Optimization

Most of the materials are taken from here

Nastaran Okati

Outline

- Introduction
- Ransom Search
- Random Local Search
- Following the Gradient
- Gradient Descent

1.
Introduction

Optimization

- Optimization is the process of finding the set of parameters W that minimize the loss function

- Loss functions are usually defined over very high-dimensional spaces (e.g. in CIFAR-10 a linear classifier weight matrix is of size [10 x 3073] for a total of 30,730 parameters

Piecewise-linear structure of the Loss function

- Zero thresholded linear functions of W
- Each row has a positive (wrong class) or negative (correct class) sign

$$L_i = \sum_{j \neq y_i} \left[\max(0, w_j^T x_i - w_{y_i}^T x_i + 1) \right]$$

Example

Consider a simple dataset that contains three 1-dimensional points and three classes. The full SVM loss (without regularization) becomes:

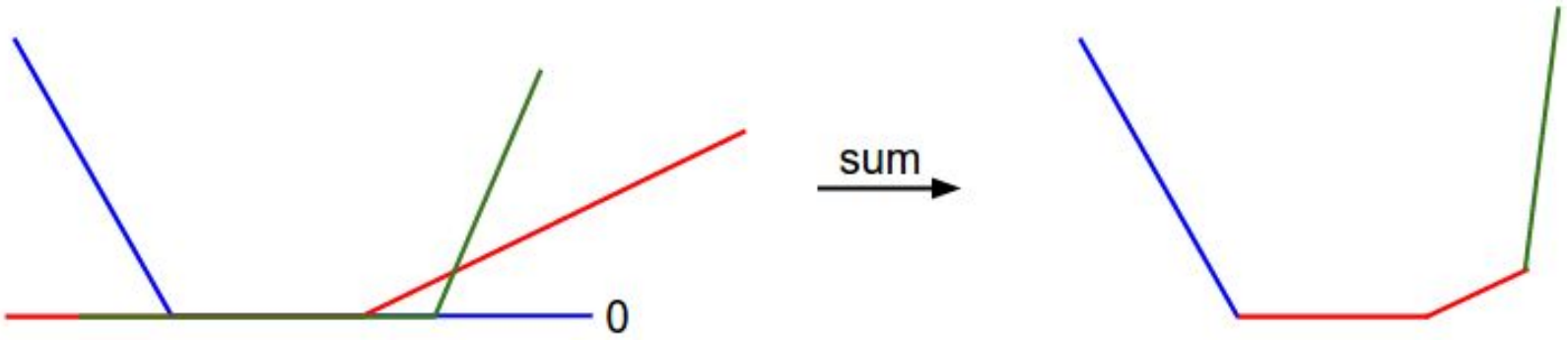
$$L_0 = \max(0, w_1^T x_0 - w_0^T x_0 + 1) + \max(0, w_2^T x_0 - w_0^T x_0 + 1)$$

$$L_1 = \max(0, w_0^T x_1 - w_1^T x_1 + 1) + \max(0, w_2^T x_1 - w_1^T x_1 + 1)$$

$$L_2 = \max(0, w_0^T x_2 - w_2^T x_2 + 1) + \max(0, w_1^T x_2 - w_2^T x_2 + 1)$$

$$L = (L_0 + L_1 + L_2)/3$$

Since these examples are 1-dimensional, the data x_i and weights w_j are numbers. Looking at, for instance, w_0, some terms above are linear functions of w_0 and each is clamped at zero.



- SVM cost function is an example of a convex function and there is a large amount of literature devoted to efficiently minimizing these types of functions

- Our goal is to eventually optimize Neural Networks where we can't easily use any of the tools developed in the Convex Optimization literature.

Random Search

The first (very bad) idea that may come to mind is to simply try out many different random weights and keep track of what works best.

Implement the random search algorithm!

Random Local Search

- Start with random weights and iteratively refine them over time to get lower loss
- Think of yourself as hiking on a hilly terrain with a blindfold on, and trying to reach the bottom.
- In the Cifar-10 example, what is the dimensionality of the hill?

Random Local Search

- Try to extend one foot in a random direction and then take a step only if it leads downhill.

- Start with a random W, generate random perturbations δW to it and if the loss at the perturbed W+δW is lower perform an update.

- Implement the Random Local Search!

Following the Gradient

- There is no need to randomly search for a good direction: we can compute the best direction along which we should change our weight
- Use the gradient of the loss function
- Feel the slope of the hill below our feet and stepping down the direction that feels steepest.

- The gradient is a generalization of slope for functions that don't take a single number but a vector of numbers.

- Gradient is just a vector of slopes (derivatives)

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Computing the Gradient

- Numeric Gradient
- Analytic Gradient

Numeric Gradient

- Use the gradient formula to compute it numerically.

- make a small change along one dimension and calculates the partial derivative of the loss function along that dimension

Implement the Numeric Gradient!

Attention!

In the mathematical formulation the gradient is defined in the limit as h goes towards zero, but in practice it is often sufficient to use a very small value

Remember to update in negative gradient direction. Why?

Step size (learning rate)

- The gradient tells us the direction in which the function has the steepest rate of increase, but it does not tell us how far along this direction we should step.

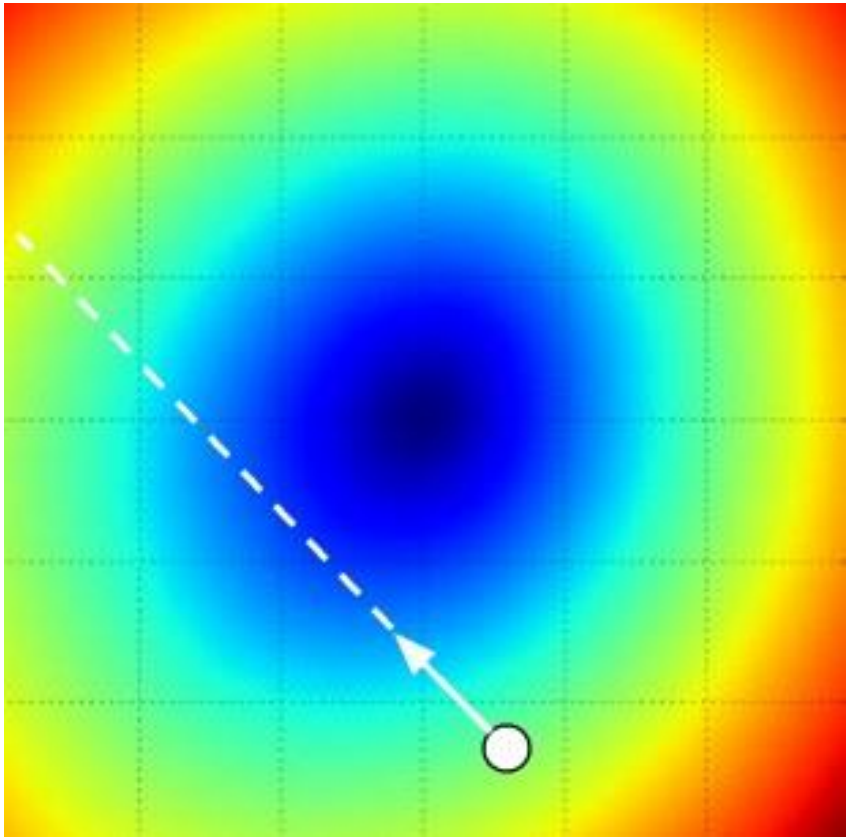
- One of the most important hyperparameter settings in training a neural network

We feel the hill below our feet sloping in some direction, but the step length we should take is uncertain.

What are the effects of choosing a small or big step size?

What is overstepping?

The effect of step size



Computing the gradient analytically with Calculus

- The numerical gradient is very simple to compute but it is an approximate and computationally expensive to compute

- Using calculus we can derive a direct formula for the gradient which is fast to compute but error-prone

- in practice it is very common to compute the analytic gradient and compare it to the numerical gradient to check the correctness of your implementation. (Gradient Check)

Example

Consider the SVM loss function

$$L_i = \sum_{j \neq y_i} \left[\max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta) \right]$$

Taking the gradient with respect to w_{y_i}

$$\nabla_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) \right) x_i$$

How to implement?

- in code you'd simply count the number of classes that didn't meet the desired margin (and hence contributed to the loss function) and then the data vector x_i scaled by this number is the gradient.

Taking the gradient with respect to other rows of W that $j \neq y_i$

$$\nabla_{w_j} L_i = \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) x_i$$

Gradient Descent

- The procedure of repeatedly evaluating the gradient and then performing a parameter update is called Gradient Descent

- Gradient Descent is currently by far the most common and established way of optimizing Neural Network loss functions.

Mini-batch gradient descent

- The training data can have millions of examples
- Wasteful to compute the full loss function over the entire training set in order to perform only a single parameter update
- Compute the gradient over batches of the training data

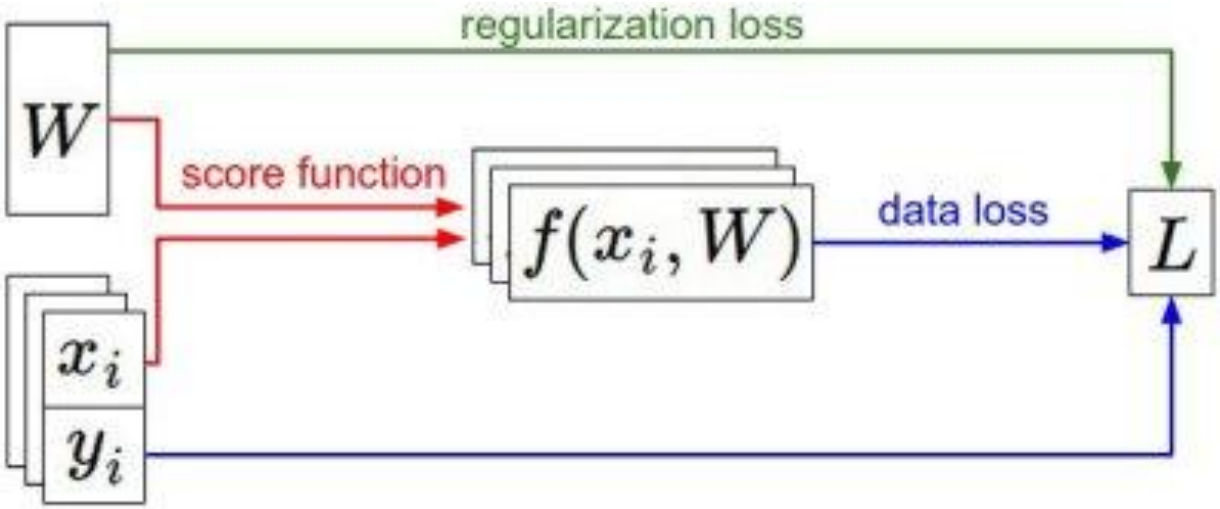
Why is it guaranteed to work?

- The examples in the training data are correlated
- the gradient from a mini-batch is a good approximation of the gradient of the full objective
- much faster convergence

Stochastic Gradient Descent

- The mini-batch contains only a single example
- Less common
- people use the term SGD even when referring to mini-batch gradient descent
- The size of the mini-batch is a hyperparameter but it is not very common to cross-validate it.

Summary



How to optimize an arbitrary loss function using calculus?

Wait until next session! 

Thanks!

Any questions?