ECS171: Machine Learning

L3 Validation of hypotheses: regression

Instructor: Prof. Maike Sonnewald
TAs: Pu Sun & Devashree Kataria

MOCO Amsterdam garden

Intended Learning Outcomes

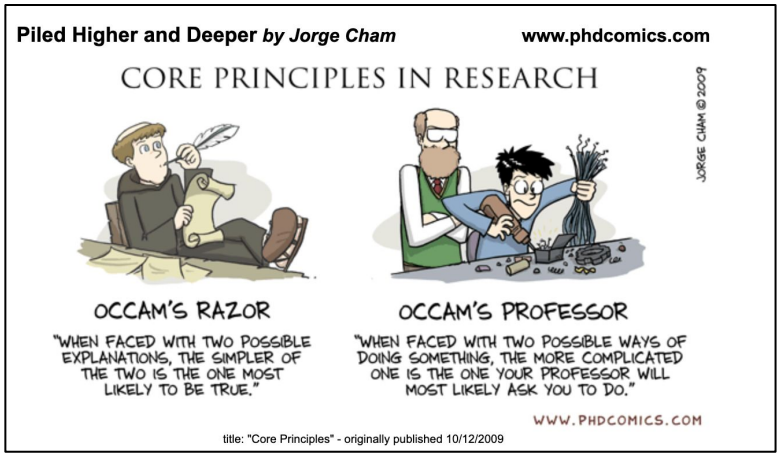
- Describe bias and variance, as well as the bias-variance trade-off
- For regression models: Describe difference in the expression for quantifying errors and calculate these for data
- Describe mathematically and apply, for linear regression and multivariate linear regression the analytical (Ordinary Least Squares) and numerical (Gradient Descent) methods for determining fit

Rec. reading: B 3.1 + R 1-7, R 3.4.6

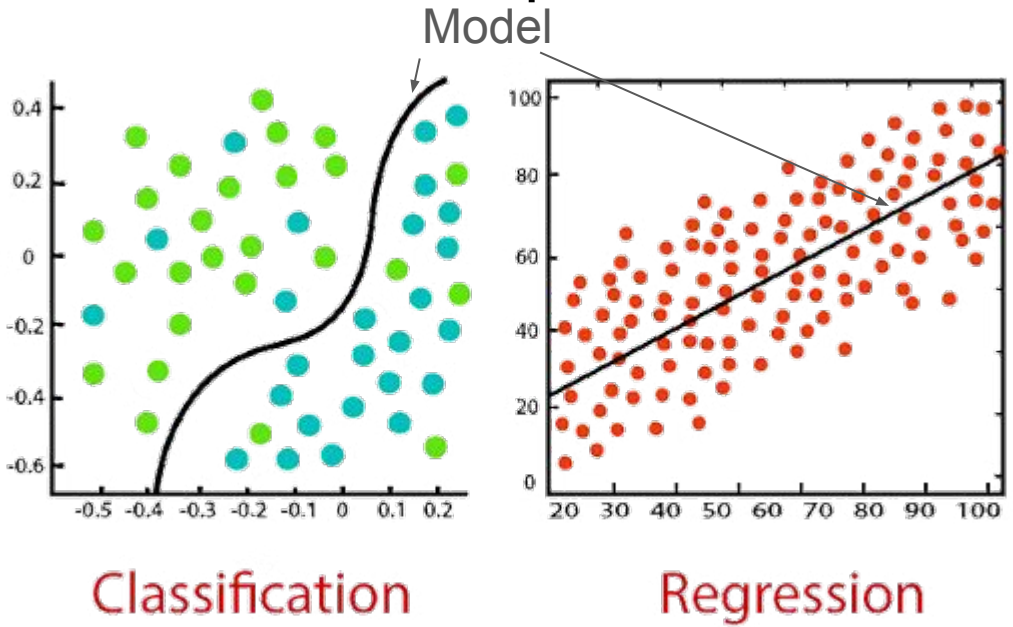
Resource: matrix_OLS_NYU_notes.pdf

Hypothesis testing: Goodness of fit

- Indicate the fit of a trained model and the causes of poor performance in machine learning.
- Goal of training a model: **Obtaining a generalized model to perform well on unseen data.**
- **Overfitting:**
  - A function (ML model) is too closely aligned with a limited set of data
- **Underfitting:**
  - A function (ML model) is not aligned with a limited dataset
- **Model complexity:**
  - As simple as possible, but no simpler



A model is a function that represents the data



Hypothesis: If I fit an ML model it will mimic the underlying ‘model’ the data came from

Predicting **continuous outputs:** Regression

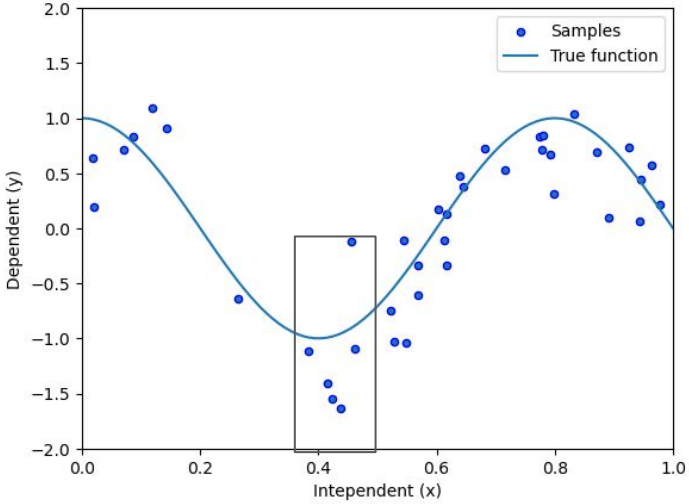
We need:

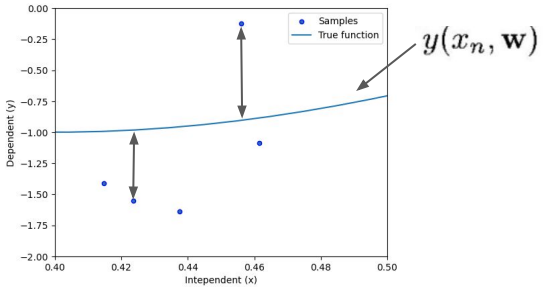
- Features (inputs): we'll call these x (or **x** if vectors)
- Training examples: many x^i for which y^i is known
- A **model:** Function that represents the relationship between x and y

$$y(x) = \text{function}(x, \mathbf{w})$$

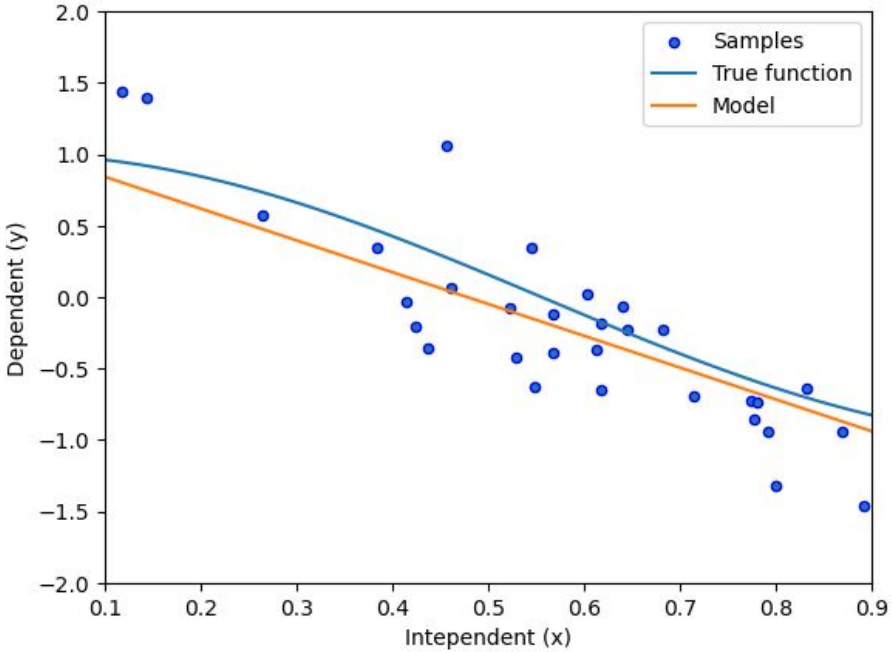
Linear: $y(x) = w_0 + w_1$

- A **loss function**: Estimate how well our model approximates the training examples (aka cost or objective function)
- **Optimization**, a way of finding the parameters of our model that minimizes the loss function





The difference between the ‘true’ function and our model is estimated using our observations



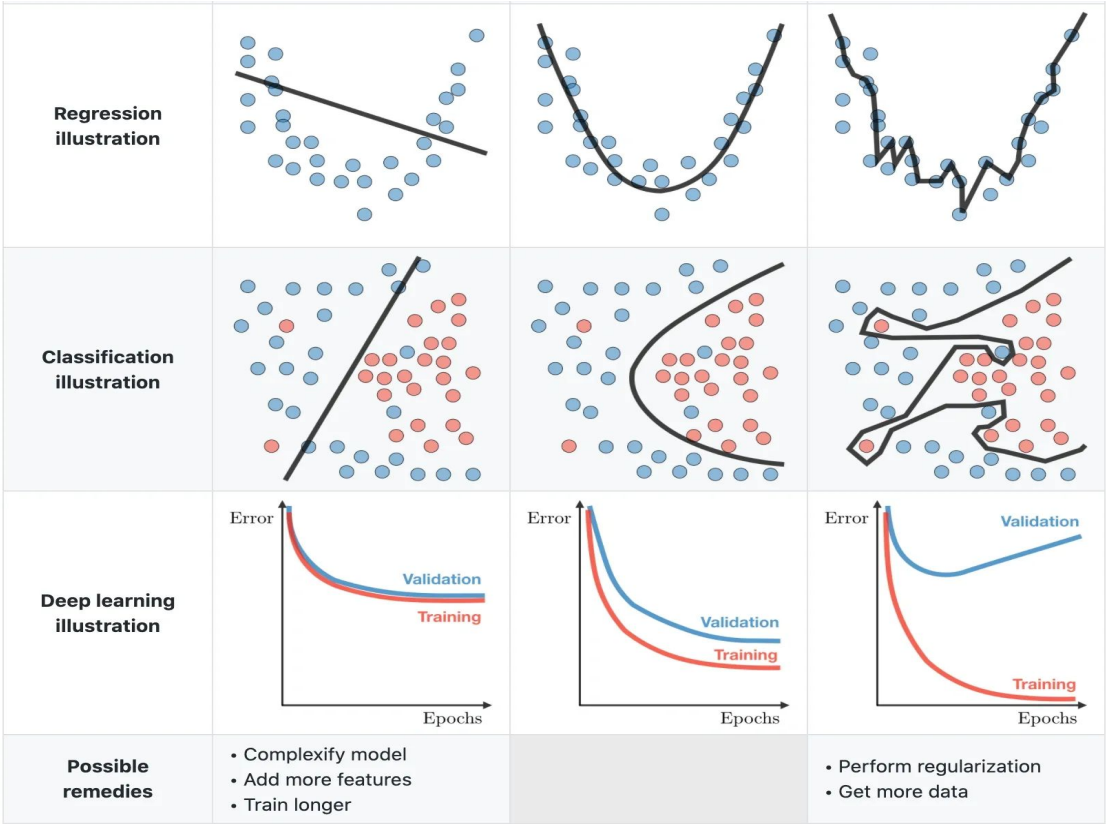
Symptoms of over and underfitting

Underfitting:

- High training error
- Training and test error similar
- High bias

Overfitting:

- Very low training error
- Test error much higher training error
- High variance on test data



How do we measure a model's performance?

1. Regression Metrics (Continuous values)

- **Sum of Squares Error (SSE)**: Measures the total deviation of the response values from the fit to the response values.
- **Mean Absolute Error (MAE)**: The average of the absolute differences between the predicted values and actual values.
- **Mean Squared Error (MSE)**: The average of the squared differences between the predicted values and actual values.
- **Root Mean Squared Error (RMSE)**: The square root of MSE, often more interpretable as it is in the same units as the response variable.
- **R-squared (Coefficient of Determination)**: Measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

2. Classification Metrics (Categorize data into labels)

- **Accuracy**: The proportion of total predictions that were correct.
- **Precision**: The proportion of positive identifications that were actually correct.
- **Recall (Sensitivity)**: The proportion of actual positives that were identified correctly.
- **F1 Score**: The harmonic mean of precision and recall.
- **Confusion Matrix**: A table used to describe the performance of a classification model, showing the actual vs. predicted values.
- **ROC-AUC**: The area under the receiver operating characteristic curve, measuring the trade-off between true positive rate and false positive rate.
- **Precision-Recall Curve**: Focuses on the performance with respect to the positive (minority) class.

3. Clustering Metrics (Grouping data into clusters)

- **Silhouette Score**: Measures how similar an object is to its own cluster compared to other clusters.
- **Davies-Bouldin Index**: The average 'similarity' between each cluster and its most similar cluster, where lower values indicate better clustering.
- **Calinski-Harabasz Index**: The ratio of the sum of between-clusters dispersion and of within-cluster dispersion.

4. Time Series Metrics

- **Mean Absolute Percentage Error (MAPE)**: The mean absolute percentage difference between the predicted and actual values.
- **Symmetric Mean Absolute Percentage Error (sMAPE)**: An adjustment to MAPE that handles zero values more effectively.

5. Other Methods

- **Cross-Validation**: A method to assess the robustness of the model by training and testing the model on different subsets of the dataset.
- **Learning Curves**: Plotting the model performance on the training set and the validation set over time or over the number of datasets.
- **Feature Importance**: Evaluating which features contribute most to the model's predictive power.

Too simple: inflexible learning due to too few/wrong features or too strict regularization $\rightarrow$ little variance but more bias

Too complex: more prediction variance

- $\hat{y}_i$ is the predicted value for the i^{th} observation.
- y_i is the true value for the i^{th} observation.
- n is the total number of observations.
- $\bar{y}$ is the mean of all observed values of the dependent variable.

Quantifying errors:

Bias is the average of the errors predictions made by the model and the true values

$$\text{Bias} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

Sum of Squares Error: Squaring avoids cancellation of pos and neg and emphasizes larger errors. Aka Residual Sum of Squares Error (RSS)

$$\text{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Squared Error: Averaged SSE, giving an average error per data point

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- $\hat{y}_i$ is the predicted value for the i^{th} observation.
- y_i is the true value for the i^{th} observation.
- n is the total number of observations.
- $\bar{y}$ is the mean of all observed values of the dependent variable.

Quantifying errors cont.

Mean Absolute Error: Avoids squaring errors, useful if large errors are not worse than smaller ones

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

Root Mean Squared Error: Same units as response variable, good when large errors are problems

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

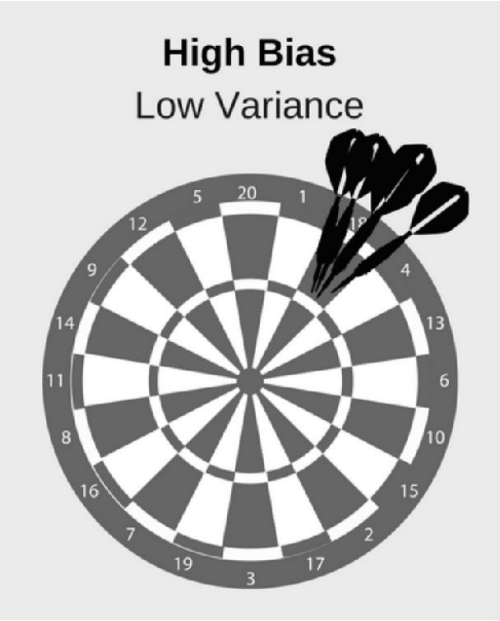
Coefficient of Determination (R^2): Measures of how well the independent variables explain the variability in the dependent variable in a regression model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Bias and variance

Bias

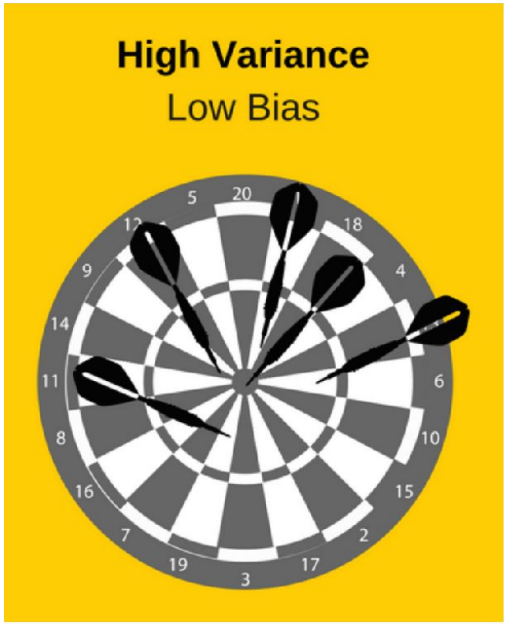
- **Definition**: Bias refers to the error due to overly simplistic assumptions in the learning algorithm. It can lead to the model underfitting the training data, meaning it does not capture the underlying trends well.
- **Characteristics**: A high-bias model is likely to have a lower level of complexity, which makes it less flexible in learning from the data. This results in the model missing relevant relations between features and target outputs.
- **Consequence**: High bias can cause the model to be less accurate on both training and testing data, leading to poor generalization.



Bias and variance

Variance

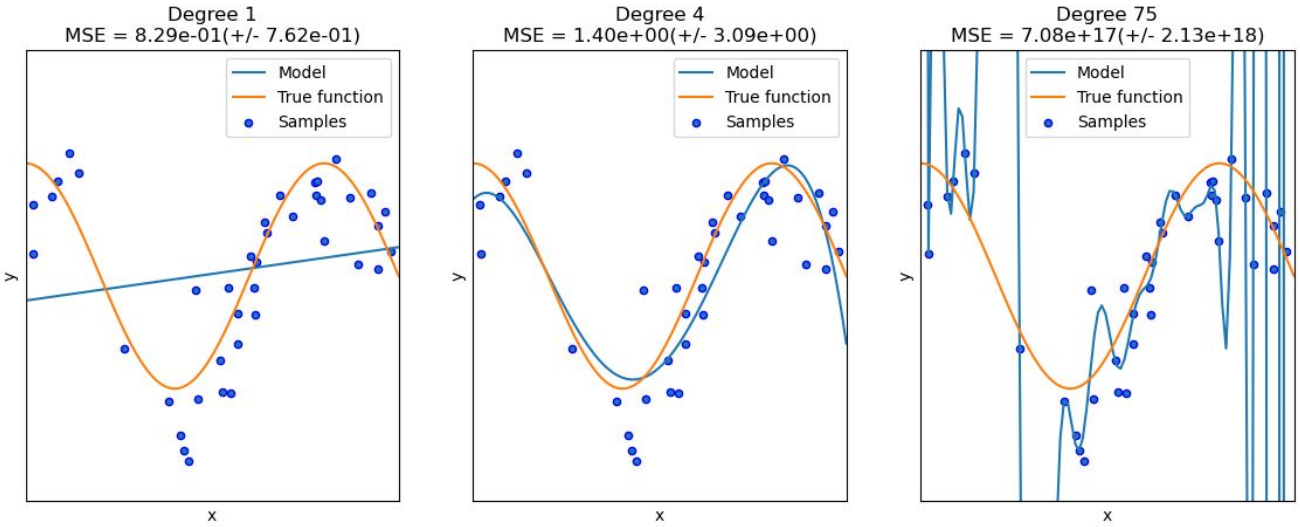
- **Definition**: Variance refers to the error due to too much complexity in the learning algorithm. It can lead to the model overfitting the training data, meaning it captures noise along with the underlying patterns.
- **Characteristics**: A high-variance model is extremely sensitive to the fluctuations in the training data. It learns features from the training data that don't generalize to unseen data.
- **Consequence**: High variance can cause the model to perform well on the training data but poorly on unseen (test) data due to overfitting.



Variance is measured through, for example, the difference between SSE for the training and the testing data. If SSE_{test} - SSE_{train} is high the model has high variance

Bias-variance trade-off

- **Bias** is about the simplicity of the model - high bias can lead to underfitting.
- **Variance** is about the complexity of the model - high variance can lead to overfitting.
- Effective machine learning involves managing this tradeoff to achieve a model that generalizes well to new, unseen data.



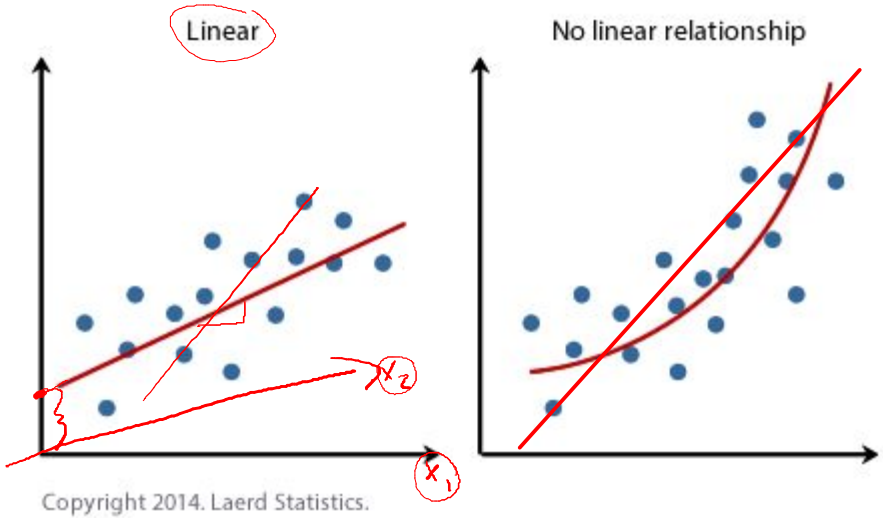
Linear Regression (LR) fits a linear function

Goal to find function f so that:

$$f(x) = y$$

Two approaches to determine f(x):

- Analytical: Ordinary Least Squares (OLS)
- Numerical: Gradient Descent

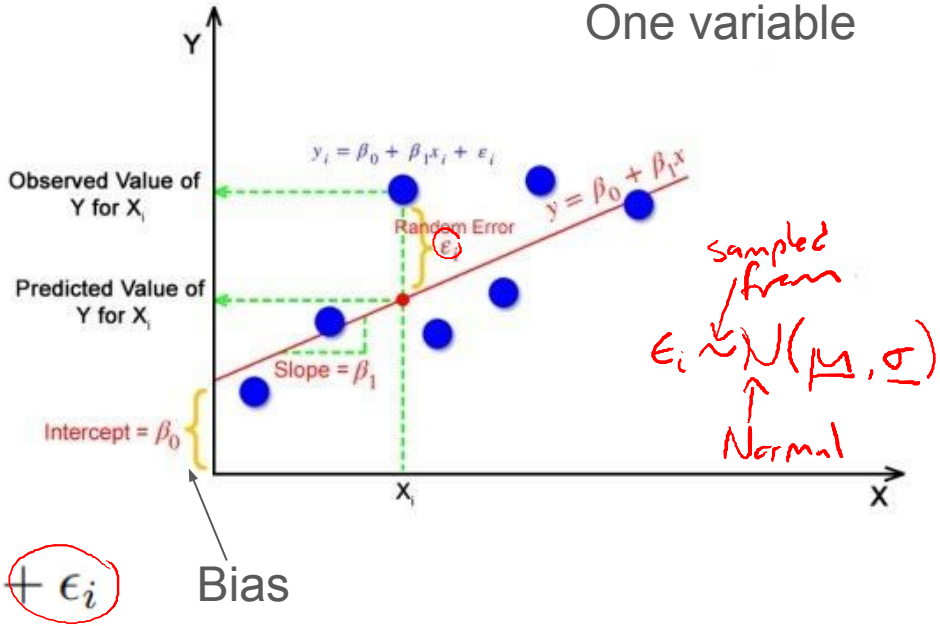


Linear Regression (LR) fits a linear function

Consider a dataset with n observations (x_i, y_i), where y_i is the dependent variable and x_i is the independent variable. The linear regression model is:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

We aim to find the values of β_0 (intercept) and β_1 (slope) that minimize the sum of squared residuals (errors), where the residual for each observation is $\epsilon_i = y_i - (\beta_0 + \beta_1 x_i)$.

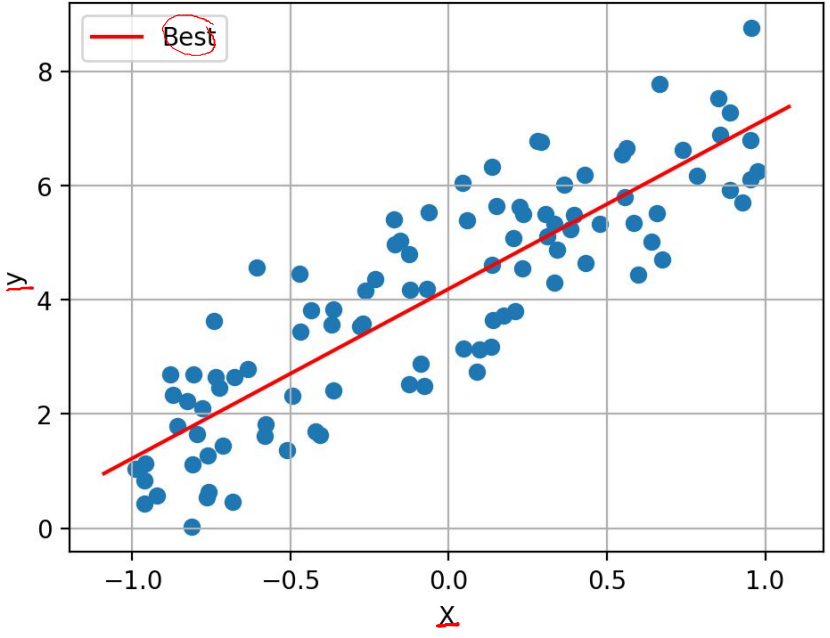


The 'true' value is: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$

The predicted value is: $\hat{y}_i = \beta_0 + \beta_1 x_i$

Estimating goodness-of-fit

- The fit of the model is estimated looking at the errors



Best line:
- bias: -5.25135e-16
- SSE: 99.2439
- MSE: 0.992439
- MAE: 0.849258
- RMSE: 0.996212
- R^2: 0.935972

Example: example_linear_regression.ipynb

Multivariate Linear Regression

Basic equation for a multivariate linear regression model:

$$y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_p x_{ip} + \epsilon_i$$

y_i is the dependent variable for the i^{th} observation.

$x_{i1}, x_{i2}, \ldots, x_{ip}$ are the independent variables (predictors) for the i^{th} observation.

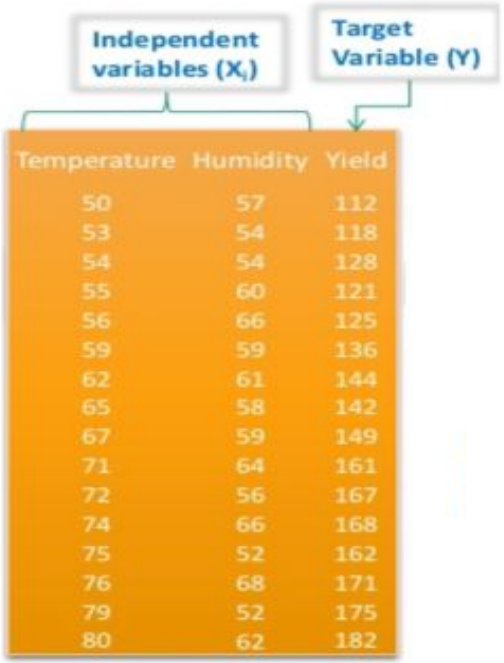
parameters
weights

$w_0, w_1, w_2, \ldots, w_p$ are the coefficients to be estimated.

ϵ_i is the error term for the i^{th} observation.

The w_0 is the bias aka the intercept

Model achieves a good fit for the regression line by finding the best coefficient values (w) that **minimize the errors.**



With many variables (predictors) we fit (hyper)planes

Each predictor (i) has an associates slope (w)

For each dependent variable (data point i), with observations indexed by p: $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$

We incorporate the bias w_0 by setting $x_0=1$, so the prediction for n observations becomes:

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i$$

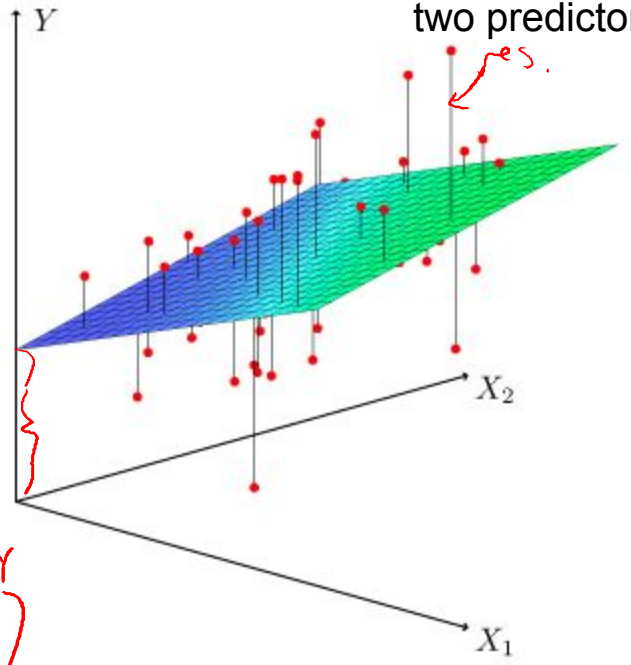
The true value includes an error term of the residuals:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \epsilon_i$$

Using matrix notation:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \epsilon$$

Example plane with two predictors

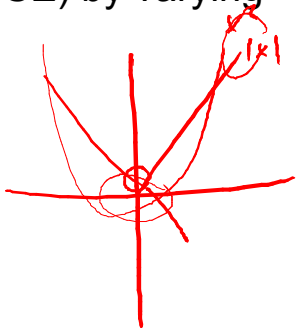


L(w)

The cost function is used to minimize the error

- The cost function (J) is here defined to minimize the sum of squared errors (SSE) by varying the coefficients **w**:

$$J(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



- Recall, the predicted value is: $\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_p x_{ip}$

- By minimizing J(**w**) the relationship between y_i and $\hat{y}_i$ can be approximated in the best available way

- Does not have to be SSE

LASSO RIDGE

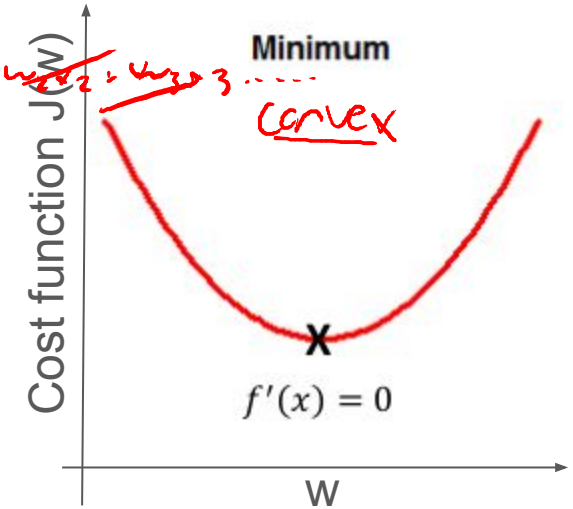
Fitting the model analytically: Ordinary Least Squares (OSL)

- Differentiation finds the point at which a function f(x) is at a minimum denoted f'(x)
- We can differentiate the cost function J(**w**) wrt **w** to find our minimum: $\frac{\partial J(w)}{\partial w_i} = 0$
- Minimization leads to a linear set of equations:

Gradient

nabla $\nabla_w J(w)$

$$= \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_p} \end{bmatrix} \begin{matrix} = -2\sum_{i=1}^{n}(y_i - \hat{y}_i) = 0 \\ = -2\sum_{i=1}^{n} x_{i1}(y_i - \hat{y}_i) = 0 \\ \\ = -2\sum_{i=1}^{n} x_{ip}(y_i - \hat{y}_i) = 0 \end{matrix}$$



Matrix notation

Given a dataset with n observations and p independent variables, the model is:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \epsilon$$

Where:

- $\mathbf{Y}$ is an $n \times 1$ column vector of the dependent variable.
- $\mathbf{X}$ is an $n \times (p+1)$ matrix of the independent variables, with the first column as 1s for the intercept term.
- $\mathbf{W}$ is a $(p+1) \times 1$ column vector of coefficients (including the intercept).
- ϵ is an $n \times 1$ column vector of the residuals.

The response vector $\mathbf{Y}$ is:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- y_i represents the value of the dependent variable for the i^{th} observation.

The data matrix $\mathbf{X}$ is structured as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

- Each row represents an observation.
- The first column is all 1's for the intercept term.
- x_{ij} represents the value of the j^{th} predictor for the i^{th} observation.
- n is the number of observations, and p is the number of predictors.

The coefficient vector $\mathbf{W}$ is structured as:

$$\mathbf{W} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix}$$

- w_0 is the intercept term.
- $w_1, \dots, w_p$ are the coefficients of the predictors.

Minimize a cost function to fit the model finding optimal w

- The optimal weights are found minimizing the cost function $\hat{\mathbf{W}}$

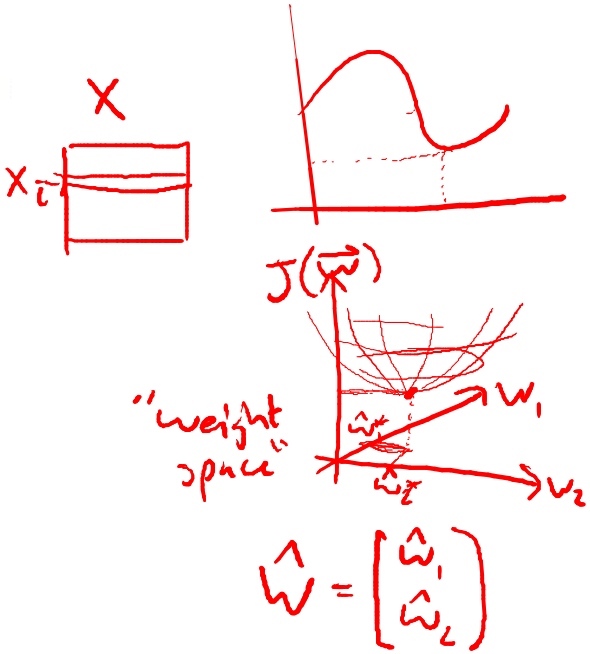
$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Using the matrix notation for $\hat{y}_i$:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{X}_i \mathbf{W})^2$$

- In more compact form:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} (\mathbf{Y} - \mathbf{X}\mathbf{W})^T (\mathbf{Y} - \mathbf{X}\mathbf{W})$$



Fitting the model analytically: Ordinary Least Squares (OSL)

- In matrix notation (J(w) is now SSE):

$$\nabla_{\mathbf{W}} J = \nabla_{\mathbf{W}} [\mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{X}\mathbf{W} - \mathbf{W}^T\mathbf{X}^T\mathbf{Y} + \mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}]$$

- Taking the derivative term by term, and noting that Y^TY is a constant with respect to W, and Y^TXW and W^TX^TY are equivalent, the derivative simplifies to:

$$\nabla_{\mathbf{W}} J = -2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\mathbf{W}$$

- Set to zero:

$$-2\mathbf{X}^T\mathbf{Y} + 2\mathbf{X}^T\mathbf{X}\mathbf{W} = 0$$

- Rearranging:

$$\mathbf{X}^T\mathbf{X}\mathbf{W} = \mathbf{X}^T\mathbf{Y}$$

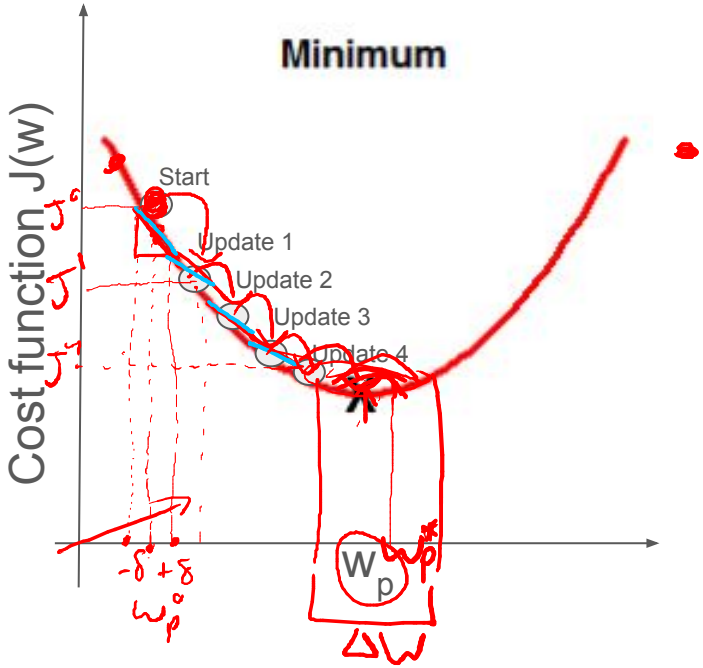
- Finally:

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- This result gives you the OLS estimator for the coefficients in multiple linear regression.

Fitting the model Numerically: Gradient Descent

- If we can't find an analytical solution we use a numerical method to find optimal weights **w**
- Gradient descent is iterative (do many times with updates):
  - Initialise **w** e.g. with random numbers
  - Calculate J(**w**), change w_i, ask: Has the error J(**w**) gotten smaller?
  - We update w_p: Is it moving towards an optimum?



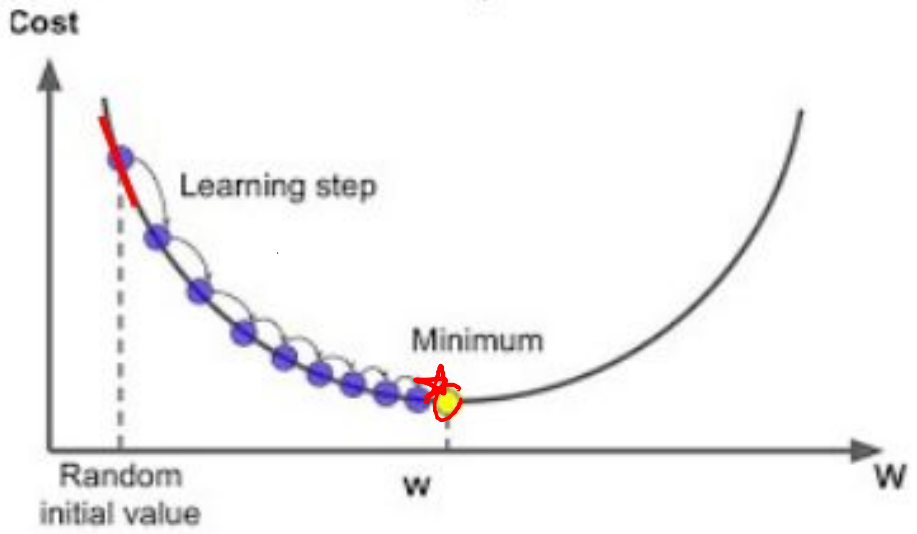
It can also be a 'hill-climber'. Take note of the sign of the cost function

Fitting the model Numerically: Gradient Descent

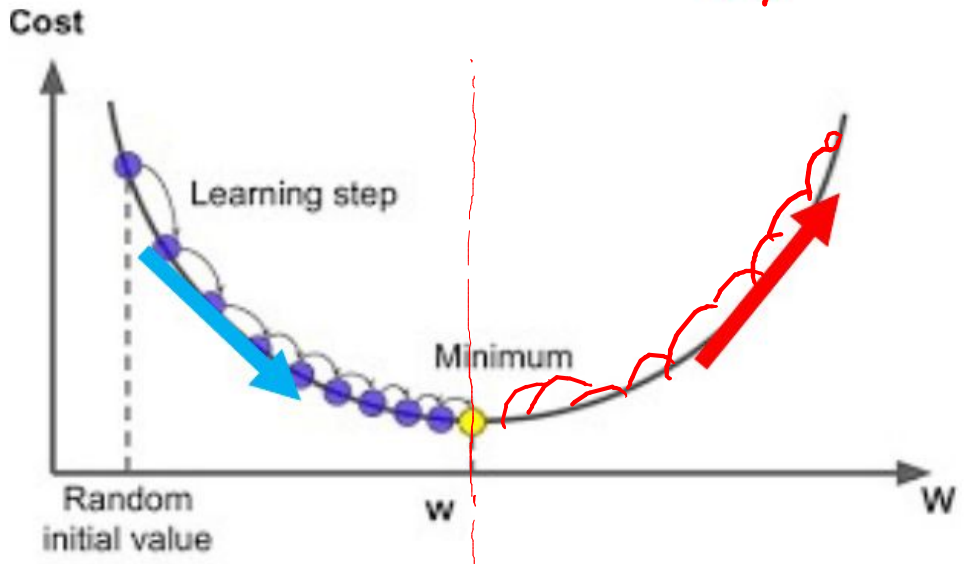
- How do we know which direction to move the weights?
  - Assess the gradient resulting from changing the weights for one observation (w_i)

$$\text{if } \frac{\partial J(\mathbf{w})}{\partial w_i} < 0 \Rightarrow \text{increase } w_i$$

$$\text{if } \frac{\partial J(\mathbf{w})}{\partial w_i} > 0 \Rightarrow \text{decrease } w_i$$



$\frac{\partial RSS}{\partial w_j} < 0 \rightarrow increase\ w_j$ — $\frac{\partial RSS}{\partial w_j} > 0 \rightarrow decrease\ w_j$



j here is i normally

Least Mean Squares (LMS) update rule

- Repeatedly update **w** based on the gradient.

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

New old

- Here, λ is the learning rate
- For a datapoint (predicting one datapoint i with p independent variables) this gives us the Least Mean Squares (LMS) update rule:

$$\mathbf{w}_{new} \leftarrow \mathbf{w}_{old} - 2\lambda(y_i - \hat{y}_i(x_i))x_i$$

Error

- As error approaches zero, so does the update (**w** changes less)

Optimizing across a training set

- To generalise to all data points in the training set options include:
  - 1: Batch updates: Sum or average updates across every example i, then change the parameter values:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \sum_{i=1}^{n} (y_i - \hat{y}_i(x_i))x_i$$

  - 2: Stochastic/online updates: Update the parameters for each data point in turn, according to its own gradients

Algorithm 1 Stochastic gradient descent

1: Randomly shuffle examples in the training set
2: **for** $i = 1$ to n **do**
3: Update:

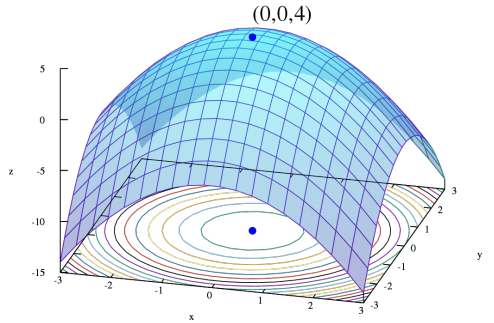
$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda(y_i - \hat{y}_i(x_i))x_i \quad \text{(update for a linear model)}$$

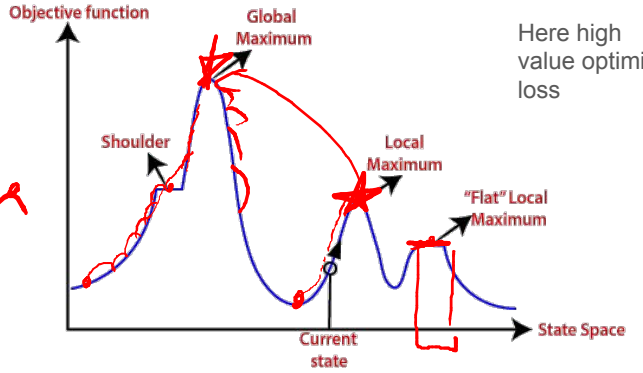
4: **end for**

Gradient Descent notes

- The learning rate (λ) represents the size of the 'steps' of the descent
- Pros:
  - Intuitive
  - Heuristic approach (stochastic optimisation)
- Cons:
  - Can take many iterations to converge
  - Only optimal for 'smooth' functions

We will get back to working through an example of the GD Update Rule for 1 observation

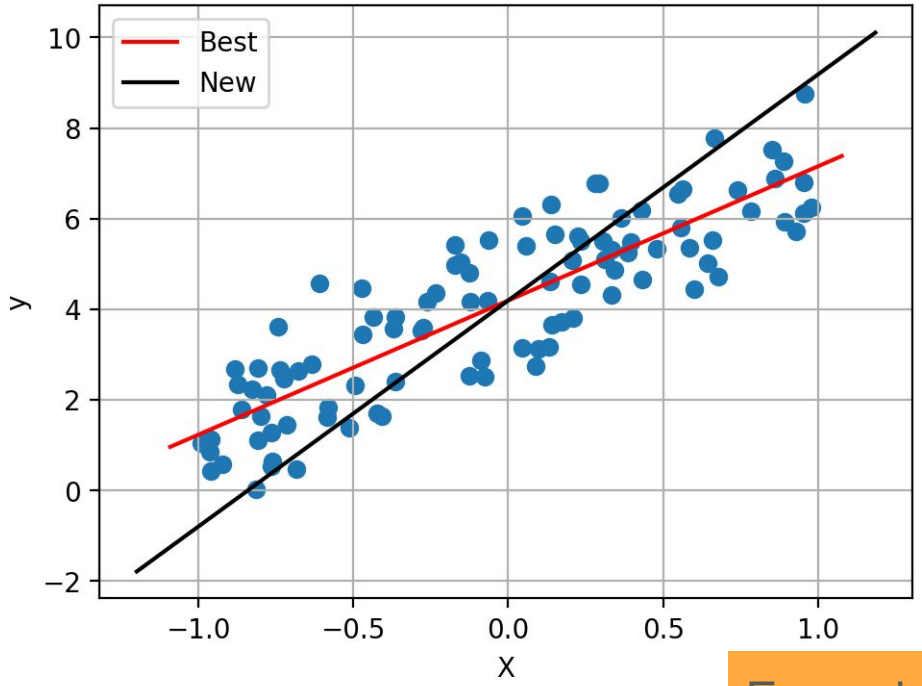




Here high value optimises loss

A one-dimensional state-space landscape in which elevation corresponds to the objective function

The choice of loss function can significantly impact results



Best line:
- bias: -5.25135e-16
- SSE: 99.2439
- MSE: 0.992439
- MAE: 0.849258
- RMSE: 0.996212
- R^2: 0.935972

New line:
- bias: 0.11054
- SSE: 237.681
- MSE: 2.37681
- MAE: 1.30382
- RMSE: 1.54169
- R^2: 0.635038

Example: example_linear_regression.ipynb