Lecture 7-9 - Module 4.1. Linear regression

COMP 551 Applied machine learning

Yue Li
Assistant Professor
School of Computer Science
McGill University

January 28, 30, & Feb 4, 2025

Most commonly used ML algorithms in Kaggle 2021 survey

Module 4 →

Module 3 & 6 ✓

Module 6

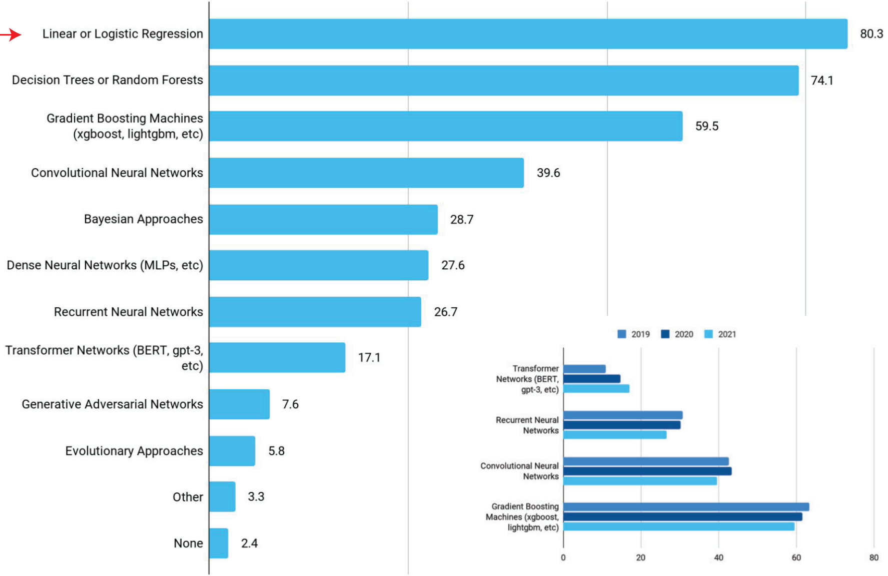
Module 5.2

Module 4 (briefly)

Module 5.1

Module 5.4

Module 5.4



Outline

Objectives

Simple linear regression

Multiple linear regression

Probabilistic interpretation

Non-linear basis function

Summary

Outline

Objectives

Simple linear regression

Multiple linear regression

Probabilistic interpretation

Non-linear basis function

Summary

Learning objectives

Understanding the following concepts

- Simple linear model
- Finding the best linear fit by minimizing SSE
- Matrix algebra in solving multiple regression
- Probabilistic interpretation
- Feature transformation by non-linear basis functions

Outline

Objectives

Simple linear regression

Multiple linear regression

Probabilistic interpretation

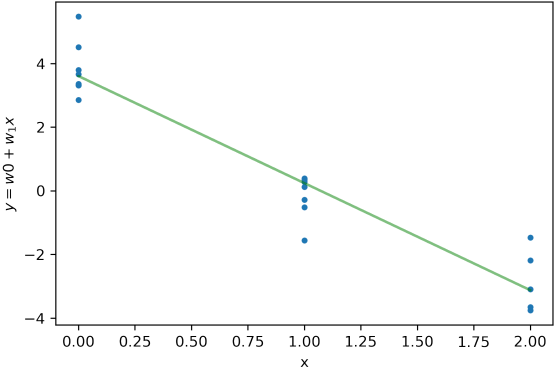
Non-linear basis function

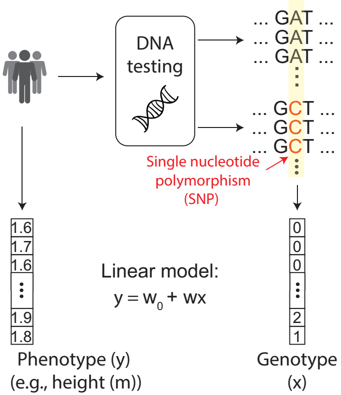
Summary

Linear regression using a one-dimensional input

We want to predict real-valued quantity or often known as **response or target variable** $y \in \mathbb{R}$ by finding a mapping function that maps from a **one-dimensional input** $\mathbf{x}$ to the real-valued y. We can fit a linear function on training examples $\{x^{(n)}, y^{(n)}\}_{n=1}^{N}$ (e.g., predicting phenotype from one genetic mutation):

$$f(x^{(n)}; w_0, w_1) = w_0 + w_1 x^{(n)} \tag{1}$$





DNA testing

... GAT ...
... GAT ...
... GAT ...

... GCT ...
... GCT ...
... GCT ...

Single nucleotide polymorphism (SNP)

1.6
1.7
1.6
⋮
1.9
1.8

Linear model:
$y = w_0 + wx$

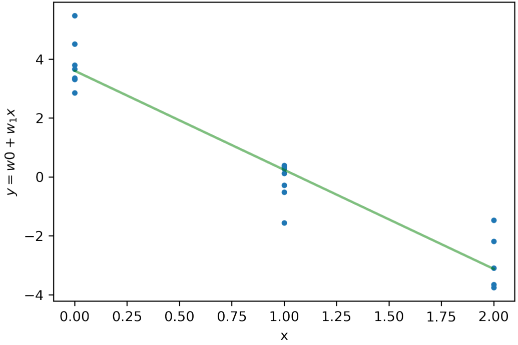
0
0
0
⋮
2
1

Phenotype (y)
(e.g., height (m))

Genotype
(x)

Linear regression using a one-dimensional input

$$f(x^{(n)}; w_0, w_1) = w_0 + w_1 x^{(n)}$$



- $w_0 = 3.4$ is the **intercept** or **bias**, which is not be confused with the "model bias"
- $w_1 = -3.25$ is the **slope** of the linear function or the **regression coefficient**.

Simple linear regression using one input feature

In statistics, we often write down the regression formula as:

$$y^{(n)} = w_0 + w_1 x^{(n)} + \epsilon^{(n)}$$

where $\epsilon^{(n)}$ is the prediction error for example n.

Using matrix notation, we can write the regression formula as:

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(N)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(N)} \end{bmatrix} \tag{2}$$

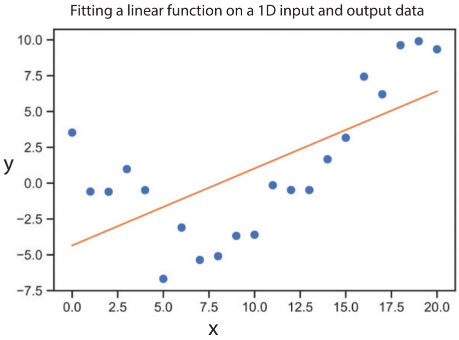
$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon} \tag{3}$$

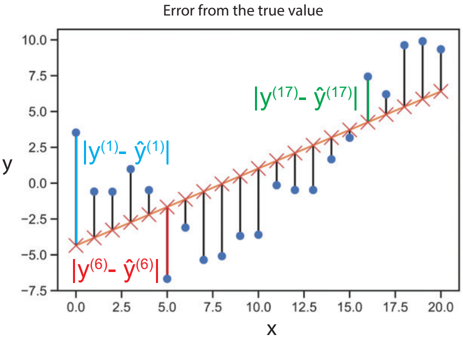
where $\mathbf{y}$ and $\boldsymbol{\epsilon}$ are both $N \times 1$ vectors, $\mathbf{X}$ is a $N \times 2$ matrix, and $\mathbf{w}$ is a 2×1 vector.

Residual error as a measure of prediction loss

Every straight line we fit incurs a prediction error on the training data point unless the fitted line goes through that data point. The **residual error** is Euclidean distance between the observed response $y^{(n)}$ value and the predicted response $\hat{y}^{(n)} = \mathbf{x}^{(n)}\mathbf{w}$:

$$l_n = ||y^{(n)} - \hat{y}^{(n)}||_2 = \sqrt{(y^{(n)} - \hat{y}^{(n)})^2} = |y^{(n)} - \hat{y}^{(n)}| \tag{4}$$


Fitting a linear function on a 1D input and output data


Error from the true value

Notation: $||\mathbf{a}||_2 = \sqrt{\sum_i a_i^2}$ is L_2-norm and $||\mathbf{a}||_2^2 = \sum_i a_i^2$ is the squared of the L2-norm.

Fitting a linear regression function by minimizing the sum of squared error

Sum of squared error (SSE) as a function of the linear coefficients **w** is defined as:

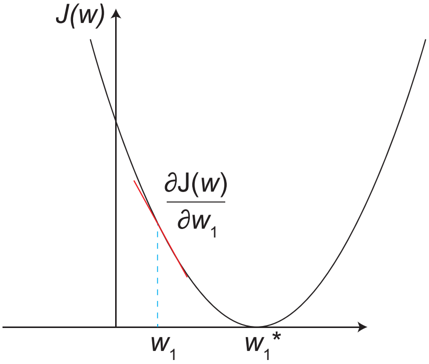
$$J(\mathbf{w}) = ||\mathbf{y}-\hat{\mathbf{y}}||_2^2 = \sum_{n=1}^{N}(y^{(n)}-\hat{y}^{(n)})^2 = \sum_{n=1}^{N}(y^{(n)}-w_0-w_1x^{(n)})^2 = (\mathbf{y}-\hat{\mathbf{y}})^\top(\mathbf{y}-\hat{\mathbf{y}}) = \boldsymbol{\epsilon}^\top\boldsymbol{\epsilon} \tag{5}$$

Goal: find the best **w** that minimizes the SSE:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} J(\mathbf{w}) \tag{6}$$

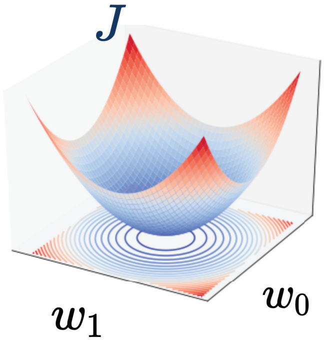
Given the error function is differentiable everywhere, the slope at the current value of w_1 projecting onto the error parabola is shown on the right.

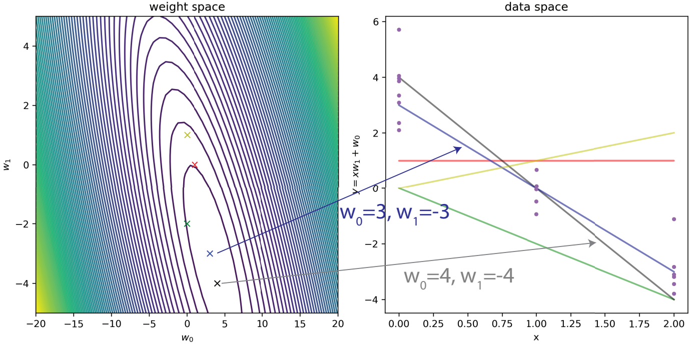
The SSE error function is **convex** (more details in Module 4.4). The optimal w_1^* is found where **the slope or the partial derivative is zero** $\frac{\partial J(\mathbf{w})}{\partial w_1^*} = 0$.



Contour plot visualizes the error surface as a function of w_1 and w_0

$$J(\mathbf{w}) = \sum_{n=1}^{N} (y^{(n)} - \hat{y}^{(n)})^2 = \sum_{n=1}^{N} (y^{(n)} - w_0 - w_1 x^{(n)})^2$$





w_0=3, w_1=-3

w_0=4, w_1=-4

Derivation of ordinary least squared (OLS) solution for w_0

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{\partial}{\partial w_0}\sum_{n=1}^{N}(y^{(n)} - w_0 - w_1 x^{(n)})^2$$

$$= \sum_{n=1}^{N} \frac{\partial (y^{(n)} - w_0 - w_1 x^{(n)})^2}{\partial (y^{(n)} - w_0 - w_1 x^{(n)})} \frac{\partial (y^{(n)} - w_0 - w_1 x^{(n)})}{\partial w_0}$$

$$= \sum_{n=1}^{N} 2(y^{(n)} - w_0 - w_1 x^{(n)})(-1)$$

Set $\frac{\partial J(w_0)}{\partial w_0}$ to zero and multiplying $\frac{1}{2}$ and -1 on both sides gives:

$$\sum_{n=1}^{N}(y^{(n)} - w_0 - w_1 x^{(n)}) = 0$$

Solving for w_0:

$$\sum_{n=1}^{N} y^{(n)} - \sum_{n=1}^{N} w_0 - \sum_{n=1}^{N} w_1 x^{(n)} = 0$$

$$\sum_{n=1}^{N} w_0 = \sum_{n=1}^{N} y^{(n)} - w_1 \sum_{n=1}^{N} x^{(n)}$$

$$N w_0 = \sum_{n=1}^{N} y^{(n)} - w_1 \sum_{n=1}^{N} x^{(n)}$$

$$w_0 = \frac{1}{N}\sum_{n=1}^{N} y^{(n)} - w_1 \frac{1}{N}\sum_{n=1}^{N} x^{(n)}$$

$$w_0 = \bar{y} - w_1 \bar{x}$$

Therefore, $\hat{w}_0 = \bar{y} - w_1 \bar{x}$

Derivation of OLS solution for w_1

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{\partial}{\partial w_1}\sum_{n=1}^{N}(y^{(n)} - w_0 - w_1 x^{(n)})^2$$

$$= \sum_{n=1}^{N} 2(y^{(n)} - w_0 - w_1 x^{(n)})(-x^{(n)})$$

Set $\frac{\partial J(\mathbf{w})}{\partial w_1}$ to zero and multiplying $\frac{1}{2}$ and -1 on both sides gives:

$$\sum_{n=1}^{N}(y^{(n)} - w_0 - w_1 x^{(n)})x^{(n)} = 0 \qquad (7)$$

Plug the bias estimate $\hat{w}_0$ in (7) and solve for w_1:

$$\sum_{n=1}^{N}(y^{(n)} - (\bar{y} - w_1\bar{x}) - w_1 x^{(n)})x^{(n)} = 0$$

$$\sum_{n=1}^{N}(y^{(n)} - \bar{y} + w_1\bar{x} - w_1 x^{(n)})x^{(n)} = 0$$

$$\sum_{n=1}^{N}(y^{(n)} - \bar{y})x^{(n)} - w_1\sum_{n=1}^{N}(x^{(n)} - \bar{x})x^{(n)} = 0$$

$$\hat{w}_1 = \frac{\sum_{n=1}^{N}(y^{(n)} - \bar{y})x^{(n)}}{\sum_{n=1}^{N}(x^{(n)} - \bar{x})x^{(n)}}$$

Derivation of OLS solution for w_1 (cont'd)

Note that

$$\sum_{n=1}^{N}(y^{(n)} - \bar{y})x^{(n)} = \sum_{n=1}^{N}(y^{(n)} - \bar{y})(x^{(n)} - \bar{x})$$

because:

$$\sum_{n=1}^{N}(y^{(n)} - \bar{y})(x^{(n)} - \bar{x}) = \sum_{n=1}^{N} y^{(n)}x^{(n)} - \sum_{n=1}^{N} y^{(n)}\bar{x} - \sum_{n=1}^{N} \bar{y}x^{(n)} + \sum_{n=1}^{N} \bar{y}\bar{x}$$

$$= \sum_{n=1}^{N} y^{(n)}x^{(n)} - N\bar{y}\bar{x} - \sum_{n=1}^{N} \bar{y}x^{(n)} + N\bar{y}\bar{x} = \sum_{n=1}^{N}(y^{(n)} - \bar{y})x^{(n)}$$

Similarly,

$$\sum_{n=1}^{N}(x^{(n)} - \bar{x})x^{(n)} = \sum_{n=1}^{N}(x^{(n)} - \bar{x})(x^{(n)} - \bar{x}) = \sum_{n=1}^{N}(x^{(n)} - \bar{x})^2$$

Update equations for simple linear regression

Therefore, the simple linear regression OLS solutions are:

$$\hat{w}_1 = \frac{\sum_{n=1}^{N}(y^{(n)} - \bar{y})x^{(n)}}{\sum_{n=1}^{N}(x^{(n)} - \bar{x})x^{(n)}} = \frac{\sum_{n=1}^{N}(y^{(n)} - \bar{y})(x^{(n)} - \bar{x})}{\sum_{n=1}^{N}(x^{(n)} - \bar{x})^2}; \quad \hat{w}_0 = \bar{y} - \hat{w}_1\bar{x} \tag{8}$$

Compare this solution with Pearson correlation coefficient (PCC) yields useful insights:

$$\hat{r}_{xy} = \frac{\sum_{n=1}^{N}(y^{(n)} - \bar{y})(x^{(n)} - \bar{x})}{\sqrt{\sum_{n=1}^{N}(x^{(n)} - \bar{x})^2}\sqrt{\sum_{n=1}^{N}(y^{(n)} - \bar{y})^2}} \in [-1, 1]$$

Therefore, we can see that

$$\hat{w}_1 = \hat{r}_{xy}\frac{\sqrt{\sum_{n=1}^{N}(y^{(n)} - \bar{y})^2}}{\sqrt{\sum_{n=1}^{N}(x^{(n)} - \bar{x})^2}}$$

If the standard deviation for y and x are the same (e.g., through standardization described next), $\hat{w}_1 = \hat{r}_{xy}$. However, in general, the regression slope and PCC are *not* the same: while $\hat{r}_{xy}$ gives a bounded measure independent of the scale of the two variables, $\hat{w}_1$ measures the change in the expected value of y corresponding to 1-unit increase/decrease of x.

Update equations for simple linear regression

A special case arises when **y** and **x** are *centered*: $\dot{y}^{(n)} = y^{(n)} - \bar{y}$ and $\dot{x}^{(n)} = x^{(n)} - \bar{x}$, such that $\bar{\dot{y}} = 0$ and $\bar{\dot{x}} = 0$ because

$$\bar{\dot{y}} = \frac{1}{N}\sum_n \dot{y}^{(n)} = \frac{1}{N}\sum_n (y^{(n)} - \bar{y}) = \frac{1}{N}\sum_n y^{(n)} - \frac{1}{N}\sum_n \bar{y} = \bar{y} - \bar{y} = 0$$

As a result,

$$\hat{w}_1 = \frac{\sum_{n=1}^N (\dot{y}^{(n)} - \bar{\dot{y}})(\dot{x}^{(n)} - \bar{\dot{x}})}{\sum_{n=1}^N (\dot{x}^{(n)} - \bar{\dot{x}})^2} = \frac{\sum_{n=1}^N \dot{y}^{(n)}\dot{x}^{(n)}}{\sum_{n=1}^N (\dot{x}^{(n)})^2}; \quad \hat{w}_0 = \bar{\dot{y}} - \hat{w}_1\bar{\dot{x}} = 0$$

Without the intercept, the linear function becomes $\hat{\dot{y}}^{(n)} = w_1\dot{x}^{(n)}$

If we only center the input variable such that $\bar{\dot{x}} = 0$ but not y, then $w_0 = \bar{y}$ and the linear function $\hat{y}^{(n)} = w_1\dot{x}^{(n)} + \bar{y}$ measures the *change* from the average $\bar{y}$.

Also, PCC of centered x and y is the same as cosine similarity between x and y.

Standardization involves centering and scaling the input feature

Another special case arises if we further scale $\dot{x}$ by its standard deviation $\sigma_{\dot{x}} = \sqrt{\frac{1}{N}\sum_n (\dot{x}^{(n)})^2}$:

$$\tilde{\dot{x}}^{(n)} = \dot{x}^{(n)}/\sigma_{\dot{x}}$$

then we have

$$\sigma^2_{\tilde{\dot{x}}} = \frac{1}{N}\sum_n (\tilde{\dot{x}}^{(n)})^2 = \frac{1}{N}\sum_n (\dot{x}^{(n)}/\sigma_{\dot{x}})^2 = \frac{1}{N}\sum_n (\dot{x}^{(n)})^2/\sigma^2_{\dot{x}} = \sigma^2_{\dot{x}}/\sigma^2_{\dot{x}} = 1$$

The regression coefficient becomes more simplified (while $w_0 = 0$ after centering x, y):

$$\hat{w}_1 = \frac{\sum_{n=1}^N \tilde{\dot{y}}^{(n)}\tilde{\dot{x}}^{(n)}}{\sum_{n=1}^N (\tilde{\dot{x}}^{(n)})^2} = \frac{1}{N}\sum_{n=1}^N \tilde{\dot{y}}^{(n)}\tilde{\dot{x}}^{(n)} = \frac{1}{N}\tilde{\dot{\mathbf{x}}}^\top\tilde{\dot{\mathbf{y}}}$$

Together, the procedure of centering and scaling of the response and/or the input features is common practice known as **standardization** mainly to simplify computation and to make the model robust to different numerical scales of the input and response.

Outline

Objectives

Simple linear regression

Multiple linear regression

Probabilistic interpretation

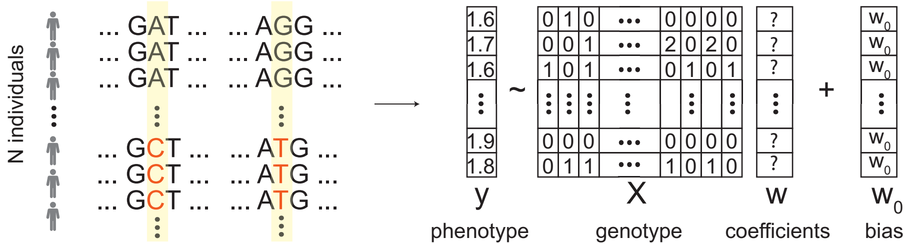
Non-linear basis function

Summary

Multiple linear regression

- Goal: learn how to fit a *multiple regression* to predict the outcome variable y using *multiple* input features
- We can write the response as a weighted linear sum of the input features:

$$f(\mathbf{x}^{(n)}; \mathbf{w}) = w_0 + w_1 x_1^{(n)} + w_2 x_2^{(n)} + \ldots + w_D x_D^{(n)} = w_0 + \sum_{d=1}^{D} w_d x_d^{(n)}$$



$$\begin{bmatrix} 1.6 \\ 1.7 \\ 1.6 \\ \vdots \\ 1.9 \\ 1.8 \end{bmatrix} \sim \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 2 & 0 & 2 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & \cdots & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} ? \\ ? \\ ? \\ \vdots \\ ? \\ ? \end{bmatrix} + \begin{bmatrix} w_0 \\ w_0 \\ w_0 \\ \vdots \\ w_0 \\ w_0 \end{bmatrix}$$

Multiple regression in matrix form

Suppose N training examples and D features. The data matrices are:

- Response: $\mathbf{y} \in \mathbb{R}^{N \times 1}$
- Input feature matrix: $[\mathbf{1}, \mathbf{X}] \in \mathbb{R}^{N \times (D+1)}$
- Regression coefficients: $[w_0; \mathbf{w}] \in \mathbb{R}^{(D+1) \times 1}$

We can rewrite the multiple regression function as:

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \ldots & x_D^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \ldots & x_D^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \ldots & x_D^{(N)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix} + \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(N)} \end{bmatrix}$$

$$\mathbf{y} = \mathbf{1} w_0 + \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

Our goal is to find the ordinary least square (OLS) solution for the coefficients $\mathbf{w}$:

$$w_0^*, \mathbf{w}^* \leftarrow \arg\min_{w_0, \mathbf{w}} ||\mathbf{y} - \mathbf{X}\mathbf{w} - w_0||_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0)^\intercal (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0)$$

Notation: $||\mathbf{a}||_2 = \sqrt{\sum_i a_i^2}$ is L_2-norm and $||\mathbf{a}||_2^2 = \sum_i a_i^2$ is the squared of the L2-norm.

Multiple regression OLS derivation for the bias term w_0

Let the loss function be $J(\mathbf{w}, w_0) = (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0)^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w} - w_0)$.

$$
\begin{aligned}
\frac{\partial J(\mathbf{w})}{\partial w_0} &= \frac{\partial}{\partial w_0}\left((\mathbf{y} - \mathbf{X}\mathbf{w}) - \mathbf{1}w_0\right)^\mathsf{T}\left((\mathbf{y} - \mathbf{X}\mathbf{w}) - \mathbf{1}w_0\right) \\
&= \frac{\partial}{\partial w_0}\left\{(\mathbf{y} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}) - (\mathbf{y} - \mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{1}w_0 - w_0\mathbf{1}^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}) + \mathbf{1}^\mathsf{T}\mathbf{1}w_0^2\right\} \\
&= \frac{\partial}{\partial w_0}(\mathbf{y} - \mathbf{X}\mathbf{w})^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\partial}{\partial w_0}2w_0\mathbf{1}^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{\partial}{\partial w_0}Nw_0^2 \\
&= 0 - 2\mathbf{1}^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}) + 2Nw_0 \\
&= -2\mathbf{1}^\mathsf{T}\mathbf{y} + 2\mathbf{1}^\mathsf{T}\mathbf{X}\mathbf{w} + 2Nw_0 \\
&= -2\sum_n y^{(n)} + 2\Big(\sum_n \mathbf{x}^{(n)}\Big)\mathbf{w} + 2Nw_0 \overset{\text{set}}{=} 0
\end{aligned}
$$

Solving $\frac{\partial J(\mathbf{w})}{\partial w_0} = 0$ for w_0:

$$
\hat{w}_0 = \frac{1}{N}\sum_n y^{(n)} - \frac{1}{N}\Big(\sum_n \mathbf{x}^{(n)}\Big)\mathbf{w} \equiv \bar{y} - \bar{\mathbf{x}}\mathbf{w}
$$

Plugging in the OLS estimate for the bias term w_0 into the loss

$$\begin{aligned} J(\mathbf{w}, \hat{w}_0) &= ||\mathbf{y} - \mathbf{X}\mathbf{w} - \mathbf{1}\hat{w}_0||_2^2 \\ &= ||\mathbf{y} - \mathbf{X}\mathbf{w} - \mathbf{1}(\bar{y} - \bar{\mathbf{x}}\mathbf{w})||_2^2 \\ &= ||(\mathbf{y} - \mathbf{1}\bar{y}) - (\mathbf{X}\mathbf{w} - \mathbf{1}\bar{\mathbf{x}}\mathbf{w})||_2^2 \\ &= ||(\mathbf{y} - \mathbf{1}\bar{y}) - (\mathbf{X} - \mathbf{1}\bar{\mathbf{x}})\mathbf{w}||_2^2 \\ &\equiv ||\dot{\mathbf{y}} - \dot{\mathbf{X}}\mathbf{w}||_2^2 \end{aligned}$$

where both the response and input features are mean-centered such that for each example n:

$$\begin{aligned} \dot{y}^{(n)} &= y^{(n)} - \bar{y} \\ \dot{\mathbf{x}}^{(n)} &= \mathbf{x}^{(n)} - \bar{\mathbf{x}} \end{aligned}$$

Note: the second equation is element-wise subtraction: for each feature d,

$$\dot{x}_d^{(n)} = x_d^{(n)} - \bar{x}_d$$

Solving all multiple regression coefficients simultaneously

Here we can assume that either the data are centered (i.e., $\mathbf{y} = \dot{\mathbf{y}}, \mathbf{X} = \dot{\mathbf{X}}$) or we have $\mathbf{x}_0 = \mathbf{1}$. In the latter case, the estimate for w_0 is the first element in $\hat{\mathbf{w}}$.

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}(\mathbf{y} - \mathbf{X}\mathbf{w})^\intercal(\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{\partial}{\partial \mathbf{w}}(\mathbf{y}^\intercal\mathbf{y} \underbrace{-\mathbf{y}^\intercal\mathbf{X}\mathbf{w} - \mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y}}_{2\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y}} + \mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w})$$

$$= \frac{\partial}{\partial \mathbf{w}}(\mathbf{y}^\intercal\mathbf{y} - 2\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y} + \mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w}) = \underbrace{\frac{\partial}{\partial \mathbf{w}}\mathbf{y}^\intercal\mathbf{y}}_{0} - 2\frac{\partial}{\partial \mathbf{w}}\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y} + \frac{\partial}{\partial \mathbf{w}}\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w}$$

$$\overset{\dagger}{=} -2\mathbf{X}^\intercal\mathbf{y} + 2\mathbf{X}^\intercal\mathbf{X}\mathbf{w} = 2\mathbf{X}^\intercal(\mathbf{X}\mathbf{w} - \mathbf{y}) = 2\mathbf{X}^\intercal(\hat{\mathbf{y}} - \mathbf{y})$$

†To get this equality, we make use of two general properties in matrix differentiation[1]:

$$\frac{\partial \mathbf{b}^\intercal\mathbf{a}}{\partial \mathbf{b}} = \mathbf{a}, \qquad \frac{\partial \mathbf{b}^\intercal\mathbf{A}\mathbf{b}}{\partial \mathbf{b}} = \mathbf{2Ab}$$

Setting the derivative to zero and solve for $\mathbf{w}$ gives the closed-form solution:

$$0 = -2\mathbf{X}^\intercal\mathbf{y} + 2\mathbf{X}^\intercal\mathbf{X}\mathbf{w} \quad \rightarrow \quad \mathbf{X}^\intercal\mathbf{X}\mathbf{w} = \mathbf{X}^\intercal\mathbf{y} \quad \rightarrow \quad \mathbf{w} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}$$

[1] Matrix cookbook Section 2.4 Eq 69 & Eq 85

Uniqueness of the OLS solution

$$\mathbf{w}^* = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y} \tag{9}$$

- The OLS solution is available when the $D \times D$ square matrix $\mathbf{A} = \mathbf{X}^\intercal \mathbf{X}$ is invertible.
- Matrix inverse $\mathbf{A}^{-1}$ can be computed by eigendecomposition:

  $$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \implies \mathbf{A}^{-1} = \mathbf{Q}^{-1}\mathbf{\Lambda}^{-1}\mathbf{Q}$$

  where
  - $\mathbf{Q}$ is the square $D \times D$ matrix, whose i^{th} column is the i^{th} eigenvector
  - $\mathbf{Q}$ is also an orthonormal matrix, which means $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I} \implies \mathbf{Q}^\top = \mathbf{Q}^{-1}$
  - Each eigenvector $\mathbf{u}$ in $\mathbf{Q}$ is orthonormal to itself: $\mathbf{u}^\top \mathbf{u} = 1$ and two different eigenvectors are orthogonal to each other: $\mathbf{u}^\top \mathbf{v} = 0$
  - $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eignevalues, i.e., $\Lambda_{i,i} = \lambda_i$
- Therefore, $\mathbf{A}$ is not invertible if some of its eigenvalues are zeros, which can happen when two features are perfectly correlated, e.g., $\mathbf{x}_2 = 1 - \mathbf{x}_1$, meaning that the number of linearly independent columns (i.e., rank) is smaller than D.

Time complexity

$$\mathbf{w}^* = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}$$

- The inner product $\mathbf{A} = \mathbf{X}^\intercal\mathbf{X}$ takes $O(ND^2)$
- The inversion of the $D \times D$ matrix $\mathbf{A}^{-1}$ takes $O(D^3)$
- Computing $\mathbf{X}^\intercal\mathbf{y}$ takes $O(ND)$

Therefore, the total time complexity is $O(ND^2 + D^3)$.

This can be very expensive or infeasible for large D (e.g., 1 million SNPs or 20,000 genes) and/or large N (e.g., half million individuals). For this reason, although we can derive closed-form solution, *Stochastic Gradient Descent* (SGD) is used for large dataset (Module 4.4).

Multivariate regression

We can also adapt the equation for multiple response variables. Instead of a response vector $\mathbf{y} \in \mathbb{R}^N$, we have a response matrix $\mathbf{Y} \in \mathbb{R}^{N \times C}$ for C response variables. The multivariate regression function is:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \tag{10}$$

where $\mathbf{W} \in \mathbb{R}^{D \times C}$.
The OLS solution for $\mathbf{W}$ is then:

$$\mathbf{W} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{Y} \tag{11}$$

Note here the OLS coefficient $\mathbf{w}_k$ for each response variable k is computed *independently* in the above solution. Therefore, the resulting OLS $\mathbf{W}$ is identical to fitting each $D \times 1$ regression coefficient $\mathbf{w}_k$ separately and then concatenate the C vectors together to form the matrix $\mathbf{W}$.

Outline

Objectives

Simple linear regression

Multiple linear regression

Probabilistic interpretation

Non-linear basis function

Summary

Probabilistic interpretation: Gaussian response variable

The general multivariate normal (MVN) distribution is:

$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \Sigma) = \det(2\pi\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\intercal \Sigma^{-1}(\mathbf{y}-\mathbf{X}\mathbf{w})\right) \tag{12}$$

where Σ is a $N \times N$ covariance matrix between the N samples. If we assume the samples are *i.i.d.*, then Σ is a diagonal matrix $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is an identity matrix:

$$\Sigma \overset{i.i.d.}{=} \begin{bmatrix} \sigma^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma^2 & 0 & \dots & 0 \\ 0 & 0 & \sigma^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma^2 \end{bmatrix} = \sigma^2\mathbf{I}, \quad \Sigma^{-1} = \sigma^{-2}\mathbf{I}, \quad \det(2\pi\Sigma)^{-1/2} = (2\pi\sigma^2)^{-N/2}$$

where $\det(\mathbf{A})$ is the determinant of a square matrix.

The MVN can then be simplified as the product of individual Gaussians:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) &= (2\pi\sigma^2)^{-N/2} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\intercal(\mathbf{y}-\mathbf{X}\mathbf{w})\right) \\ &= (2\pi\sigma^2)^{-N/2} \exp\left(-\frac{1}{2\sigma^2}\sum_n (y^{(n)}-\mathbf{x}^{(n)}\mathbf{w})^2\right) \end{aligned}$$

Taking the logarithm of the likelihood, we have

$$\begin{aligned} \ln p(\mathbf{y}|\mathbf{X}) &= \underbrace{-\frac{N}{2}\ln(2\pi\sigma^2)}_{\text{constant w.r.t. } \mathbf{w}} - \sum_n \left(\frac{1}{2\sigma^2}(y^{(n)}-\mathbf{x}^{(n)}\mathbf{w})^2\right) \\ &\propto -\frac{1}{2\sigma^2}\sum_n (y^{(n)}-\mathbf{x}^{(n)}\mathbf{w})^2 \\ &= -\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 \end{aligned}$$

The last equation indicates that the log likelihood is proportional to the negative SSE.

Maximum likelihood estimation

Given that,

$$\ln p(\mathbf{y}|\mathbf{X}) \propto -\frac{1}{2\sigma^2}\sum_n (y^{(n)} - \mathbf{x}^{(n)}\mathbf{w})^2 = -\frac{1}{2\sigma^2}J(\mathbf{w})$$

Since σ^2 is a constant, minimizing SSE w.r.t. **w** is equivalent to maximizing the Gaussian likelihood w.r.t. **w**:

$$\arg\min_{\mathbf{w}} J(\mathbf{w}) = \arg\max_{\mathbf{w}} \ln p(\mathbf{y}|\mathbf{X})$$

The maximum likelihood estimator for **w** is then identical to the OLS **w**:

$$\mathbf{w}^* = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}$$

Outline

Objectives

Simple linear regression

Multiple linear regression

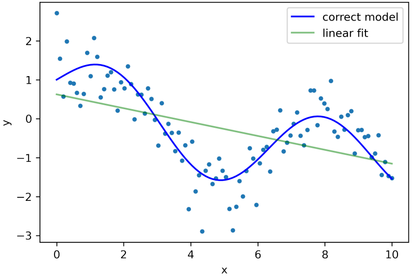
Probabilistic interpretation

Non-linear basis function

Summary

Fitting non-linear data by transforming the input features

Consider the toy dataset below. It is obvious that our attempt to model y as a linear function of $\hat{y} = w_0 + xw_1$ would produce a bad fit.



Idea: we can create more features using the given features. For example, we can use a M-degree polynomial function and treat each power degree as a standalone feature:

$$\hat{y} = x^0 w_0 + x^1 w_1 + x^2 w_2 + \ldots + x^M w_M = \sum_{m=0}^{M} x^m w_m$$

Fitting non-linear data by transforming the input features

In general, we can transform input feature x with a (non-linear) **basis function** $\phi(x)$. The multiple linear regression operates on the basis-transformed features:

$$\hat{y} = \sum_{m=0}^{M} \phi_m(x) w_m = \Phi(x)\mathbf{w} \tag{13}$$

We then simply replace all of the occurrences of x with $\Phi(x)$ in the OLS solution:

$$\hat{\mathbf{w}} = (\Phi(x)^\intercal \Phi(x))^{-1} \Phi(x)^\intercal \mathbf{y}, \quad \text{where}$$

$$\Phi(\mathbf{x}) = \begin{bmatrix} \phi_0(x^{(1)}) & \phi_1(x^{(1)}) & \dots & \phi_M(x^{(1)}) \\ \phi_0(x^{(2)}) & \phi_1(x^{(2)}) & \dots & \phi_M(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x^{(N)}) & \phi_1(x^{(N)}) & \dots & \phi_M(x^{(N)}) \end{bmatrix}$$

Transforming high-dimensional features

- This can also be done on high-dimensional input feature $\mathbf{x}^{(n)}$ by transforming each feature with $x_d^{(n)}$ with say M-degree polynomial:

$$\Phi(x_d^{(n)}) = [(x_d^{(n)})^1, \ldots, (x_d^{(n)})^M]$$

- We can then concatenate all transformed $D \times M$ features

$$\mathbf{x}^{(n)} = [(x_1^{(n)})^1, \ldots, (x_1^{(n)})^M, \ldots, (x_D^{(n)})^1, \ldots, (x_D^{(n)})^M] = [\Phi(x_1^{(n)}), \ldots, \Phi(x_D^{(n)})]$$

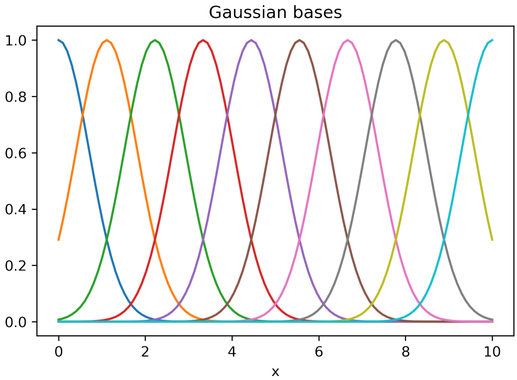
- This create a input matrix of dimension $N \times (D \times M)$:

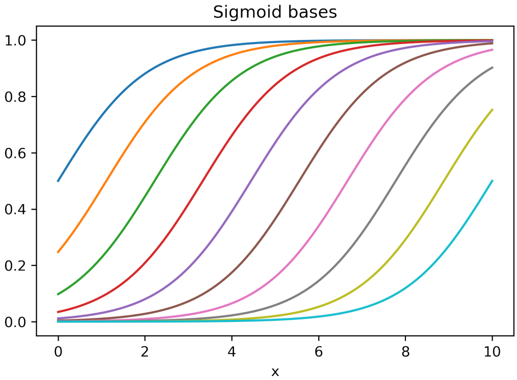
$$\Phi(\mathbf{X}) = \begin{bmatrix} \Phi(x_1^{(1)}) & \Phi(x_2^{(1)}) & \ldots & \Phi(x_D^{(1)}) \\ \Phi(x_1^{(2)}) & \Phi(x_2^{(2)}) & \ldots & \Phi(x_D^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(x_1^{(N)}) & \Phi(x_2^{(N)}) & \ldots & \Phi(x_D^{(N)}) \end{bmatrix}$$

- Regression: $\hat{y} = w_0 + \Phi(\mathbf{X})\mathbf{w}$

Nonlinear basis functions

There are many nonlinear basis functions. Using scalar input $x \in \mathbb{R}$ as an example,



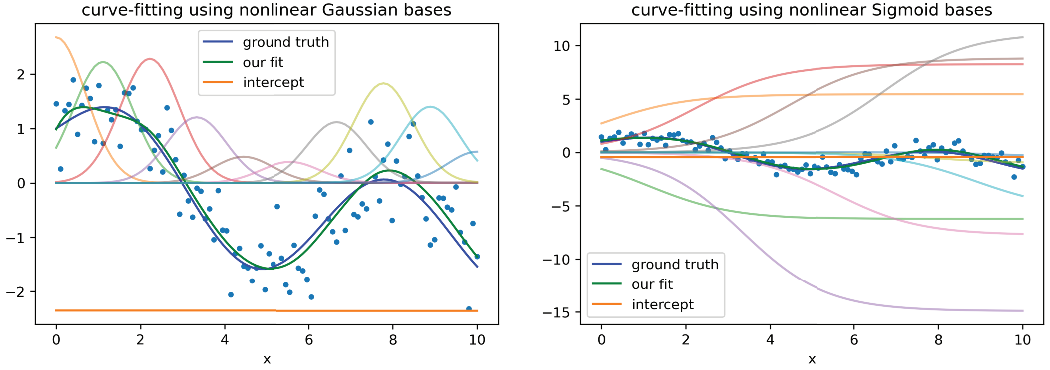


$$\phi_d(x) = \exp\left(-\frac{(x-\mu_d)^2}{2s^2}\right)$$

$$\phi_d(x) = \frac{1}{1+\exp\left(-\frac{(x-\mu_d)^2}{s}\right)}$$

where each type of basis function has a different mean $\mu_d \in [0, 10]$ and $s = 1$.

Linear regression with nonlinear basis (See Colab for the implementations)



In both plots, the green curve (our fit) is the sum of weighted Gaussian bases (i.e., the colorful curves) plus the intercept:

$$\hat{y} = w_0 + \sum_d w_d \phi_d(\mathbf{X})$$

Summary

- Response variable y is modelled as a weighted linear sum of the features $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$
- We fit the model by minimizing the sum of squared errors (SSE) $\sum_n (y^{(n)} - \mathbf{X}^{(n)}\mathbf{w})^2$
- OLS solution: $\mathbf{w}^* = (\mathbf{X}^\intercal \mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{y}$ with $O(ND^2 + D^3)$ time complexity
- Minimizing SSE is equivalent to maximizing the log Gaussian likelihood given that the data points are $i.i.d$.
- Using non-linear basis functions to construct features can help model non-linear data. However, these non-linear basis functions are rigid and non-learnable.
- In Module 5.1, we will discuss neural networks which have *learnable* non-linear basis functions.