Lecture 10

Math Foundations Team

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Introduction

- We will look at nonlinear optimization concepts in this lecture.
- We already know how to compute gradient, but there are some minutiae of gradient descent that we need to address.
- Machine learning algorithms depend heavily on the correctness of the gradient since if the gradient is computed erroneously, the algorihms might fail to find the local or global optimum.
- We will also look into some challenges in non-linear optimization.

Initialisation

- Gradient descent must start at some value of the parameters. How is an initial point chosen?
- For many applications, the vector components of the initial point can be chosen from $[-1, +1]$.
- In case the parameters are constrained to be non-negative, the components can be chosen in the range $[0, 1]$.
- What about more complex applications where the individual components are correlated with each other? In this case the choice of the initial point can be critical.
- It may also happen that choosing improper magnitudes of some components may cause overflow or underflow errors during updates.

Properties of optimization in machine learning

- Most objective functions in machine learning penalize the deviation of a predicted value from an observed value in one form or the other.
- A common loss function is $J(\boldsymbol{w}) = \sum_{i=1}^{i=n} \|\boldsymbol{w}^T \boldsymbol{X}_i - y_i\|^2$. Here $\boldsymbol{X}_i^T$ is the ith row vector in a matrix consisting of n row vectors where each row vector represents a training point, and y_i contains the real-valued observation of the ith training point.
- The above loss function occurs in least squared regression, and represents the sum of squared differences between the observed values y_i in the data and the predicted values $\hat{y}_i = \boldsymbol{w}^T \boldsymbol{X}_i$.

Log-likelihood objective function

- Another form of penalization is the log-likelihood objective function.
- This form of objective function uses the probability that the model's prediction of a dependent variable matches the observed value in the data.
- Higher values of these probabilities are desirable and the model should learn parameters to maximize these probabilities.
- The model might output the probability of a particular class in a binary classification setting, and a good goal for the model would be to maximize the probability of predicting the correct class.

Log-likelihood objective function

- Consider the pair $(\boldsymbol{X}_i, y_i$ where $\boldsymbol{X}_i$ is the ith training point and y_i is the observed class. Let the probability of predicting class y_i for the training point $\boldsymbol{X}_i^T$ given model parameters $\boldsymbol{w}$ be $P(\boldsymbol{X}_i, y_i, \boldsymbol{w})$.
- The probability of correct prediction over all training pairs $\boldsymbol{X}_i, y_i$ is the product of probabilities $P(\boldsymbol{X}_i, y_i, \boldsymbol{w})$ over all $\boldsymbol{X}_i, y_i$.
- The goal of the machine learning algorithm is to find model parameters $\boldsymbol{w}$ that maximizes this product of probabilities.
- We convert this maximization problem into a minimization one by taking the negative logarithm of the product of all the probabilities.

Log-likelihood objective function

- One benefit of working with the logarithm of the product of probabilities rather than the product itself is that we can avoid underflow issues owing to the product of a large number of numbers that lie between 0 and 1.

$$
\begin{aligned}
J(\boldsymbol{w}) &= -\log_e \prod_{i=1}^{i=n} P(\boldsymbol{X}_i, y_i, \boldsymbol{w}) \\
&= -\sum_{i=1}^{i=n} \log_e P(\boldsymbol{X}_i, y_i, \boldsymbol{w})
\end{aligned}
$$

- The objective function is now an additionally separable sum over the training examples.

Additionally separable sum

- We can write the total objective function $J(\boldsymbol{w}) = \sum_{i=}^{n} J_i(\boldsymbol{w})$.
- This type of linear separability is useful since it enables the use of techniques like stochastic gradient descent and mini-batch stochastic gradient descent.
- The idea here is that we can replace the gradient of the entire objective function with a sampled approximation.
- The advantage of using stochastic gradient descent is that optimization can be speeded up.

Optimization in machine learning

- There are some subtle differences between traditional optimization and optimization in the machine learning context.
- In traditional optimization, we focus on updating parameters so that the objective function is minimized as much as possible.
- In machine learning, minimization of the objective function is performed over training data but the model is applied on training data which is unseen.
- Optimizing the objective function too much on the training data may cause the model to perform poorly on the test data!
- We need to avoid the problem of overfitting the training data.

Overfitting

- Suppose we have 4-dimensional data and one dependent variable, i,e output that is a function of the four inputs. Let the input parameters be x_1, x_2, x_3, x_4 and the output be y.
- We seek to learn parameters w_1, w_2, w_3, w_4, w_5 such that out prediction $\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x4 + w_5$ where y is the observed value for the given values of the inputs.
- We would like to minimize the squared error between prediction and actual output over all the given training examples.

Overfitting

- Assuming that we have fewer than 5 training examples, we can actually find values of w_1, w_2, w_3, w_4, w_5 to get the squared error to be zero since we can solve for values of w_1, w_2, w_3, w_4, w_5 by solving fewer than five linear equations. Such a solution is possible because we have fewer equations than variables.
- Our model parameters will perform very well on the training data but might perform poorly on the test data.

Example of overfitting

Consider the following data on 4 variables x_1, x_2, x_3, x_4 and associated output variable y. Let us say that this is a sample of real-life data where the output $y \approx x_1$.

x_1	x_2	x_3	x_4	y
61	2	3	0.1	59
40	0	4	0.5	40
68	0	10	1.0	70

Minimizing squared error, we notice that one good solution is $w_1 = 1, w_2 = w_3 = w_4 = 0, w_5 = 0$. This solution does not give zero squared error with respect to the actual observations but gives an error close to zero.

Example of overfitting

- Consider the solution $w_1 = 0, w_2 = 7, w_3 = 5, w_4 = 0, w_5 = 20$. This solution gives zero training error. It is a very poor solution since there is no dependence of the output variable y on x_1 while we know that there is actually a strong dependence between y and x_1. Therefore it will incur a high error on test-data.
- On the other hand, the previous solution $w_1 = 1, w_2 = w_3 = w_4 = 0, w_5 = 0$ is a very good one since it captures the real-life relationship between the output variable y and x_1. This example illustrates the idea that minimizing the loss function to the greatest extent may not be a good thing since the model may then perform poorly on real-life data.

Tuning hyperparameters

- The term "hyperparameter" is used to refer to parameters used to regulate the design of the model like learning rate. The learning rate is not to be confused with actual parameters like the weights of a linear regression model.
- Machine learning algorithms operate in a two-tiered way - first the values of hyperparameters are fixed, and these values are then used in the model to learn the actual model parameters.
- The hyperparameters should not be tuned using the same data set used for gradient descent in order to avoid overfitting the training data.

Tuning hyperparameters

- A portion of the training data is held out as validation data, and model performance is recorded for various choices of the hyperparameters on the validation data.
- The main challenge in hyperparameter tuning is that different combinations of parameters need to be tested for their performance.
- We can perform a grid-search to test all possible combinations of values for the hyperparameters, but the number of grid points to be considered will grow exponentially in the number of hyperparameters.
- One way to get around this is to use coarse grids initially to narrow down the range of the hyperparameters and then use a finer grid.

Feature processing

- The loss function can have vastly different sensitivities to different model parameters and this can have an impact in the learning process.
- Consider a model where a person's wealth y is modeled in terms of his age x_1 and number of years of college education x_2.
- The age variable has a range of 100, ie $[0, 100]$ and the number of years in the college education is in the range $[0, 10]$.
- The formula for wealth y is $y = w_1 x_1^2 + w_2 x_2^2$.
- We have $\frac{\partial y}{\partial w_1} = x_1^2$ and $\frac{\partial y}{\partial w_2} = x_2^2$. Since x_1 and x_2 are generally very different in magnitude, we take small steps in respect of w_2 and large steps in respect of w_1.

Feature processing

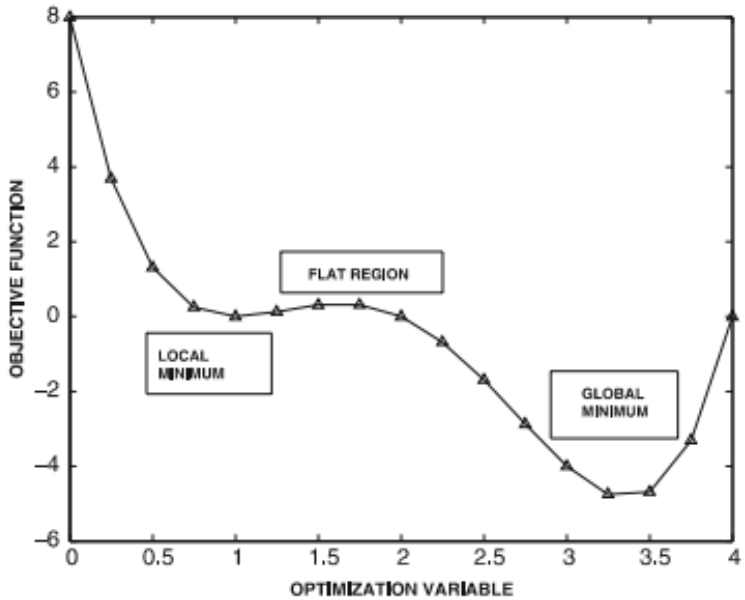
- Taking small steps in w_2 and large steps in w_1 will make us go steadily towards the optimal value for w_2 but oscillate with respect to the optimal value of w_1, overshooting the target each time. This makes convergence very slow.
- It is therefore helpful to have features with similar variance.
- Two techniques used for achieving similar variance are mean-centering and feature normalization.
- In case of mean-centering a vector of column-wise means is subtracted from each data point.
- In case of feature normalization, each feature value is divided by its standard deviation.
- In case of min-max normalization we scale the jth feature of the ith datapoint as follows: $x_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j}$.

Local optima and flat regions

- Consider the following one-dimensional function: $F(x) = (x-1)^2((x-3)^2 - 1)$.
- Taking the derivative and setting it to zero, we have $F'(x) = 2(x-1)((x-1)(x-3) + (x-3)^2 - 1) = 0$.
- The solutions to this equation are $x = 1, x = \frac{5-\sqrt{3}}{2} = 1.634, x = \frac{5+\sqrt{3}}{2} = 3.366$.
- We can show that the first and third roots are minima since $F''(x) > 0$ at these points while the second point is a maximum since $F''(x) < 0$.
- Evaluating the function values at these points, we find $F(1) = 0$, $F(\frac{5-\sqrt{3}}{2}) = 0.348, F(\frac{5+\sqrt{3}}{2}) = -4.848$.

Local optima and flat regions

If we start gradient descent from any point less than 1.634, we will arrive only at a local minimum.



(a) Local optima with flat regions

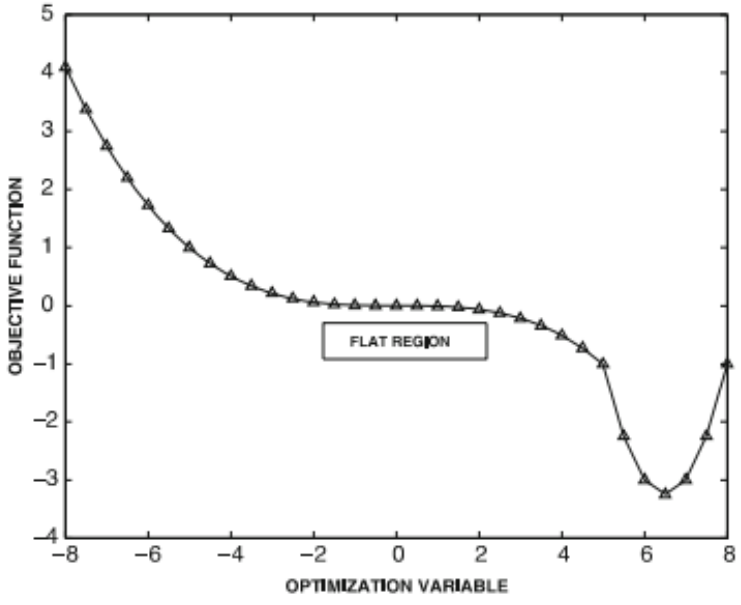
Local optima and flat regions

- We might never arrive at a global minimum if we keep choosing wrong starting points. Thus we will get stuck at a local minimum, not knowing that a better solution exists.
- The problem becomes worse with high-dimensionality.
- Consider an objective function consisting of the sum of d-univariate functions in the variables $x_1, x_2, \ldots, x_d$, each a function of a different variable, say $F(x_1, x_2, \ldots x_d) = A_1(x_1) + A_2(x_2) + \ldots + A_n(x_n)$.
- Let $A_i(x_i)$ have k_i local minima. Setting $\frac{\partial F}{\partial x_i} = 0 \forall i$, we note that any point $(x_1^*, x_2^*, \ldots, x_d^*)$, where x_i^* is a local minima of the function $A_i(x_i)$, is a solution to $\frac{\partial F}{\partial x_i} = 0$.
- This is because $\frac{\partial F}{\partial x_i}|_{(x_1^*, x_2^*, \ldots x_d^*)} = A_i'(x_i^*) = 0$ since x_i^* is a local minimum of $A_i(x_i)$.

Local optima and flat regions

- There are therefore $\prod_{i=1}^{i=d} k_i$ local minima for the function $F(x_1, x_2, \ldots, x_d)$, which is very large number of points. Gradient descent could be stuck at any one of these points which might be far from the global optimum.
- Another problem to contend with is the presence of flat regions where the gradient is close to zero. An example of this situation is shown in the next slide.
- Flat regions are problematic because the speed of descent depends on the magnitude of the gradient, given a fixed learning rate. The optimization process will take a long time to cross a flat region of space which will make convergence slow.

Local optima and flat regions



(b) Only global optimum with flat region

Differential curvature

- In multi-dimensional settings, the components of the gradient with respect to different parameters can vary widely. As has already been mentioned in a previous slide, this will cause convergence problems since there is oscillation in the update step with respect to some components and a steady movement with respect to other components.
- Consider the simplest possible case of a bowl-like convex, quadratic objective function with a single global minimum - $L = x^2 + y^2$ represents a perfectly circular bowl, and the function $L = x^2 + 4y^2$.
- We shall show contour plots of both functions and how gradient descent performs on finding the minimum of the two functions.

Differential curvature

- What is the qualitative difference between $L = x^2 + y^2$ and $L = x^2 + 4y^2$. Intuitively one looks symmetric in x and y, and the other is not.
- The second loss function is more sensitive to changes in y as compared to x - it looks like an elliptical bowl. The specific sensitivity depends on the position of x, y.
- Looking at the second-order derivatives we can see that for the second function $\frac{\partial^2 L}{\partial^2 x^2}$, and $\frac{\partial^2 L}{\partial^2 y^2}$ are very different.

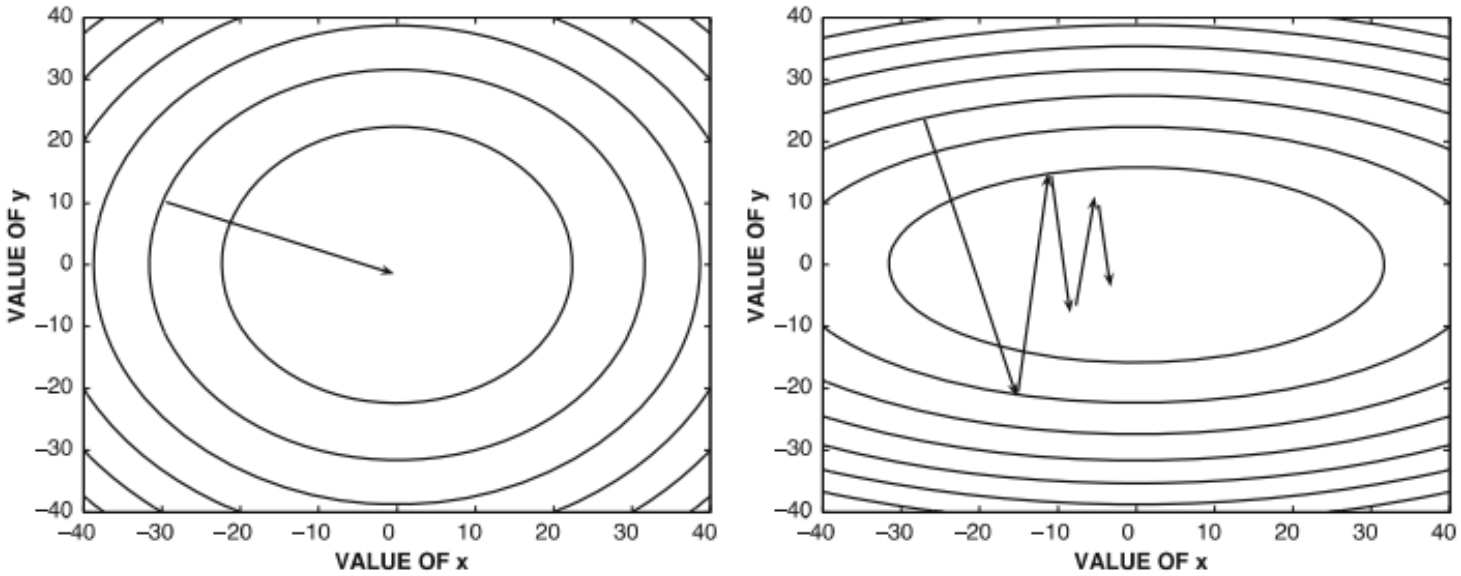
Differential curvature

- The second-prder derivative measures the rate of change of the gradient - a high second-order derivative means high curvature.
- From the point of view of gradient descent we want moderate curvature in all dimensions as it would means that the gradient does not change too much in some dimensions compared to others. We can then make gradient-descent steps of large sizes.

Differential curvature

- In the next slide we show contour plots of the perfect and elliptical bowls discussed previously. We see that in case of the perfect bowl, a sufficiently large step-size from any point can take us directly to the optimum of the function in one-step, since the gradient at any point points towards the optimum of the function. This is not true for the elliptical bowl, the gradient at any point does not point to the optimum of the function.
- Note that the gradient at any point is orthogonal to the contour line at that point. This because the dot product of the gradient ∇F and a small displacement $\delta \boldsymbol{x}$ along the contour line gives the change in the value of the function along the displacement $\boldsymbol{x}$. Since the function remains constant along the contour line, $\nabla F.\boldsymbol{x} = 0$

Contour plots



(a) Loss function is circular bowl
$L = x^2 + y^2$

(b) Loss function is elliptical bowl
$L = x^2 + 4y^2$

Figure 5.2: The effect of the shape of the loss function on steepest-gradient descent

Differential curvature

- A closer look at the controur plot for the elliptical bowl case shows that in the y-direction, we see oscillatory movement as in each step we correct the mistake of overshooting made in the previous step. The gradient component along the y-direction is more than the component along the x-direction.
- Along the x-direction, we make small movements towards the optimum x-value.
- Overall, after many training steps we find that we have made little progress to the optimum.
- It needs to be kept in mind that the path of steepest descent in most objective functions is only an instantaneous direction of best improvement, and is not the correct direction of descent in the longer term.

Revisiting feature normalization

We show how to address in some measure the differential curvature problem by feature normalization. Consider the following toy dataset, where the two input attributes are Guns and Butter respectively, and the output attribute is Happiness.

Guns (number per capita)	Butter (ounces per capita)	Happiness (index)
0.1	25	7
0.8	10	1
0.4	10	4

We intend to find a relationship of the form $y = w_1 x_1 + w_2 x_2$ from the data. The coefficients w_1 and w_2 are found using gradient descent on the loss function computed from the above data.

Revisiting feature normalization

- From the given three examples we can set up the loss function as follows:

$$J(\boldsymbol{w}) = (0.1w_1+25w_2-7)^2+(0.8w_1+10w_2-1)^2+(0.4w_1+10w_2-4)^2$$

.

- We note that this objective function is much more sensitive to w_2 than w_1 since the coefficients for w_2 in the expression above are much larger than those for w_1.
- One way to get around this issue is to standardize each column to zero mean and unit variance, the coefficients for w_1 and w_2 will become much more similar, and differential curvature will be reduced.