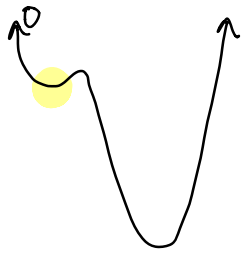
Q6: Which converges faster — GD or SGD — for convex optimization problems?



Intuition: SGD is still moving in right direction on average, just more noise in each step

For nonconvex optimization, SGD can even be better!

e.g.



Often only takes a few passes over your data to converge

Aside: what if you are working with sensitive data, that must be kept private?

Worry: After you solve logistic regression you release your model — i.e. w — but can someone figure out information about a patient from it?

Maybe! More complex the model the more things like this seem to happen

Solution: Add more noise

$$w_k = w_{k-1} + \frac{\gamma y_i x_i}{1 + e^{y_i w_{k-1}^T x_i}} + z$$

(z: Gaussian noise)

Can show that no data point biases your algorithm too much (for large noise)

I.e. it is impossible to learn too much about the private data you used to train the model

But it comes at a cost — worse performance

These are serious issues/tradeoffs:

> Homer et al. "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays."
>
> i.e. if you know an individual's DNA, can figure out if they participated in a genetic study even if just aggregate (pooled) data is released

Same is true for models you learn from solving optimization problems!