Applied Machine Learning
Class 2

Master 2 Physique Fondamentale et Applications

Daniel Förster
daniel.forster@univ-orleans.fr
UFR Sciences et Techniques Université d'Orléans

Chapter 3: Supervised Learning: Regression and Classification

- Regression (linear, multi-linear, non-linear, heteroscedastic)
- Logistic regression
- Decision trees
- Support vector machines
- k-nearest neighbors algorithm
- neural networks

Chapter 3: Supervised Learning
A: Regression

Problem formulation: Regression

Problem: Predict a continuous numerical value from input data.

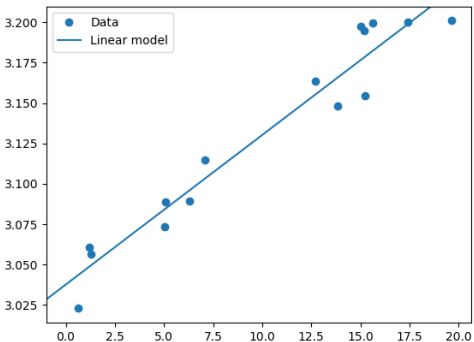
Example: Weight of a baby as a function of age.

Loss function: Typically Mean Squared Error (MSE) or Mean Absolute Error (MAE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

Linear Regression as Machine Learning Model



Goal: Fit a line to minimize the sum of squared errors.

Our "**model**":

$$y = wx + b$$

Find w ("**weight**") and b ("**bias**") by minimizing the following cost/objective/loss function:

$$L(w, b) = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2$$

with

$$\hat{y}_j = wx_j + b$$

This is the "mean square error" **loss**

Derivation: Intercept (b)

Derivation: Intercept (b) Start with the Loss Function:

$$L = \sum_{i=1}^{n} (y_i - (wx_i + b))^2$$

Step 1: Take the Partial Derivative with Respect to b:

$$\frac{\partial L}{\partial b} = -2 \sum_{i=1}^{n} (y_i - (wx_i + b))$$

Step 2: Set $\frac{\partial L}{\partial b} = 0$:

$$\sum_{i=1}^{n} y_i = nb + w \sum_{i=1}^{n} x_i$$

Step 3: Solve for b:

$$b = \overline{y} - w\overline{x}$$

Derivation: Slope (w)

Step 1: Take the Partial Derivative with Respect to w:

$$\frac{\partial L}{\partial w} = -2 \sum_{i=1}^{n} x_i \left(y_i - (w x_i + b) \right)$$

Step 2: Set $\frac{\partial L}{\partial w} = 0$:

$$\sum_{i=1}^{n} x_i y_i = b \sum_{i=1}^{n} x_i + w \sum_{i=1}^{n} x_i^2$$

Step 3: Substitute $b = \overline{y} - w\overline{x}$:

$$\overline{xy} = (\overline{y} - w\overline{x})\overline{x} + w\overline{x^2}$$

Step 4: Solve for w:

$$w = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}$$

Linear Regression: Finding b and w

Therefore:

- Intercept or bias (b):

$$b = \frac{\overline{y}\,\overline{x^2} - \overline{x}\,\overline{xy}}{\overline{x^2} - \overline{x}^2}$$

- Slope or weight (w):

$$w = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}$$

Definitions:

- $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
- $\overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$
- $\overline{x^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^2$

Gradient Descent as Optimization Algorithm

Goal: Minimize a function $L(\{p\})$ by iteratively updating parameters $\{p\}$.
Here $p = \{w, b\}$, where:

- w: Slope of the linear regression line
- b: Intercept of the line

Gradient descent update rule for linear regression:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}, \quad b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

where:

- $L(\{w, b\})$: Loss function (e.g., mean squared error)
- η: **Learning rate**, controlling the step size.
- $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}$: Partial derivatives with respect to w and b.

Gradient Descent: Why does it work?

Taylor series (1D):

$$L(x) = L(x_0) + \left.\frac{\partial L}{\partial x}\right|_{x_0} (x - x_0) + \frac{1}{2} \left.\frac{\partial^2 L}{\partial x^2}\right|_{x_0} (x - x_0)^2 + \ldots$$

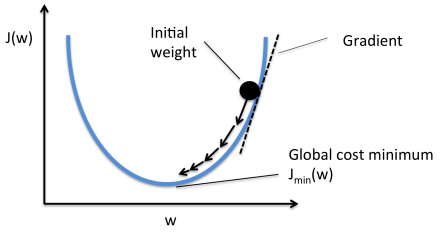


Image source [1]

Therefore, with a small $\eta > 0$:

$$\begin{aligned} L\left(x_0 - \eta \left.\frac{\partial L}{\partial x}\right|_{x_0}\right) &\approx L(x_0) + \left.\frac{\partial L}{\partial x}\right|_{x_0} \left(x_0 - \eta \left.\frac{\partial L}{\partial x}\right|_{x_0} - x_0\right) \\ &\approx L(x_0) - \eta \left(\left.\frac{\partial L}{\partial x}\right|_{x_0}\right)^2 \\ &< L(x_0) \end{aligned}$$

Gradient descent algorithm $x_0 \leftarrow x_0 - \eta \left.\frac{\partial L}{\partial x}\right|_{x_0}$ decreases for a continuous, differentiable, and convex L.

This can be improved

Problem with Standard Gradient Descent:

- Large gradients cause large parameter updates, leading to instability.
- Small gradients cause small updates, slowing down convergence.
- Different directions in the loss surface might have different gradient magnitudes, making it challenging to choose an optimal learning rate.

3 Ideas:

- Learning rate scheduling
- Smooth the gradient updates using a "momentum" term.
- Normalize updates to make consistent progress in all directions.

Learning Rate Scheduling

Reduce learning rate η over time to balance faster initial learning with finer convergence later.

Typically:

- **Step Decay:** Reduce η by a fixed factor at regular intervals
- **Exponential Decay:** Gradually reduce η according to $\eta_t = \eta_0 e^{-\lambda t}$, where λ is the decay rate and t is the iteration step
- **Cosine Annealing:** Cyclical decay where η decreases following a cosine curve, useful for avoiding local minima

Momentum in Gradient Descent

Why Add Momentum?

- Momentum smooths the updates by combining the current gradient with the direction moved in previous steps.
- Helps to accelerate convergence along directions with consistent gradients.
- Reduces oscillations in regions with varying gradient directions.

Update Equations:

$$m_i \leftarrow \beta m_i + (1 - \beta)\nabla_i L(p)$$
$$p_i \leftarrow p_i - \eta m_i$$

where:

- m_i: Momentum term for each dimension i
- $\beta \approx 0.9$: Momentum coefficient, controlling the influence of past gradients.

Let's Normalize the Gradient

$$p_i \leftarrow p_i - \frac{\eta}{\sqrt{s+\epsilon}} \nabla_i L(p)$$

with

$$s = (\nabla L(p))^2 \qquad \text{and} \qquad \epsilon \quad \text{a small number} (\approx 10^{-5})$$

$$m_i \leftarrow \beta m_i + (1-\beta) \nabla_i L(p)$$
$$v_i \leftarrow \gamma v_i + (1-\gamma) (\nabla_i L)^2$$

where:

- m_i is the moving average of gradients.
- v_i is the moving average of squared gradients.
- β and γ are momentum coefficients for m_i and v_i.

ADAptive Moment estimation: Adam (2014)

Initial values of m_i and v_i are close to zero.
To correct for this bias, Adam adjusts m_i and v_i as follows at iteration t:

$$\hat{m}_{i,t} = \frac{m_{i,t}}{1 - \beta^{t+1}}$$
$$\hat{v}_{i,t} = \frac{v_{i,t}}{1 - \gamma^{t+1}}$$

The denominators approach 1 as t increases, reducing the effect of bias over time.

Final Update Rule:

$$p_i \leftarrow p_i - \eta \frac{\hat{m}_i}{\sqrt{\hat{v}_i + \epsilon}}$$

Types of Gradient Descent

Standard gradient descent uses the entire dataset.

Stochastic gradient descent:

- One data point at a time
- **or**: small batches of data.

Batch: pass through the entire training dataset

Multi-linear Regression

Predicts y as a linear combination of x.
Model:

$$y = b + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p$$

- b: Bias or intercept.
- $w_1, w_2, \ldots, w_p$: Weights for the independent variables.

Linearity: y is a linear function of x.

Example: House price from features like size, location, and number of bedrooms

Calculating the Weights w

Objective:

- Find the coefficients $b, w_1, \ldots, w_p$ that minimize the error between predicted and actual values.

Mathematical Formulation:

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - (b + \sum_{j=1}^{p} w_j x_{ij}) \right)^2$$

Non-Linear Regression

Relationship between the dependent variable y and independent variables x is modeled by a non-linear function.

- The model is represented as:

$$y = f(\mathrm{x}, \boldsymbol{\theta})$$

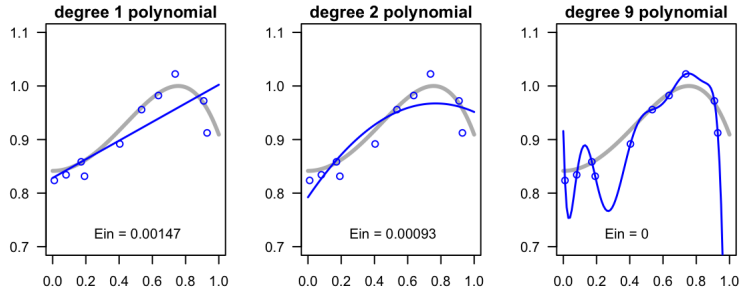
where:

- $f(\mathrm{x}, \boldsymbol{\theta})$: Non-linear function parameterized by $\boldsymbol{\theta}$.

Examples of Non-Linear Models:

- Polynomial regression: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n$
- Exponential regression: $y = \beta_0 e^{\beta_1 x}$
- Logistic growth models: $y = \frac{L}{1+e^{-k(x-x_0)}}$

Under- and overfitting

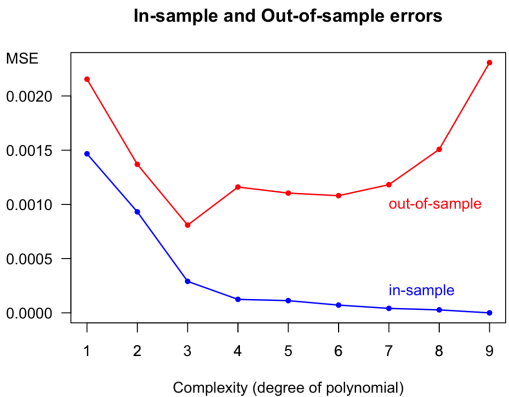


Overfitting: Model performs well on training data, but poorly on unseen data. It has essentially "memorized" the training data and fails to generalize to new instances.

Underfitting: Model performs poorly on both training and unseen data. It's too simplistic and fails to capture underlying patterns in the data.

Image source [2]

How to Diagnose Under- and Overfitting



- **Underfitting:**
  - Model has limited learning capacity.
  - Captures only the general trend, missing important details.
- **Overfitting:**
  - Model performs well on training data but poorly on unseen data.
  - Learns noise or irrelevant patterns in the training set.

Image source [3]

Heteroscedasticity

Variance of the data points is not constant.

Weighted Approach:

- Use weights to account for heteroscedasticity.
- Weight for each data point:

$$w_i = \frac{1}{s_i^2}$$

where s_i^2 is the variance (or squared standard deviation) of the corresponding y value.

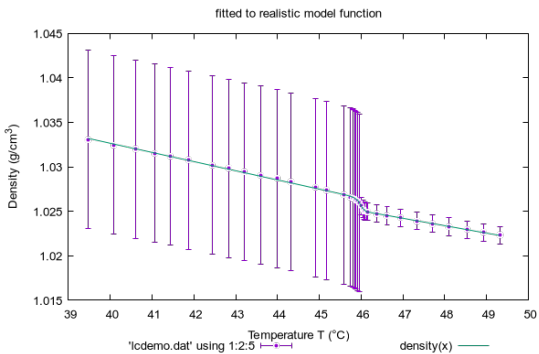


Image source [4]