DA 507 - Modeling and Optimization

Nonlinear Optimization: Gradient Descent

Birol Yüceoğlu

Migros T.A.Ş.

December 18, 2021

Agenda

- Optimization in data science
- Gradient descent
- Simple examples
- Linear regression

Motivation & Machine Learning

Why optimization?

- Machine learning (ML) is a new programming paradigm that answers the following question: "Can a computer go beyond what we order it to perform and learn on its own to perform a task?"

Why optimization?

- Machine learning (ML) is a new programming paradigm that answers the following question: "Can a computer go beyond what we order it to perform and learn on its own to perform a task?"
- A ML system is trained (not programmed) to learn from many examples relevant to the task. In other words we are given inputs and outputs and learn *rules* from data.
  - Translation: Text (lots of text) and labels (bilingual documents, EU, Canada)
  - Autonomous cars: Images and labels (Google Captcha)
  - Credit scoring: Customer past history and previous credit records

Learning

In the context of ML, learning means using appropriate representations of data to get closer to the expected output. To do ML, we need:

- Input data points

Learning

In the context of ML, learning means using appropriate representations of data to get closer to the expected output. To do ML, we need:

- Input data points
- Examples of the expected outputs

Learning

In the context of ML, learning means using appropriate representations of data to get closer to the expected output. To do ML, we need:

- Input data points
- Examples of the expected outputs
- A way to measure whether the algorithm is doing a good job. This measure how far the output of the algorithm is to the real output. This provides a feedback to improve the outputs of the algorithm.

Learning

In the context of ML, learning means using appropriate representations of data to get closer to the expected output. To do ML, we need:

- Input data points
- Examples of the expected outputs
- A way to measure whether the algorithm is doing a good job. This measure how far the output of the algorithm is to the real output. This provides a feedback to improve the outputs of the algorithm.
- Feedback & Improvement $\rightarrow$ OPTIMIZATION

Optimization

Optimization is used in ML to get the output(s) of a model close to the expected output. Three approaches rely heavily on optimization. These are **linear regression**, **logistic regression**, **neural networks**.
Optimizing the outputs of these algorithms requires three things:

- *Loss function* measures the performance of the model on the training data. We want to decrease the loss function to steer the approach in the right direction.

Optimization

Optimization is used in ML to get the output(s) of a model close to the expected output. Three approaches rely heavily on optimization. These are **linear regression**, **logistic regression**, **neural networks**. Optimizing the outputs of these algorithms requires three things:

- *Loss function* measures the performance of the model on the training data. We want to decrease the loss function to steer the approach in the right direction.
- *Optimizer* is the mechanism we use to decrease the loss function by updating the model parameters based on the data it sees, its loss function, and model parameters.

Optimization

Optimization is used in ML to get the output(s) of a model close to the expected output. Three approaches rely heavily on optimization. These are **linear regression**, **logistic regression**, **neural networks**. Optimizing the outputs of these algorithms requires three things:

- *Loss function* measures the performance of the model on the training data. We want to decrease the loss function to steer the approach in the right direction.
- *Optimizer* is the mechanism we use to decrease the loss function by updating the model parameters based on the data it sees, its loss function, and model parameters.
- *Metrics* are used to monitor the model during training and testing.

Loss Functions vs Metrics

Loss functions are used to steer the model parameters in the right direction with the help of an optimizer. We should be able to calculate the loss function of a single observation. Furthermore, we should know the derivative of a loss function to be able to use a gradient based optimizer.

Loss Functions vs Metrics

Loss functions are used to steer the model parameters in the right direction with the help of an optimizer. We should be able to calculate the loss function of a single observation. Furthermore, we should know the derivative of a loss function to be able to use a gradient based optimizer.

On the other hand, metrics show the performance of the model but do not have to be used in the optimization. The most common example is 'accuracy'. There is no gradient based optimizer that is guaranteed to increase the accuracy of a model. Note that the accuracy can be calculated when you can predict the output of all observations.

Gradient Descent Basics

Gradient-Based Optimization

Gradient-based optimization algorithms are iterative approaches that rely on the gradient to gradually adjust model parameters, based on a feedback signal. The feedback signal consists of your loss function and its derivative.

Gradient-Based Optimization

Gradient-based optimization algorithms are iterative approaches that rely on the gradient to gradually adjust model parameters, based on a feedback signal. The feedback signal consists of your loss function and its derivative.

The derivative tells you how much a small increase in your model parameters increases your loss function. Moving in the opposite direction to the derivative should help you decrease the value of your loss function.

Gradient-Based Optimization

Gradient-based optimization algorithms are iterative approaches that rely on the gradient to gradually adjust model parameters, based on a feedback signal. The feedback signal consists of your loss function and its derivative.

The derivative tells you how much a small increase in your model parameters increases your loss function. Moving in the opposite direction to the derivative should help you decrease the value of your loss function.

Gradient descent is the most basic variant of gradient-based optimization algorithms.

Gradient Descent

- A generic optimization algorithm.

Gradient Descent

- A generic optimization algorithm.
- Used for solving underlying optimization problems in linear/logistic regression and neural networks.

Gradient Descent

- A generic optimization algorithm.
- Used for solving underlying optimization problems in linear/logistic regression and neural networks.
- Uses first-order information (derivative).

Gradient Descent

- A generic optimization algorithm.
- Used for solving underlying optimization problems in linear/logistic regression and neural networks.
- Uses first-order information (derivative).
- Finds a local minimum in general.

Gradient Descent

- A generic optimization algorithm.
- Used for solving underlying optimization problems in linear/logistic regression and neural networks.
- Uses first-order information (derivative).
- Finds a local minimum in general.
- Local minimum is also global minimum in convex functions.

Gradient Descent

- A generic optimization algorithm.
- Used for solving underlying optimization problems in linear/logistic regression and neural networks.
- Uses first-order information (derivative).
- Finds a local minimum in general.
- Local minimum is also global minimum in convex functions.
- Gradient descent is used in linear / logistic regression and neural networks.

Gradient Descent

Gradient descent algorithms consists of three basic steps:

- Given your input variables calculate the output of the model.

Gradient Descent

Gradient descent algorithms consists of three basic steps:

- Given your input variables calculate the output of the model.
- Compute the loss of the model based on your predicted and the real output.

Gradient Descent

Gradient descent algorithms consists of three basic steps:

- Given your input variables calculate the output of the model.
- Compute the loss of the model based on your predicted and the real output.
- Update your model paramters in a way that slightly reduces the loss by moving in the opposite direction to the gradient.

Gradient Descent

Gradient descent algorithms consists of three basic steps:

- Given your input variables calculate the output of the model.
- Compute the loss of the model based on your predicted and the real output.
- Update your model paramters in a way that slightly reduces the loss by moving in the opposite direction to the gradient.
- Repeat these steps

Gradient Descent

Gradient descent algorithms consists of three basic steps:

- Given your input variables calculate the output of the model.
- Compute the loss of the model based on your predicted and the real output.
- Update your model paramters in a way that slightly reduces the loss by moving in the opposite direction to the gradient.
- Repeat these steps
- Stop when the derivative is close enough to 0
  An example:

$$\min f(x) = x^2$$

Gradient Descent

$$minimize_x \; f(x) = x^2$$

Gradient Descent Algorithm:

- Start from an initial value of x_0 (may be a random guess)

Gradient Descent

$$minimize_x \ f(x) = x^2$$

Gradient Descent Algorithm:

- Start from an initial value of x_0 (may be a random guess)
- Evaluate $\nabla f(x_t)$

Gradient Descent

$$minimize_x \ f(x) = x^2$$

Gradient Descent Algorithm:

- Start from an initial value of x_0 (may be a random guess)
- Evaluate $\nabla f(x_t)$
- Set $x_{t+1} = x_t - \lambda \nabla f(x_t)$ for some step size λ.

Gradient Descent

$$minimize_x \; f(x) = x^2$$

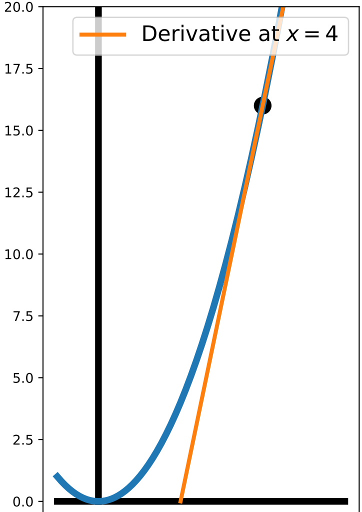
Gradient Descent Algorithm:

- Start from an initial value of x_0 (may be a random guess)
- Evaluate $\nabla f(x_t)$
- Set $x_{t+1} = x_t - \lambda \nabla f(x_t)$ for some step size λ.
- Stop when the number of iterations exceed the set limit or when $|\nabla f(x_t)| < \epsilon$ for some small ϵ.

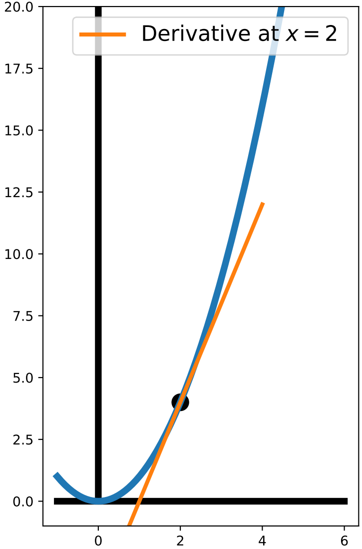
First Example $f(x) = x^2$

Gradient Descent

Start at $x_0 = 4$. Move in the opposite direction to the gradient.
$x_1 = x_0 - \lambda \nabla f(4) = 4 - \frac{1}{4}8 = 2$

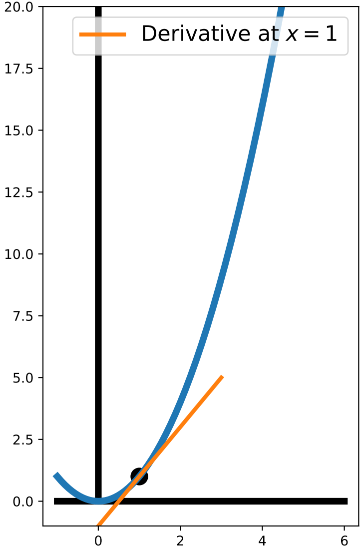


$x_2 = x_1 - \lambda \nabla f(2) = 2 - \frac{1}{4}4 = 1$



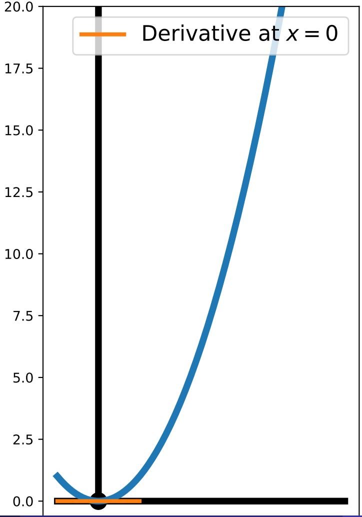
Gradient-Based Optimization

$$x_3 = x_2 - \lambda \nabla f(1) = 1 - \frac{1}{4} 2 = \frac{1}{2}$$



Gradient-Based Optimization

Eventually, we reach $x = 0$ or a point close enough to 0. At that point $\nabla f(0) = 0$ and we reach local (global) minimum.



Multiple Variables

Example (Two Variables)

$$\min f(x, y) = x^2 + 3y^2$$

Example (Two Variables)

$$\min f(x, y) = x^2 + 3y^2$$

What is the derivative in this case?

$$\frac{df}{dx} = \frac{\partial f}{\partial x} = \nabla_x f = 2x$$

$$\frac{df}{dx} = \frac{\partial f}{\partial y} = \nabla_y f = 6y$$

Example (Two Variables)

$$\min f(x, y) = x^2 + 3y^2$$

What is the derivative in this case?

$$\frac{df}{dx} = \frac{\partial f}{\partial x} = \nabla_x f = 2x$$

$$\frac{df}{dx} = \frac{\partial f}{\partial y} = \nabla_y f = 6y$$

We have to move in two directions

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \lambda \begin{bmatrix} \nabla_x f \\ \nabla_y f \end{bmatrix}$$

Visualizing Gradient Descent Algorithm

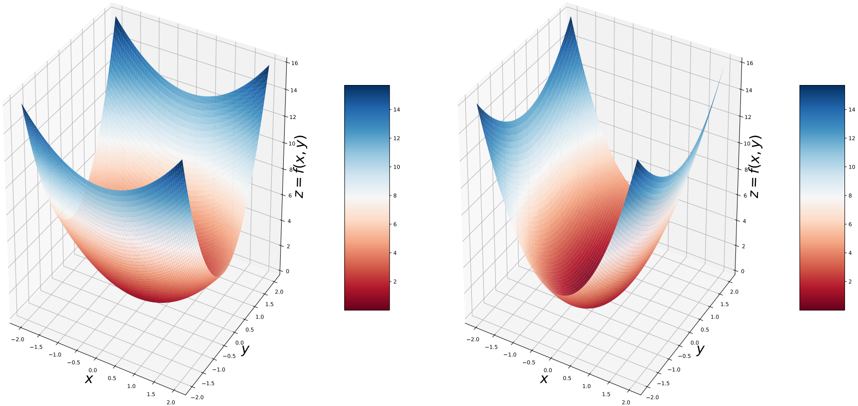


Figure: $f(x, y)$ from two different angles.

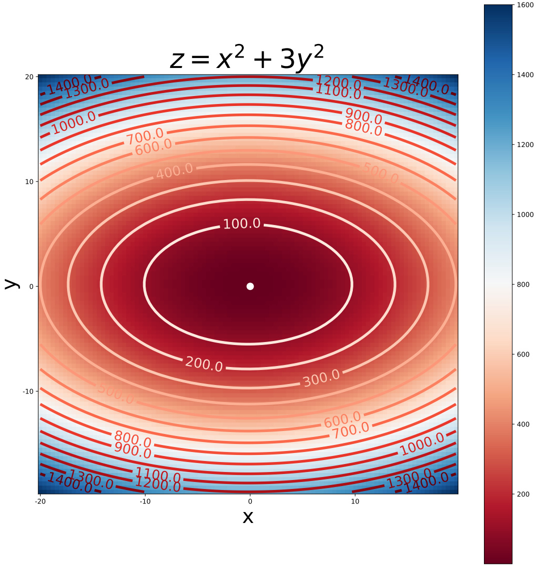


Figure: Bird's eye view.

$$\min f(x, y) = x^2 + 3y^2$$

$$\min f(x,y) = x^2 + 3y^2$$

Let us start at $x_0 = y_0 = 2$. Set $\lambda = \frac{1}{4}$. The derivative is $\begin{bmatrix} 2x \\ 6y \end{bmatrix}$.

$$\min f(x,y) = x^2 + 3y^2$$

Let us start at $x_0 = y_0 = 2$. Set $\lambda = \frac{1}{4}$. The derivative is $\begin{bmatrix} 2x \\ 6y \end{bmatrix}$. We have to move in two directions

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\min f(x,y) = x^2 + 3y^2$$

Let us start at $x_0 = y_0 = 2$. Set $\lambda = \frac{1}{4}$. The derivative is $\begin{bmatrix} 2x \\ 6y \end{bmatrix}$. We have to move in two directions

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 2 \\ -6 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\min f(x,y) = x^2 + 3y^2$$

Let us start at $x_0 = y_0 = 2$. Set $\lambda = \frac{1}{4}$. The derivative is $\begin{bmatrix} 2x \\ 6y \end{bmatrix}$. We have to move in two directions

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 2 \\ -6 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.25 \\ -0.25 \end{bmatrix}$$

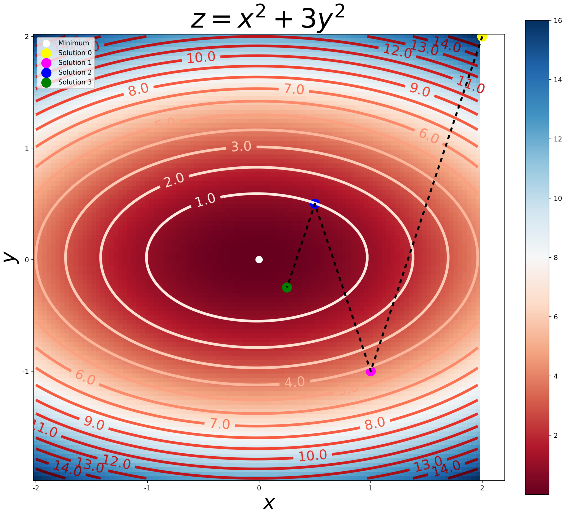


Figure: Illustrating gradient descent.

Back to Python

Linear Regression

- Linear regression is a supervised learning problem where the target variable is continuous.

Linear Regression

- Linear regression is a supervised learning problem where the target variable is continuous.
- It is based on the hypothesis that the input variables are linearly related with the target variable.

Linear Regression

- Linear regression is a supervised learning problem where the target variable is continuous.
- It is based on the hypothesis that the input variables are linearly related with the target variable.
- This relation is captured with the parameters of the model, which are learned using data.

Linear Regression

- Linear regression is a supervised learning problem where the target variable is continuous.
- It is based on the hypothesis that the input variables are linearly related with the target variable.
- This relation is captured with the parameters of the model, which are learned using data.
- The linear regression problem can be solved analytically. However, in practice it is solved using gradient descent.

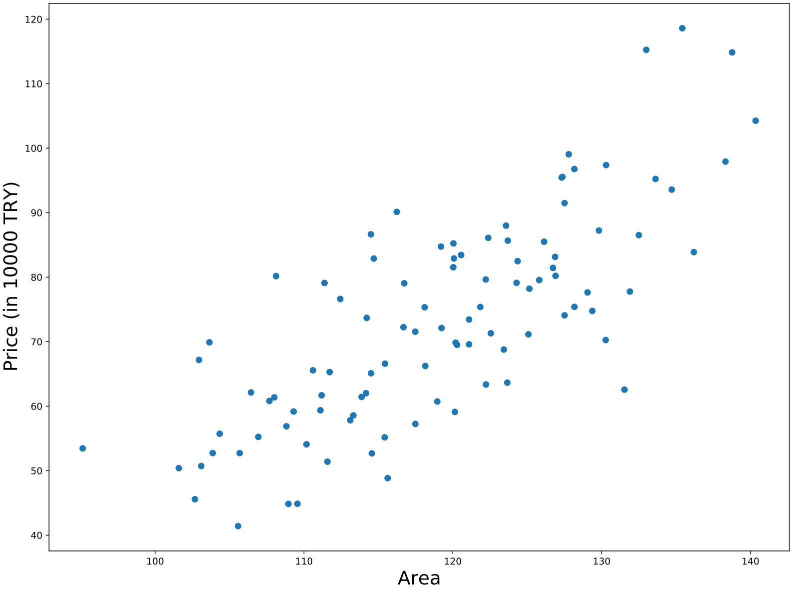


Figure: $Price = -80.21 + 1.29 * Area.$

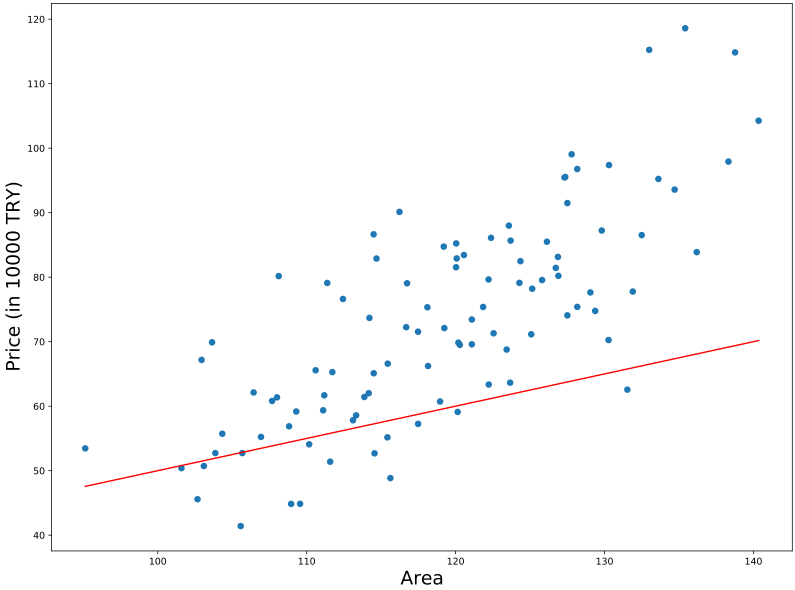


Figure: A fit without optimal parameters.

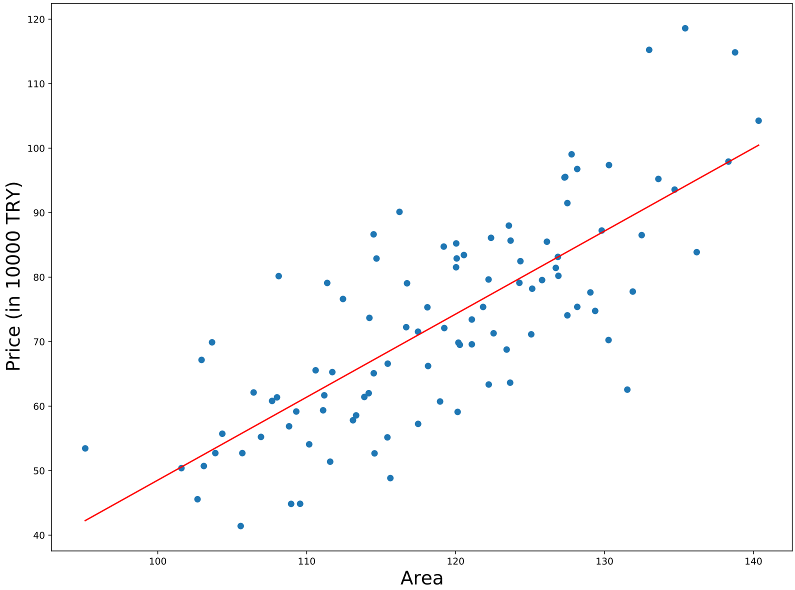


Figure: A fit using optimal parameters.

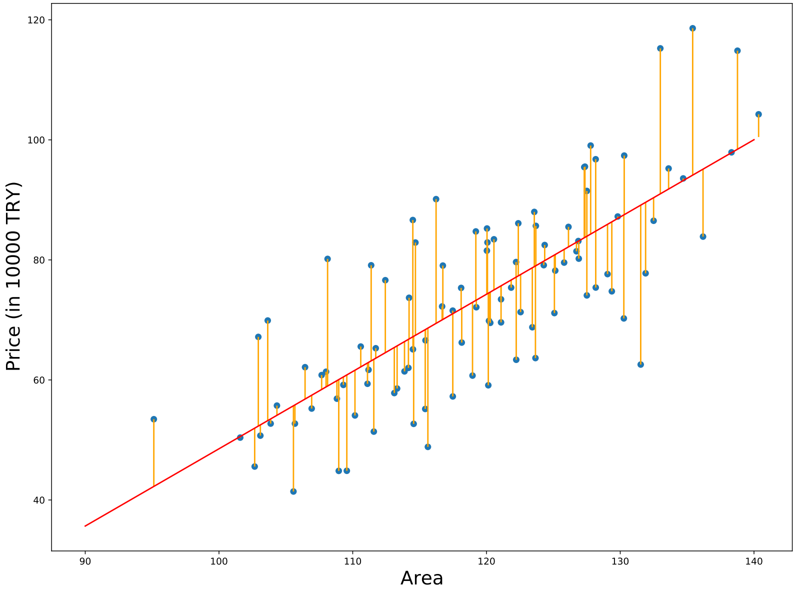


Figure: Visualizing error terms.

Notation

x and y to denote input variable and target variable. We are given m training examples, consisting of n features.

In linear regression, we approximate target variable y using the following equation

$$\hat{y} = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$$

Here θ's are the parameters of the model that we want to learn via optimization. For the sake of simplicity, we set $x_0 = 1$, so

$$\hat{y} = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

Notation

We want to choose parameters θ so that $\hat{y}$ is as close to y as possible for the training examples provided to us. The *loss function* that we use is the (root) mean squared error (RMSE or MSE). We want to minimize the MSE.

$$MSE(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

equivalently

$$MSE(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$$

Thus, the objective of regression is to find

$$\arg\min_{\theta} MSE(\theta)$$

.

Gradient Descent

We want to choose θ which minimizes $MSE(\theta)$. Even though the problem can be solved analytically, it involves matrix inversion, which is computationally costly. We will solve the linear regression problem using gradient descent.

- Start with some random $\theta^0 = [\theta_0, \theta_1]$

Gradient Descent

We want to choose θ which minimizes $MSE(\theta)$. Even though the problem can be solved analytically, it involves matrix inversion, which is computationally costly. We will solve the linear regression problem using gradient descent.

- Start with some random $\theta^0 = [\theta_0, \theta_1]$
- Set $\theta^{t+1} = \theta^t - \lambda \frac{\partial MSE(\theta)}{\partial \theta}$

When updating the parameters, we consider all training observations. There are variants of the gradient descent algorithm that updates the parameters using a subset of observations.

Gradient Descent

We want to choose θ which minimizes $MSE(\theta)$. Even though the problem can be solved analytically, it involves matrix inversion, which is computationally costly. We will solve the linear regression problem using gradient descent.

- Start with some random $\theta^0 = [\theta_0, \theta_1]$
- Set $\theta^{t+1} = \theta^t - \lambda \frac{\partial MSE(\theta)}{\partial \theta}$
- Stop when $|\frac{\partial MSE(\theta)}{\partial \theta}| < \epsilon$

When updating the parameters, we consider all training observations. There are variants of the gradient descent algorithm that updates the parameters using a subset of observations. The gradient descent algorithm finds a local optimum in general. However, the associated optimization problem for linear regression has only one local optimum, which is also global optimum. This is because the objective function (MSE) is convex.

Simple Example

We are given a data set consisting of house prices and their areas. We want to explain the price of the houses as a linear function of the area.

$$Price = \theta_0 + \theta_1 * Area$$

The underlying optimization problem is a problem with two variables, θ_0 and θ_1. In other words we want to find values of θ_0 and θ_1 that will yield the minimum MSE.

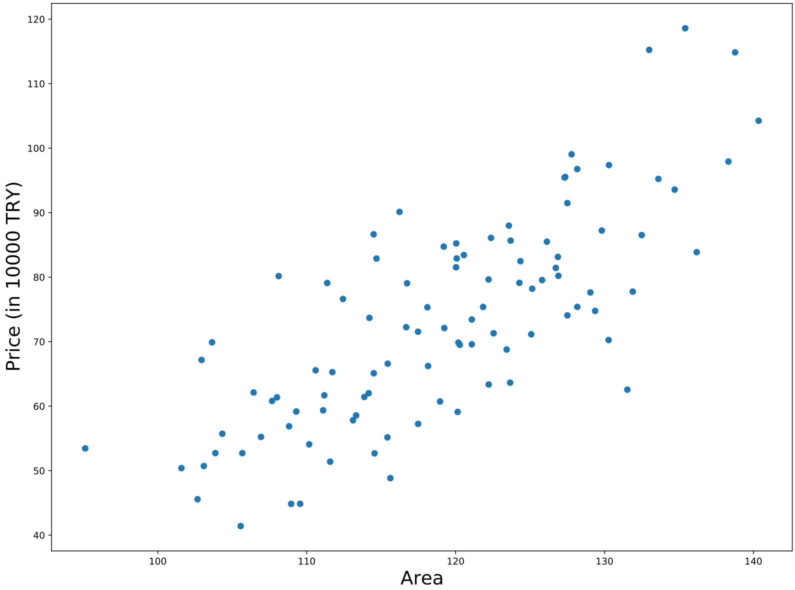


Figure: $Price = -80.21 + 1.29 * Area$.

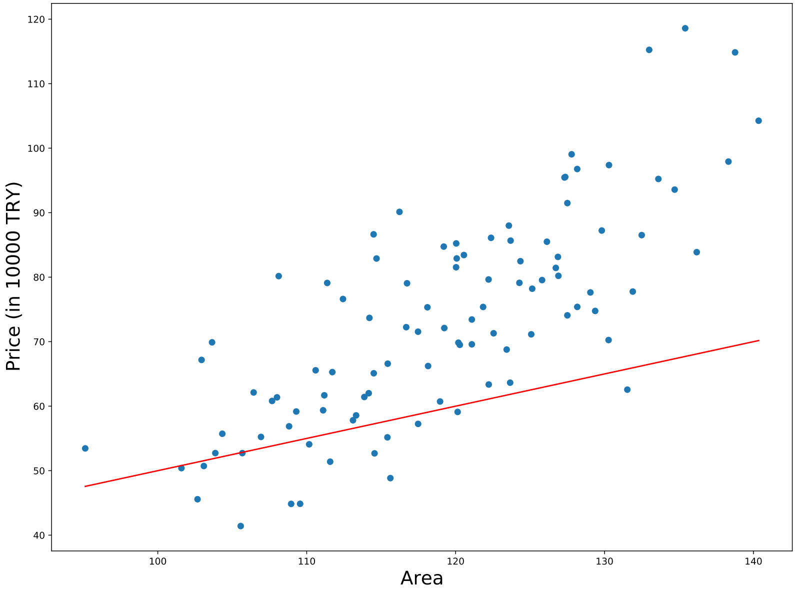


Figure: A fit without optimal parameters.

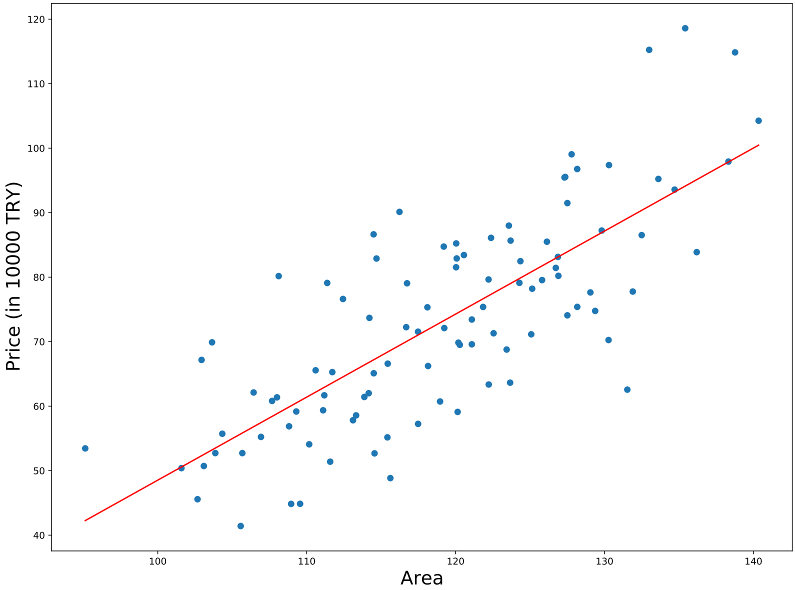


Figure: A fit using optimal parameters.

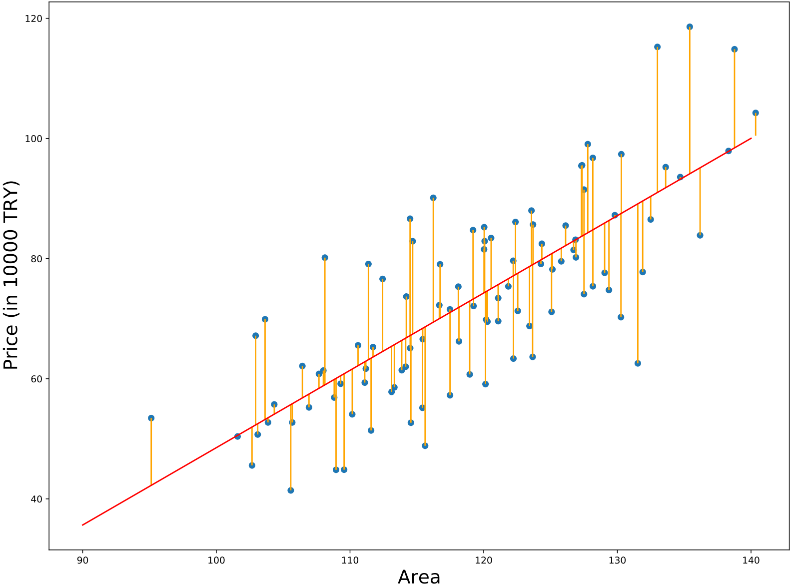


Figure: Visualizing error terms.

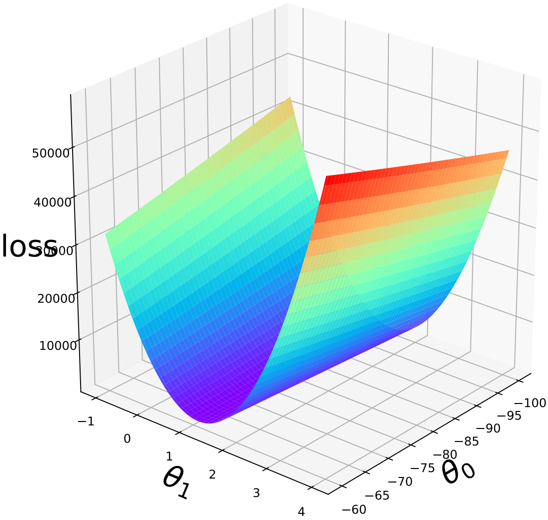


Figure: Objective function.

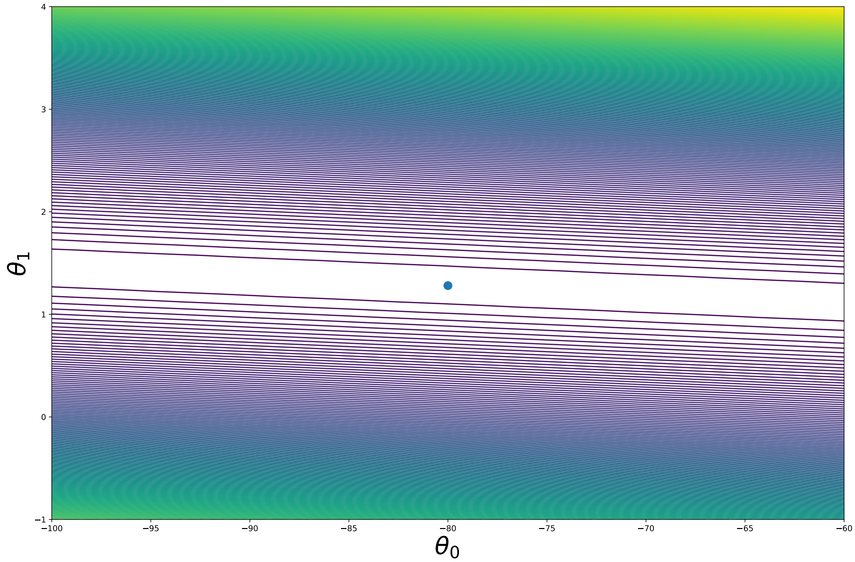


Figure: Objective function (Bird's eye).

Scaling

Scaling the input variables allows us to improve the optimization process by allowing us to correct the shape of the objective function. It does not disrupt the shape of the input variables. It just allows us to find the optimal parameters faster.

- Standard Scaler: The variables are scaled with respect to their means and standard deviations.

$$x_{scaled} = \frac{x - x_{mean}}{x_{std}}$$

- MinMax Scaling: The variables are scaled in the range [0,1] according to the following formula

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

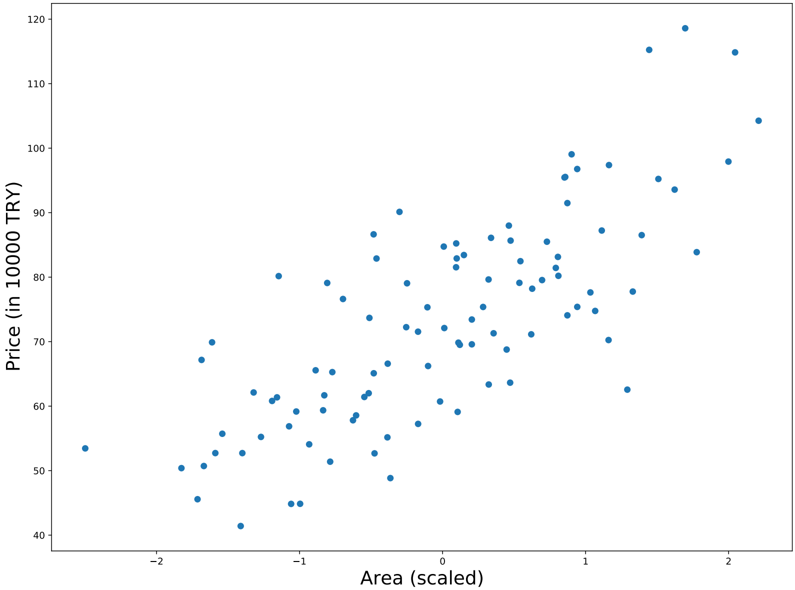


Figure: Scaled data set.

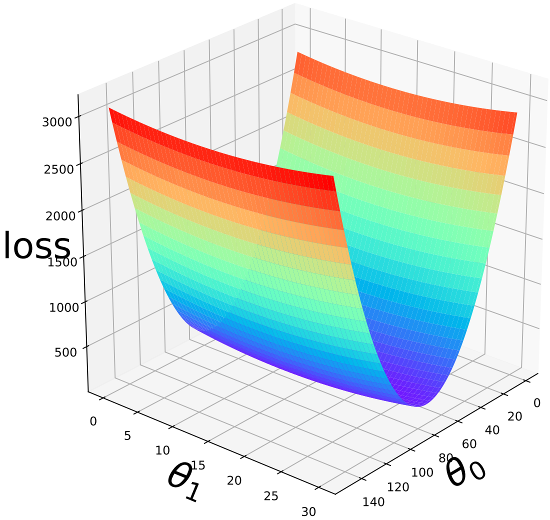


Figure: Scaled objective function.

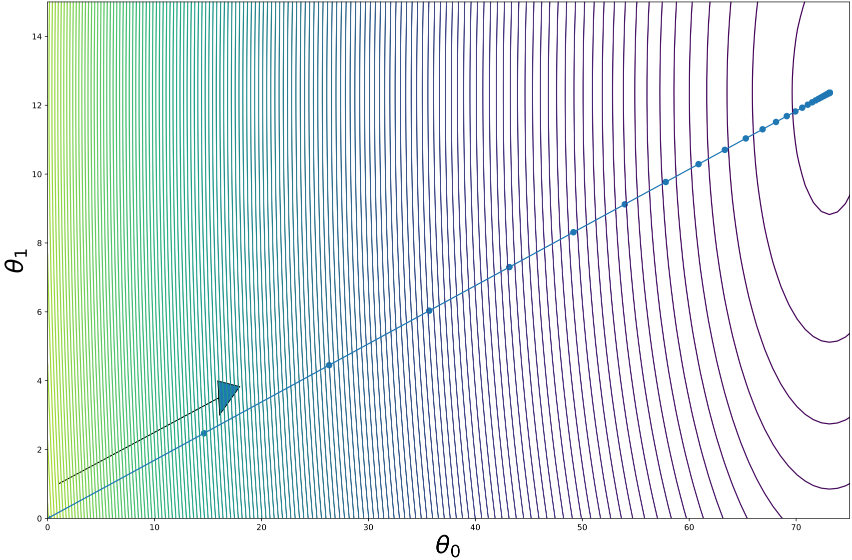


Figure: How we move using the gradient descent algorithm starting from $\theta_0 = \theta_1 = 0$.

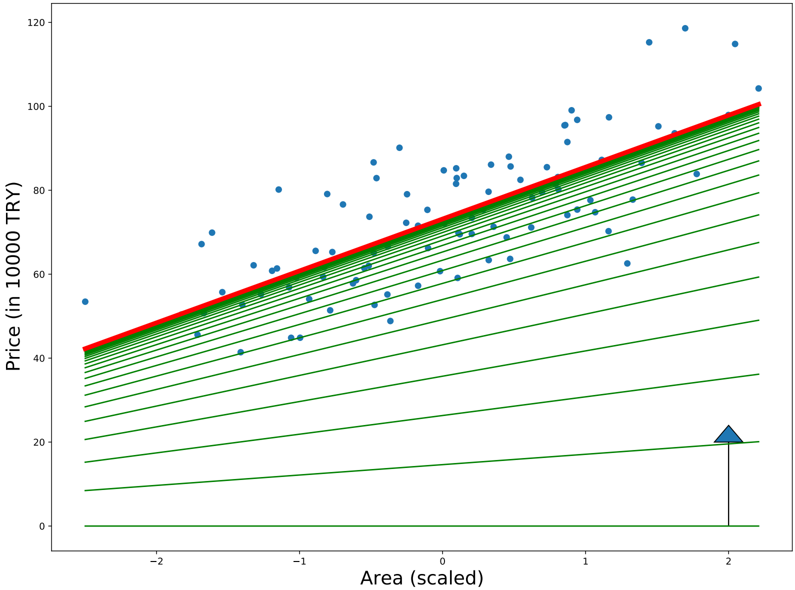


Figure: How we improve the predictions starting from below.

References

Andrew Ng lecture notes
http://cs229.stanford.edu/notes/cs229-notes1.pdf
François Chollet, Deep Learning with Python