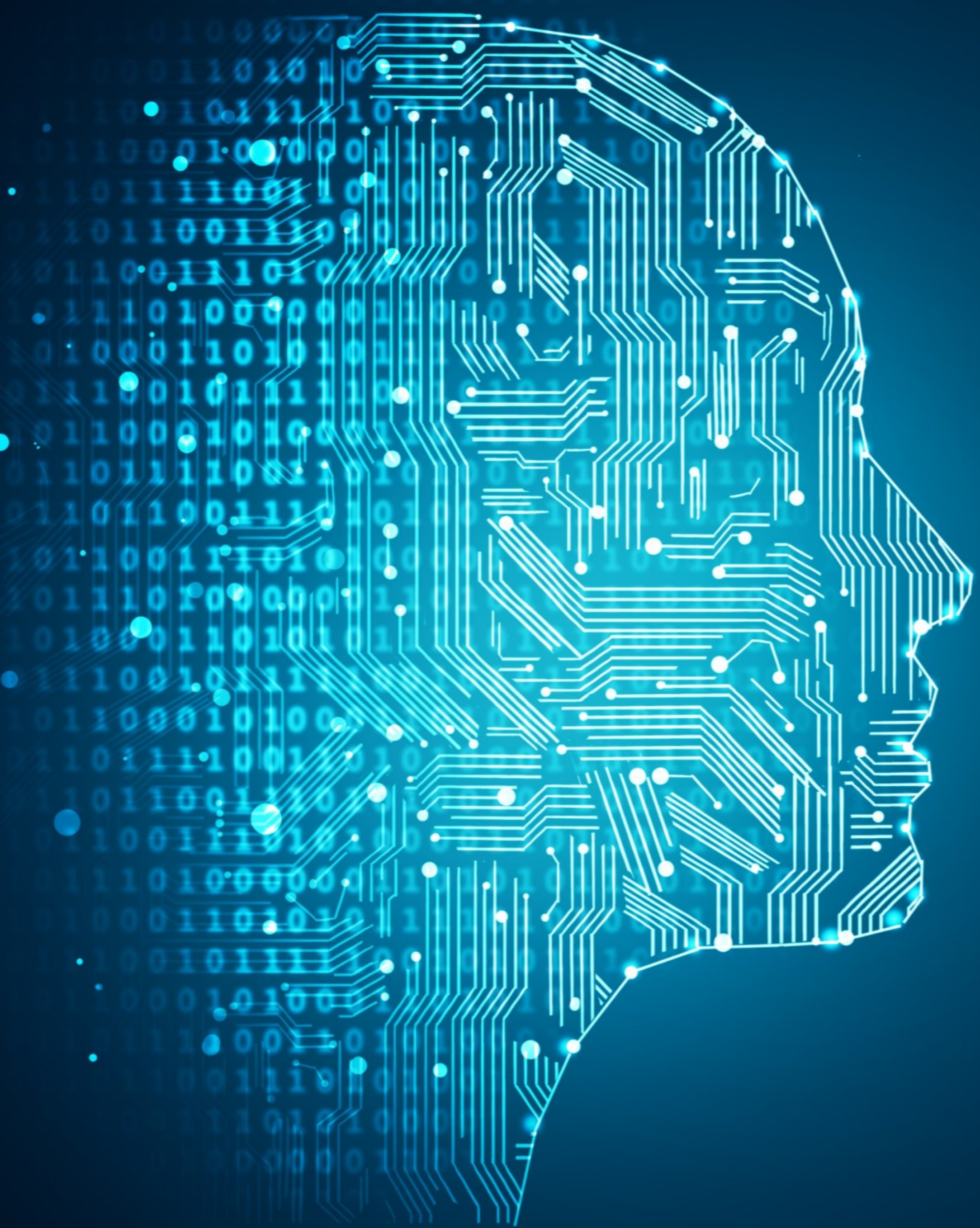


AI6103 Machine Learning Foundamentals

Boyang Li, Albert

School of Computer Science and Engineering

Nanyang Technological University

Outline of Lecture 3

- Information Theory
- Basics of Machine Learning
- Linear Regression
- Ridge Regression
- Logistic Regression

Information Theory

- The scientific study of the quantification, storage, and communication of digital information.
- How can we communicate through a noisy channel?
- How can we encode information into binary form efficiently?

- We will only scrape the surface of this vast and rich subject

Entropy

- The degree of uncertainty or “chaos / surprise / information” in a random variable
- Expectation of negative log probability

$$H(P(X)) = E[-\log X] = -\int P(x) \log P(x)\, dx$$

- For discrete variables

$$H(P(X)) = -\sum_i P(X = x_i)\ \log P(X = x_i)$$

- Example: Fair die with probabilities {1/6, 1/6, 1/6, 1/6, 1/6, 1/6}

$$H = -\left(\frac{1}{6}\log\frac{1}{6}\right) \times 6 = 0.78$$

- Bias die with probabilities {1/12, 1/12, 1/12, 1/12, 1/3, 1/3}

$$H = -\left(\frac{1}{12}\log\frac{1}{12}\right) \times 4 - \left(\frac{1}{3}\log\frac{1}{3}\right) \times 2 = 0.67$$

More uncertainty when the distribution is closer to uniform.

Entropy: Relation to Event Encoding

- If we observe 26 letters with equal probability, we can use $\log_2 26 = -\log_2 \frac{1}{26}$ bits to encode each character.
- No fractional bits, so $\lceil \log_2 26 \rceil = 5$
- A = 00000, B = 00001, C=00010, D=00011, etc.
- To encode three letters, we need 15 bits.

Entropy: Relation to Event Encoding

- However, if one letter is more common than others, we can design the encoding such that we use fewer bits for more frequent letters.
- A = 001, B=0001, C=10100, D=10011, etc.
- BAD=000100110011 (12 bits)
- Using fewer bits in expectation because less frequent letters have longer encoding.

Entropy: Relation to Event Encoding

- $-\log_2 P(A)$ is the "information content" of event A.
- It is the number of bits we need to tell people that this event happened.
- Its expectation is the entropy.

$$H(P) = E[-\log X] = -\sum_i P(X = x_i)\ \log P(X = x_i)$$

Cross-Entropy

- For two probability distributions P and Q, the cross-entropy is

$$H(Q,P) = E_Q[-\log P(X)] = -\sum_i Q(X = x_i)\ \log P(X = x_i)$$

- Interpretation: We designed an encoding scheme for the probability distribution P. However, the actual distribution is Q. What is the number of bits we need to encode the information?

Kullback–Leibler divergence (relative entropy)

- A measure for the differences between distributions

$$KL(P||Q) = E_P\left[\log\frac{P(X)}{Q(X)}\right] = \sum_i P(X = x_i)\ \log\frac{P(X = x_i)}{Q(X = x_i)}$$

- Continuous distributions with probability density functions P and Q

$$KL(P||Q) = \int P(x)\log\frac{P(x)}{Q(x)}dx$$

KL divergence and cross entropy

$$KL(P||Q) = E_P\left[\log\frac{P}{Q}\right] = \sum_i P(X = x_i)\ \log\frac{P(X = x_i)}{Q(X = x_i)}$$

$$H(Q, P) = -\sum_i Q(X = x_i)\ \log P(X = x_i)$$ Cross entropy here

$$= -\sum_i Q(X = x_i)\ \log P(X = x_i) + \sum_i Q(X = x_i)\ \log Q(X = x_i) - \sum_i Q(X = x_i)\ \log Q(X = x_i)$$

$$= \sum_i Q(X = x_i)\ \log\frac{Q(X = x_i)}{P(X = x_i)} + H(Q(X))$$

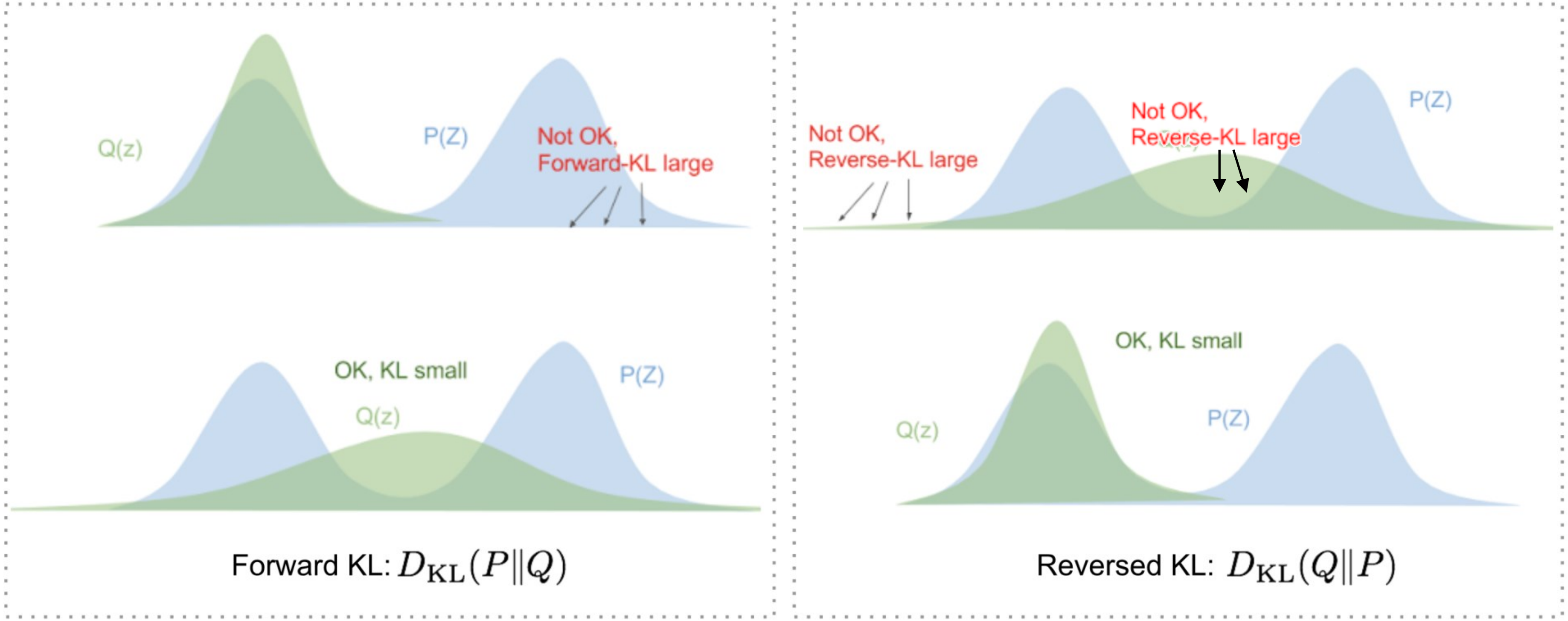
$$= H(Q) + KL(Q\ ||\ P)$$

KL divergence is asymmetric

$$KL(P \,||\, Q) = \sum_i P(X = x_i) \, \log \frac{P(X = x_i)}{Q(X = x_i)}$$

Forward KL:

- P is large when Q is small -> large divergence
- Q is large when P is small -> small divergence.



Jensen–Shannon divergence

- A symmetric measure for the differences between distributions

$$JS(Q,P) = \frac{1}{2}KL(Q,P) + \frac{1}{2}KL(P,Q)$$

Machine Learning Basics

Intelligence?

- Human intelligence has an innate aspect and an environmental aspect
  - Chimpanzees, dolphins, or parrots can demonstrate some levels of intelligence, but they can't reach the human level of intelligence even if training starts very early.
  - As humans, we observe and interact with the world for a long time and learn about knowledge accumulated over thousands of years

Machine Learning: Analogies

- The “innate” aspect is to specify a machine learning model, which defines the parameters that can be learned and the parameters that are determined before learning (“hyperparameters”).
- The “experiential” aspect is to learn the model parameters from data, a.k.a. training.

Machine Learning Basics

- There is a function $\mathrm{y} = f(x)$ that we want to approximate
  - x is the input to the machine learning model.
  - y is what the machine learning model tries to predict
- The exact function is unknown, but we have access to historic data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- Our goal is to find out this function from data

Machine Learning Basics

- There is a function $y = f(x)$ that we want to approximate
- Example: Automobile insurance risk
  - x = characteristics of the car, such as make, model, year, safety features, engine type, length, weight, height, fuel efficiency, etc.
  - y = probability of accident in a year, or average cost of repairs
- Example: Heart disease diagnosis
  - x = characteristics of the patient, such as age, sex, chest pain location, cholesterol level, blood sugar, etc.
  - y = medical diagnosis made by a human doctor

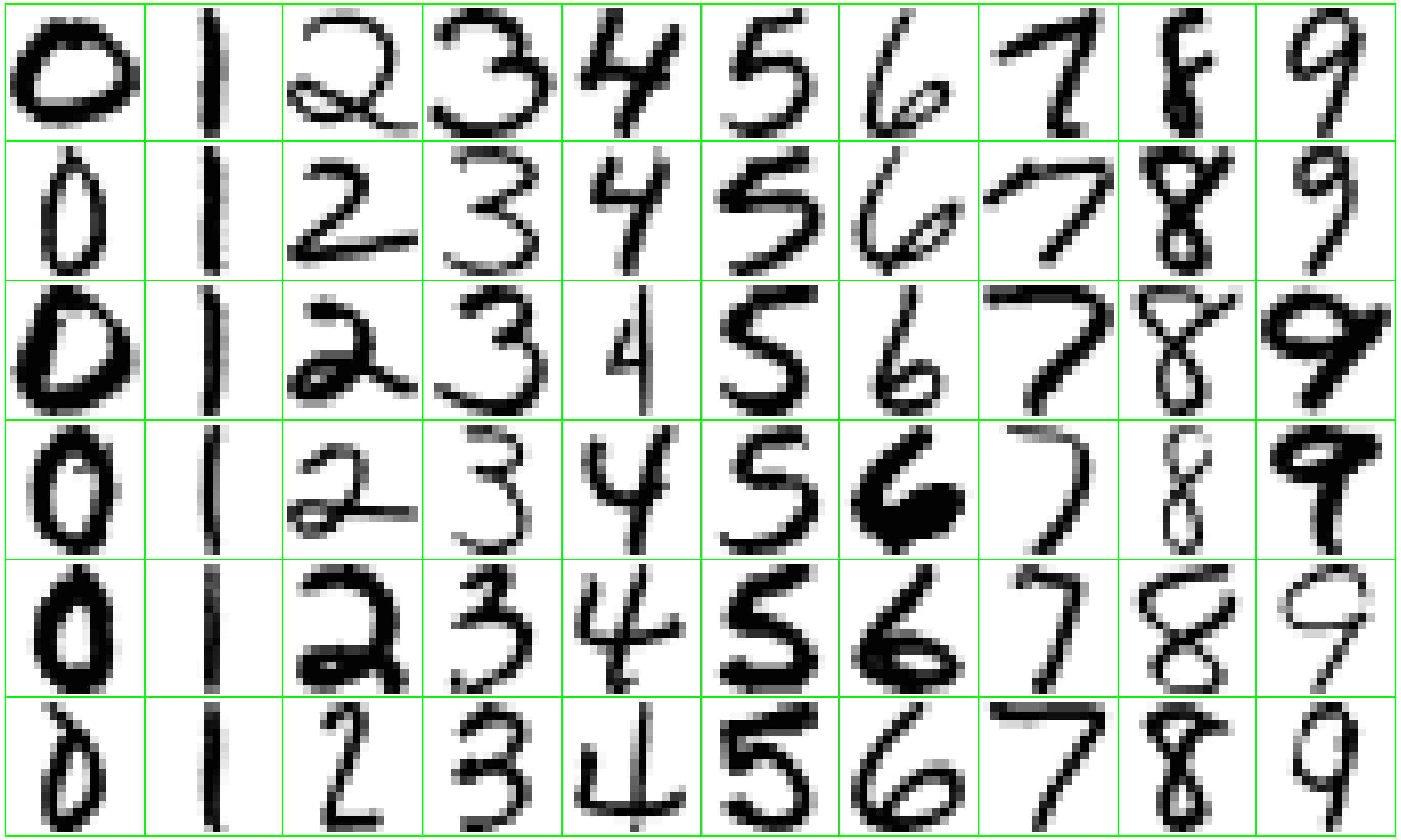
Machine Learning Basics

- There is a function $y = f(x)$ that we want to approximate
- Example: Image classification
  - x = image pixels
  - y = predefined classes, such as dog, cat, truck, airplane, apple, orange, etc.
- Example: Tweet emotion recognition
  - x = text of the tweet
  - y = human label of the reflected emotion: fear, anger, joy, sad, contempt, disgust, and surprise (Ekman's basic emotions).

Classification vs. Regression

- Classification: the output y is discrete and represents distinct categories
- Examples
  - Image categories: dog, cat, truck, airplane, apple, orange
  - Emotion categories: fear, anger, joy, sad, contempt, disgust, and surprise

MNIST Classification



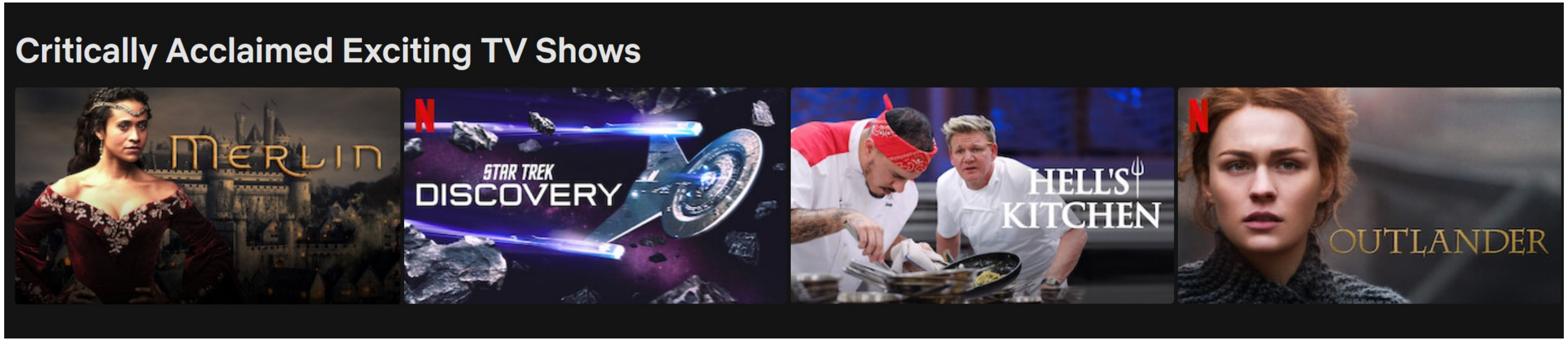
Classification for Skin Problems

Classification vs. Regression

- Regression: the output y represents a continuously varying quantity. Typically, a real number
- Examples
  - Stock price in a week
  - Heart disease: no disease (0), very mild (1), mild (2), severe (3), immediate danger (4)
- Key difference: measurement of error

Ranking

- Recommender systems provide a list of recommended items.



- We care about not only the first item, but the first N items.

Useful Features for Wine Quality Rating?

- The following features are probably causal factors to the quality of wine:
  - Alcohol content, pH, residual sugar, free sulfur dioxide, citric acid, tannin, color
- The shape of the bottle is probably unrelated to the quality.
- The year and the winery are probably correlated with the quality.
  - However, we may want to exclude these factors in order to avoid any bias from fame.
- Initial user response may be very good indicators.

Useful Features for Movie Recommendation?

- The following features are probably useful features for movie recommendation:
  - The list of movies that a user has seen in the past
  - The ratings that the user gave to these movies
  - The user's personal information, such as age, level of education, etc.

General Guidelines for Feature Selection

- Use features that are strongly correlated with the target variable, but they don’t have to be causal.
  - Rooster and sunrise
- Avoid features that you don’t want to model to consider, such as the year and the winery in wine quality regression.
  - Beware of information leaks
- Consider data collection, privacy, and ethical concerns

Linear Regression

- We have a p-dimensional feature vector $\boldsymbol{x} = (x_1, x_2, \dots, x_p)$ and a scalar output y
- Our model is linear

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \beta_0$$

- In vector form

$$\hat{y} = \boldsymbol{\beta}^\top \boldsymbol{x} + \beta_0$$

Linear Regression

- We have n number of data points

$$\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \dots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$$

- Assumption: Every data point follows the same model

$$\hat{y}^{(i)} = \boldsymbol{\beta}^{\top} \boldsymbol{x}^{(i)} + \beta_0$$

- Central question of machine learning

How do we find the parameters $\boldsymbol{\beta}$ and β_0?

One Small Tweak …

- Adding one dimension to $\boldsymbol{x}$,

$$\boldsymbol{x} = \left(x_1, x_2, \dots, x_p, 1\right)^\top$$

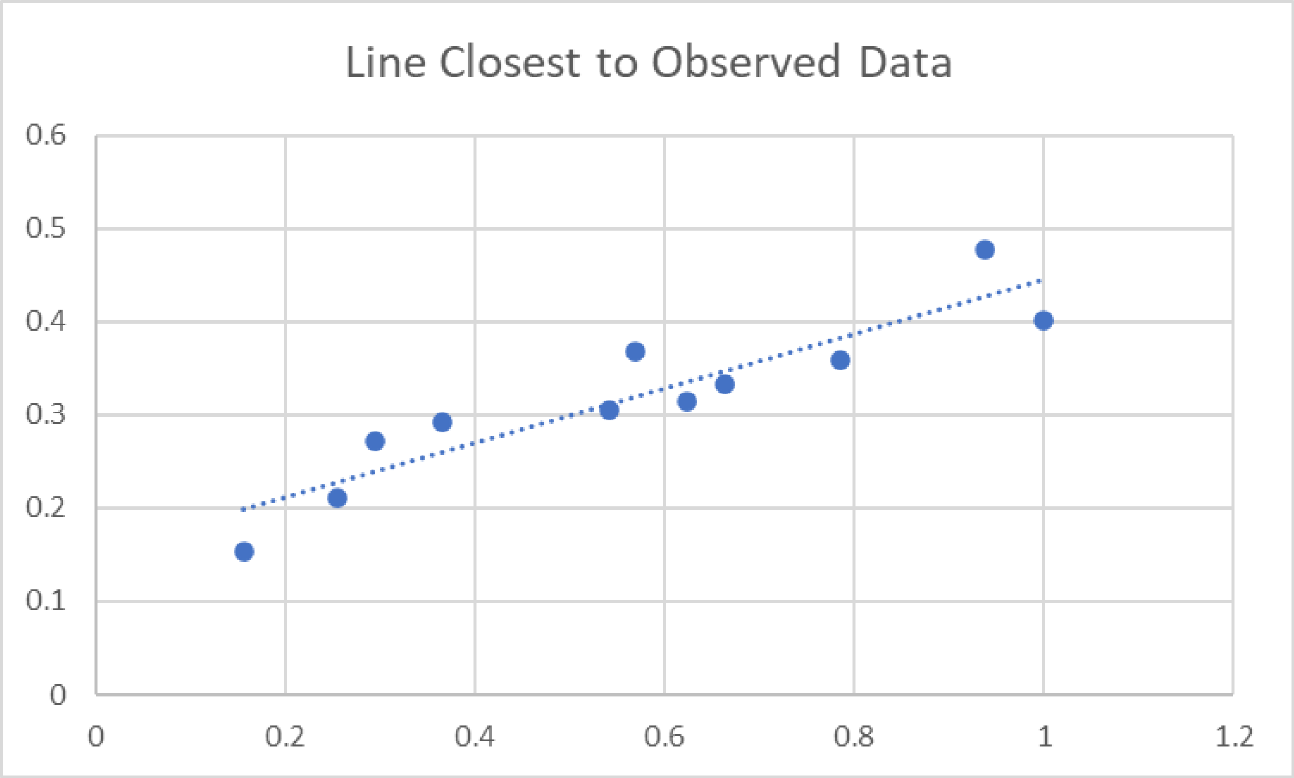
- Adding one dimension to $\boldsymbol{\beta}$,

$$\boldsymbol{\beta} = \left(\beta_1, \beta_2, \dots, \beta_p, \beta_0\right)^\top$$

- Our model becomes $\hat{y}^{(i)} = \boldsymbol{\beta}^\top \boldsymbol{x}^{(i)}$

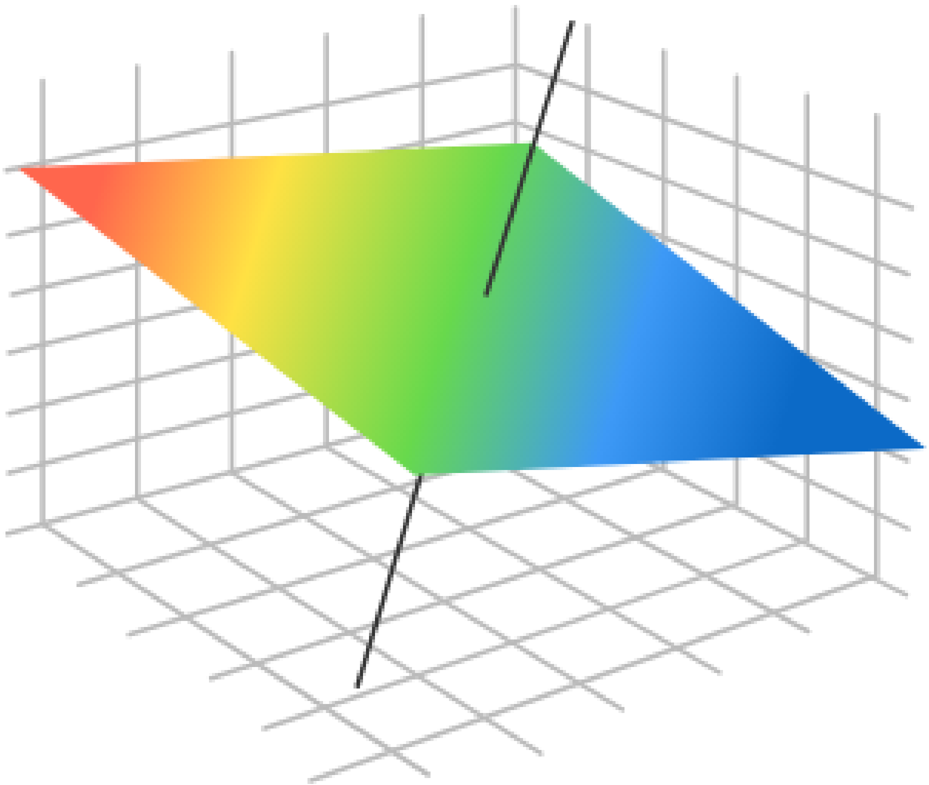
Geometric Intuition

- Find the straight line that is closest to observed data points



Geometric Intuition in 2D

- Find the 2D plane that is closest to observed data points



Higher dimensions?

- Can't visualize them because we live in a 3D world.
- Geometric intuition: Find the <u>hyperplane</u> that is closest to observed data points
- Key point: The function we fit is linear. A unit change in x_i always causes a change in $\hat{y}$ of the magnitude β_i, no matter the value of $\boldsymbol{x}$ or y.

The Loss Function

- We must define a measure of error.
- How wrong is our model?
- Mean Square Error: the average squared distance between $y^{(i)}$ and $\hat{y}^{(i)}$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{n}\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2$$

Linear Regression

- One central tenet of supervised learning

Find the model parameters that lead to smallest error on all possible data.

- We only observe limited amount of data, so it is usually taken as <u>minimizing error on training data while hoping to achieve low error on test data (data unseen during training)</u>
- When training data are too few or when they are not very representative, we need to use regularization

A Closed-form Solution

- In matrix form, the loss function is

$$\mathrm{MSE} = \frac{1}{n}(X\beta - y)^\top(X\beta - y)$$

- To find the minimum, we find the derivative against β

$$\frac{\partial \mathrm{MSE}}{\partial \beta} = \frac{2}{n}\left\{X^\top X\beta - X^\top y\right\}$$

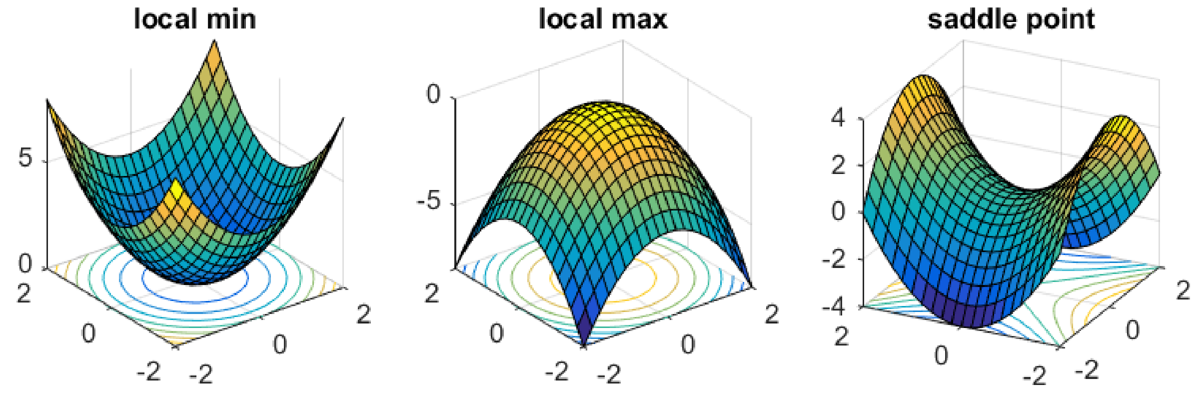
- Necessary condition for minimum: the derivative is zero
- Setting the above to zero and simplify, we get

$$\beta^* = (X^\top X)^{-1}X^\top y$$

Wait a minute …

$$\beta = (X^\top X)^{-1} X^\top Y$$

- Is this a local minimum? What about the second-order condition?



Second-order Conditions

- Consider the 1D function
$$y = ax^2 + bx + c$$
- When $a > 0$, it has a minimum, but no maximum
- When $a < 0$, it has a maximum, but no minimum
- When $a = 0$ and $b \neq 0$, it has neither a minimum nor a maximum

- For multidimensional functions, we consider the Hessian matrix.
- This is a symmetric matrix.

$$H = \begin{bmatrix} \frac{\partial^2 \mathrm{L}}{\partial \beta_1^2} & \frac{\partial^2 \mathrm{L}}{\partial \beta_1 \partial \beta_2} & \cdots & \frac{\partial^2 \mathrm{L}}{\partial \beta_1 \partial \beta_p} \\ \frac{\partial^2 \mathrm{L}}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 \mathrm{L}}{\partial \beta_2^2} & \cdots & \frac{\partial^2 \mathrm{L}}{\partial \beta_2 \partial \beta_p} \\ \vdots & \vdots & \cdots & \cdots \\ \frac{\partial^2 \mathrm{L}}{\partial \beta_p \partial \beta_1} & \frac{\partial^2 \mathrm{L}}{\partial \beta_p \partial \beta_2} & \cdots & \frac{\partial^2 \mathrm{L}}{\partial \beta_p^2} \end{bmatrix}$$

Linear Algebra: Positive Definiteness

- A square matrix A is symmetric iff $A_{ij} = A_{ji}, \forall i, j$

$$\begin{bmatrix} 1 & 3 & 7 \\ 3 & 2 & 6 \\ 7 & 6 & 8 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & -4 & 38 \\ 2 & -1 & 71 & 2 \\ -4 & 71 & 9 & 56 \\ 38 & 2 & 56 & 30 \end{bmatrix}$$

- A square matrix is anti-symmetric (or skew-symmetric) iff $A_{ij} = -A_{ji}, \forall i \neq j$

$$\begin{bmatrix} 1 & -3 & -7 \\ 3 & 2 & -6 \\ 7 & 6 & 8 \end{bmatrix}$$

- A matrix A is positive definite (PD) iff for all vector $x \neq 0$, $x^\top A x > 0$
- Recall $x^\top A x = \sum_i \sum_j A_{ij} x_i x_j$

$$x^\top \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} x = x_1^2 + x_2^2 + x_3^2 + (x_1 - x_2)^2 + (x_3 - x_2)^2 > 0, \forall x \neq 0$$

- Similarly,

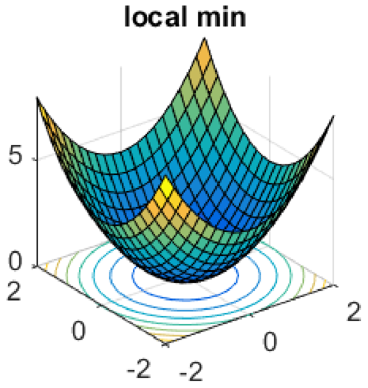
$$x^\top \begin{bmatrix} 2 & -2 & 0 \\ 0 & 3 & 0 \\ 0 & -2 & 2 \end{bmatrix} x = x_1^2 + x_2^2 + x_3^2 + (x_1 - x_2)^2 + (x_3 - x_2)^2 > 0, \forall x \neq 0$$

Linear Algebra: Positive Definiteness

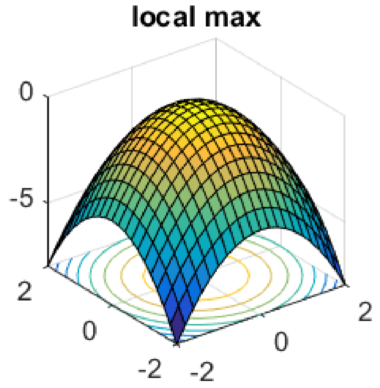
- A matrix A is positive definite (PD) iff for all vector $x \neq 0$, $x^\top Ax > 0$
- A matrix A is positive **semi-**definite (PSD) iff for all vector $x \neq 0$, $x^\top Ax \geq 0$

- A matrix A is negative definite (ND) iff for all vector $x \neq 0$, $x^\top Ax < 0$
- A matrix A is negative **semi-**definite (NSD) iff for all vector $x \neq 0$, $x^\top Ax \leq 0$

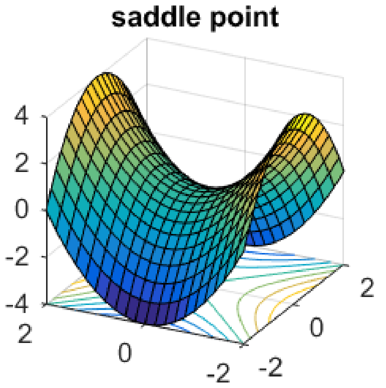
Second-order Conditions



Hessian is PSD



Hessian is NSD



Hessian is neither PSD nor NSD

$$\mathrm{L} = \frac{1}{n}(X\beta - y)^{\top}(X\beta - y)$$

$$\mathrm{H} = \frac{\partial^2 L}{\partial \beta^2} = \frac{2}{n} X^{\top} X$$

Conditions for Minimum

- The second-order derivative is $\frac{\partial^2 \text{MSE}}{\partial \beta^2} = \frac{2}{n} X^\top X$
- It is positive semi-definite because for an arbitrary vector $z \neq 0$
$$\mathrm{z}^\top X^\top X z = (Xz)^\top Xz$$
- Letting $a = Xz, a^\top a$ is always greater than or equal to zero
- So this is truly a local minimum.
- Since the loss function is convex (which we will not show), the local minimum is also the global minimum.

Solution to Linear Regression

$$\beta = (X^\top X)^{-1} X^\top Y$$

is the model parameter that minimizes the loss

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$$= \frac{1}{n}(X\beta - Y)^\top (X\beta - Y)$$

A.k.a. Ordinary Least Squares

Another detail: Is $X^\top X$ invertible?

$$\beta^* = (X^\top X)^{-1} X^\top y$$

- When the number of data points n is greater than the feature dimension p, and at least p data points are linearly independent, $X^\top X$ is invertible.
- When we have more features than data points $(p > n)$, we have a problem!
- This can be solved by regularization such as ridge regression.

Ridge Regression

- The ordinary least squares (OLS) estimator:

$$\hat{\beta}_{\mathrm{OLS}} = (X^\top X)^{-1} X^\top y$$

- If $n < p$, $X^\top X$ is not invertible. We can use ridge regression.

$$\hat{\beta}_{\mathrm{RR}} = (X^\top X + \lambda I)^{-1} X^\top y$$

- Here λ is a small positive number.
- We can show (but will not) that the ridge regression estimator for β is biased but has lower variance than the OLS estimator [bias-variance tradeoff]

Ridge Regression Is L2-regularized Linear Regression

- Ridge Regression can be understood as optimizing a different loss function.

$$\mathrm{L} = \frac{1}{n}(X\beta - y)^\top(X\beta - y) + \frac{1}{n}\lambda||\beta||^2$$

- We again take the derivative against β and set it to zero

$$\frac{\partial \mathrm{L}}{\partial \beta} = \frac{2}{n}\{X^\top X\beta - X^\top y + \lambda\beta\} = 0$$

$$\hat{\beta}_{\mathrm{RR}} = (X^\top X + \lambda I)^{-1}X^\top y$$

Recap: What did we do?

- Collected some data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \dots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- Specified a model $\hat{y}^{(i)} = \boldsymbol{\beta}^{\top}\boldsymbol{x}^{(i)}$
- Defined a loss function $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$
  - Or $\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \frac{1}{n}\lambda\boldsymbol{\beta}^{\top}\boldsymbol{\beta}$
- Found the parameters $\boldsymbol{\beta}$ that minimizes the loss function

Innate,
human design

Experiential,
data driven

Recap: Regularization

- The L2 regularization term $\boldsymbol{\beta}^\top \boldsymbol{\beta}$ reduces variance of the estimated $\boldsymbol{\beta}$.
- This is especially useful when we have limited data.
- We will see many other forms of regularization later.

Probabilistic Perspective

- The model is parameterized by $\boldsymbol{\beta}$ and takes input $\boldsymbol{x}$
- We write its output as $f_{\boldsymbol{\beta}}(\boldsymbol{x})$
- We interpret $f_{\boldsymbol{\beta}}(\boldsymbol{x})$ as the (input-dependent) parameter μ to a Gaussian distribution with unit standard deviation ($\sigma = 1$).
- The ground truth $y^{(i)}$ is drawn from this distribution

$$y^{(i)} \sim \mathcal{N}\left(f_{\boldsymbol{\beta}}\left(\boldsymbol{x}^{(i)}\right), 1\right)$$

- Equivalently

$$y^{(i)} = f_{\boldsymbol{\beta}}\left(\boldsymbol{x}^{(i)}\right) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 1)$$

Maximum Likelihood for Gaussian

- Data: $y^{(1)}, \ldots, y^{(N)}$
- The Gaussian probability is

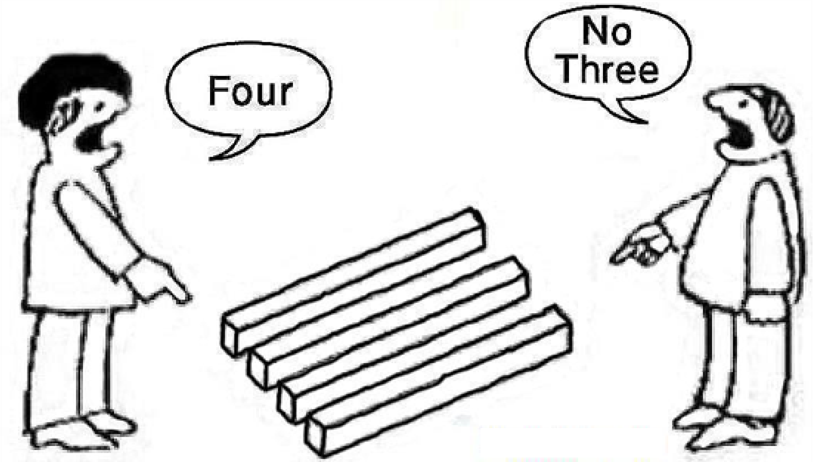
$$\prod_{i=1}^{N} P\left(y^{(i)} \middle| \mu, \sigma\right) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{\left(y^{(i)} - \mu\right)^2}{2\sigma^2}$$

- Taking log and remove anything unrelated to μ

$$\mu^* = \underset{\mu}{\operatorname{argmax}} \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{\left(y^{(i)} - \mu\right)^2}{2\sigma^2} = \underset{\mu}{\operatorname{argmax}} \sum_{i=1}^{N} - \frac{\left(y^{(i)} - \mu\right)^2}{2\sigma^2}$$

$$\mu^* = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y^{(i)} - \mu\right)^2$$

Plugging in …



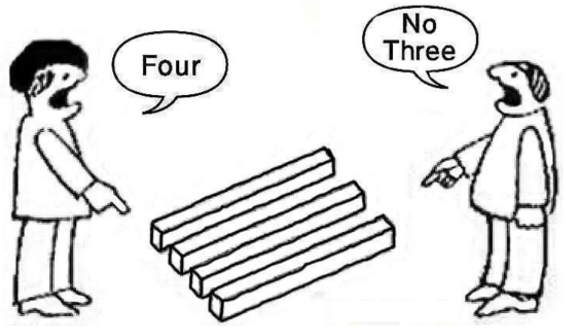
- $\mu^{(i)} = \hat{y}^{(i)} = f_{\boldsymbol{\beta}}(\boldsymbol{x})$

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y^{(i)} - f_{\boldsymbol{\beta}}\left(\boldsymbol{x}^{(i)}\right)\right)^2$$

- Linear regression can be understood as MLE if we assume the label contains noise from the Gaussian distribution.

$$y^{(i)} = f_{\boldsymbol{\beta}}\left(\boldsymbol{x}^{(i)}\right) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

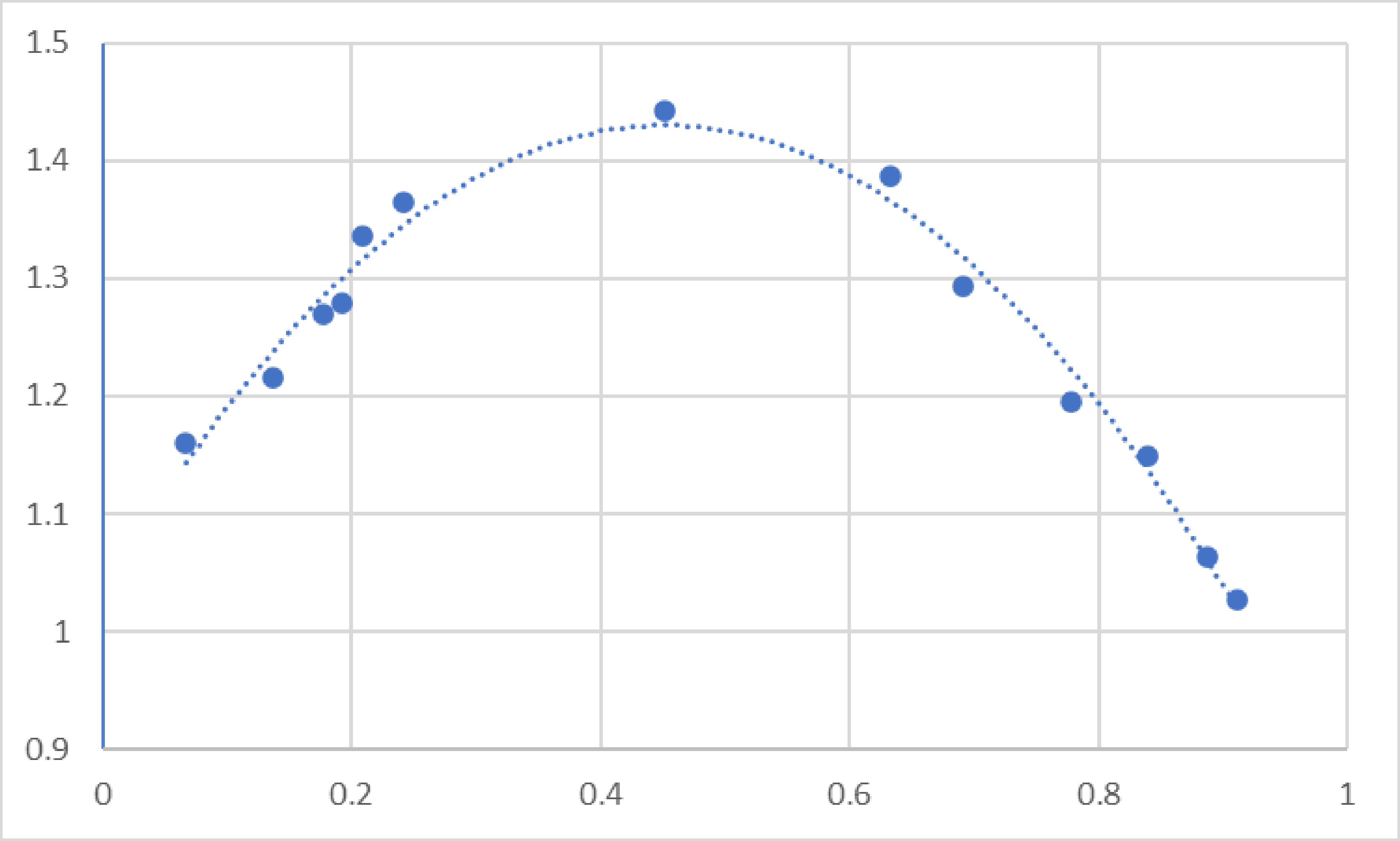
Ridge Regression [Optional]



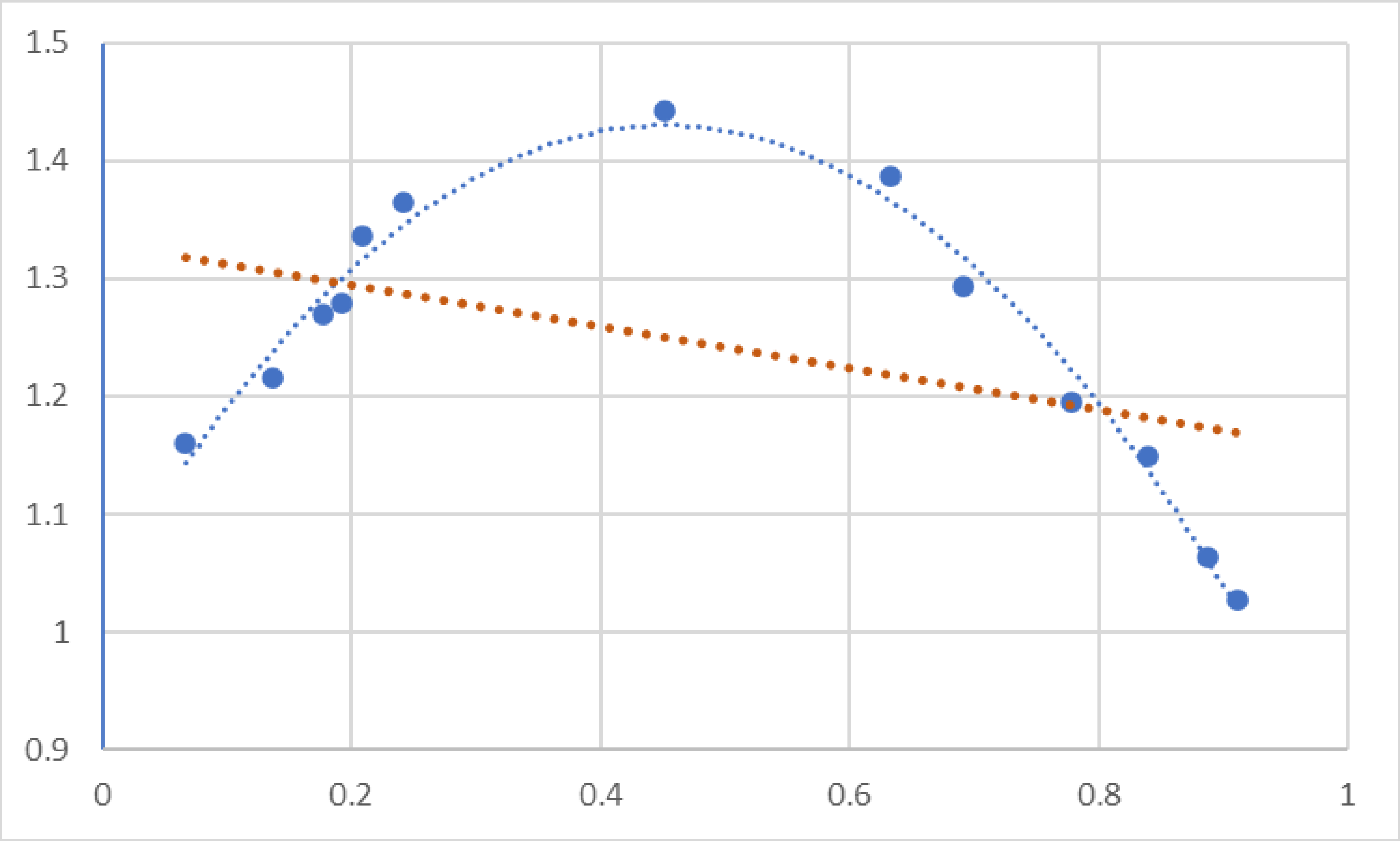
Ridge regression can be understood as Bayesian maximum a posteriori (MAP) estimation with a Gaussian prior $\mathcal{N}(0, \frac{1}{\lambda})$ for the model parameters $\boldsymbol{\beta}$.

We omit the details for the purpose of this course.

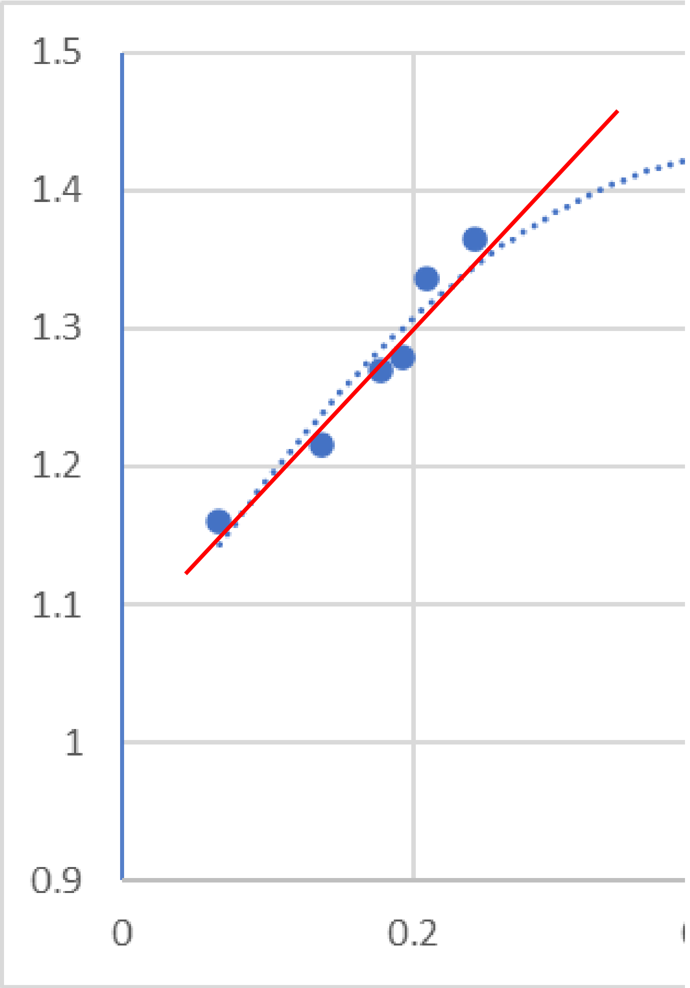
Linear Models are Limited



Linear Models are Limited



Linear Models are Limited



Non-Linear Models

Logistic Regression: a Single “Neuron”

- The simplest non-linear model.
- Sometimes referred to as “generalized linear model” as the decision boundary is still linear.
- Here we emphasize the fact that the model function has a non-linear form.

The Supervised Learning Recipe

- $\boxed{\text{Collect some data } \left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \dots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)}$
- Specify a model $\hat{y}^{(i)} = f\left(\boldsymbol{x}^{(i)}\right)$
- Define a loss function
- Find the parameters $\boldsymbol{\beta}$ that minimize the loss function

The Data

- Collect some data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- $\boldsymbol{x}^{(i)}$ is a p-vector.
- $y^{(i)}$ is either 0 or 1, denoting the two classes.

The Supervised Learning Recipe

- Collect some data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- $\boxed{\text{Specify a model } \hat{y}^{(i)} = f\left(\boldsymbol{x}^{(i)}\right)}$
- Define a loss function
- Find the parameters $\boldsymbol{\beta}$ that minimize the loss function

Logistic Regression: a Single "Neuron"

- Model

$$\hat{y}^{(i)} = \sigma\left(\boldsymbol{\beta}^\top \boldsymbol{x}^{(i)}\right)$$

- Activation function

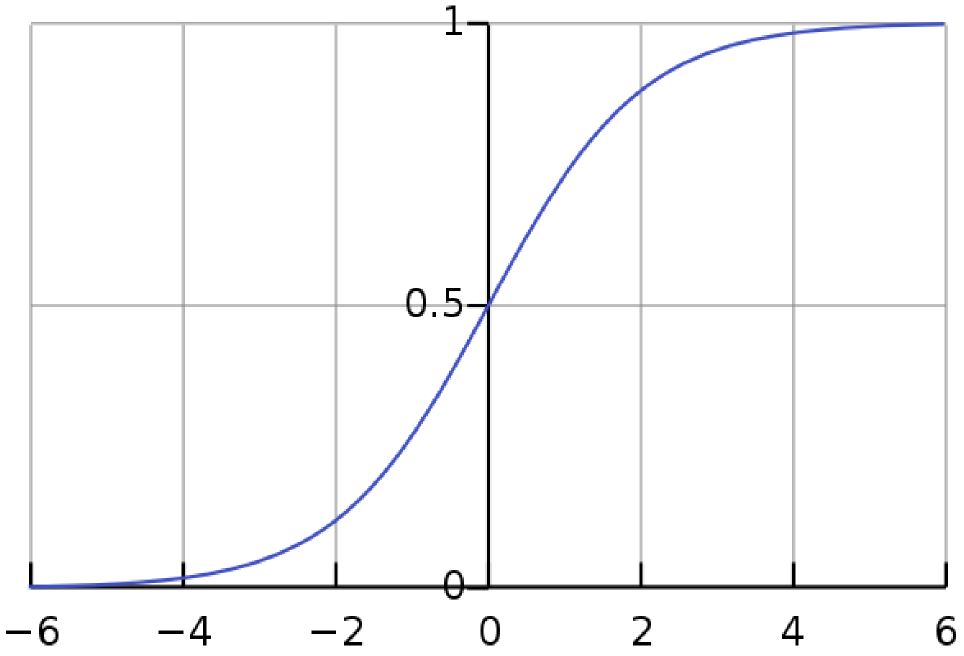
$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}} = \frac{\mathrm{e}^{z}}{1 + \mathrm{e}^{z}}$$

Activation Function: Sigmoid

- Activation function

$$\sigma(z) = \frac{1}{1 + \mathrm{e}^{-z}} = \frac{\mathrm{e}^{z}}{1 + \mathrm{e}^{z}}$$

- Squashes all real numbers into the range [0, 1]
- Thus, good for binary classification
- $\sigma(z)$ denotes the probability for one of the two classes.



Logistic Regression

- Collect some data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- Specify a model $\hat{y}^{(i)} = \sigma\left(\boldsymbol{\beta}^{\top}\boldsymbol{x}^{(i)}\right)$
- Define a loss function
- Find the parameters $\boldsymbol{\beta}$ that minimize the loss function

The Cross-entropy Loss

- The label $y^{(\mathrm{i})}$ either 0 or 1
- $\hat{y}^{(\mathrm{i})} \in (0, 1)$ is the output of the model.
- The cross-entropy loss

$$L = -\frac{1}{N}\sum_{i=1}^{N} y^{(\mathrm{i})} \log \hat{y}^{(\mathrm{i})} + \left(1 - y^{(\mathrm{i})}\right) \log\left(1 - \hat{y}^{(\mathrm{i})}\right)$$

For data point i, only one term exists.

Why the Name?

- Do they look very similar to you?

$$L_{\mathrm{XE}} = -\frac{1}{N}\sum_{i=1}^{N} y^{(\mathrm{i})} \log \hat{y}^{(\mathrm{i})} + \left(1 - y^{(\mathrm{i})}\right) \log\left(1 - \hat{y}^{(\mathrm{i})}\right)$$

$$H(P, Q) = E_P[-\log Q(X)]$$

Why the Name?

- Monte Carlo estimation of an expectation

$$E_P[f(x)] = \int f(x)P(x)dx$$

- The integral can be approximated if we can draw samples $x^{(1)}, \ldots, x^{(K)}$ from $P(x)$

$$E_P[f(x)] \approx \frac{1}{K}\sum_{i=1}^{K} f\left(x^{(i)}\right)$$

Why the Name?

Cross entropy:

$$H(P, Q) = E_P[-\log Q(X)] = -\sum_i P(X = x_i)\ \log Q(X = x_i)$$

- $y^{(\mathrm{i})}$ is drawn from an unknown distribution $P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}\right)$
- $\hat{y}^{(\mathrm{i})}$ is the probability $Q\left(y^{(\mathrm{i})} = 1 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\beta}\right)$
- $1 - \hat{y}^{(\mathrm{i})}$ is the probability $Q\left(y^{(\mathrm{i})} = 0 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\beta}\right)$

$$-\frac{1}{N}\sum_{i=1}^{N} y^{(\mathrm{i})} \log \hat{y}^{(\mathrm{i})} + \left(1 - y^{(\mathrm{i})}\right) \log\left(1 - \hat{y}^{(\mathrm{i})}\right) \approx E_P\left[-\log Q\left(y^{(\mathrm{i})} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\beta}\right)\right]$$

Information Theoretical Perspective

- The cross-entropy is related to the KL divergence

$$
\begin{aligned}
H(P,Q) &= -E_{P(x)}[\log Q(x)] \\
&= H(P) + KL(P||Q)
\end{aligned}
$$

- Minimizing the loss minimizes the distance between the GT distribution $P\left(y^{(i)}\middle|\boldsymbol{x}^{(i)}\right)$ and estimated distribution $Q\left(\hat{y}^{(i)}\middle|\boldsymbol{x}^{(i)},\boldsymbol{\beta}\right)$.

MLE Perspective

- Optimizing the cross-entropy can also be seen as the maximum likelihood estimation of $\boldsymbol{\beta}$ under the binomial distribution.
- We omit the details.

Logistic Regression

- Collect some data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- Specify a model $\hat{y}^{(i)} = \sigma\left(\boldsymbol{\beta}^\top \boldsymbol{x}^{(i)}\right)$
- Define a loss function: cross entropy
- Find the parameters $\boldsymbol{\beta}$ that minimize the loss function

Optimization

- We seek $\boldsymbol{\beta}$ that minimizes a function $L(\boldsymbol{\beta})$
- Assumption: We can evaluate the function and its first-order derivative $\frac{\mathrm{d}f(x,w)}{\mathrm{d}w}$

What is a Minimum?

- $\boldsymbol{x}$ is called a local minimum of function $f(\boldsymbol{x})$ if there is ϵ such that

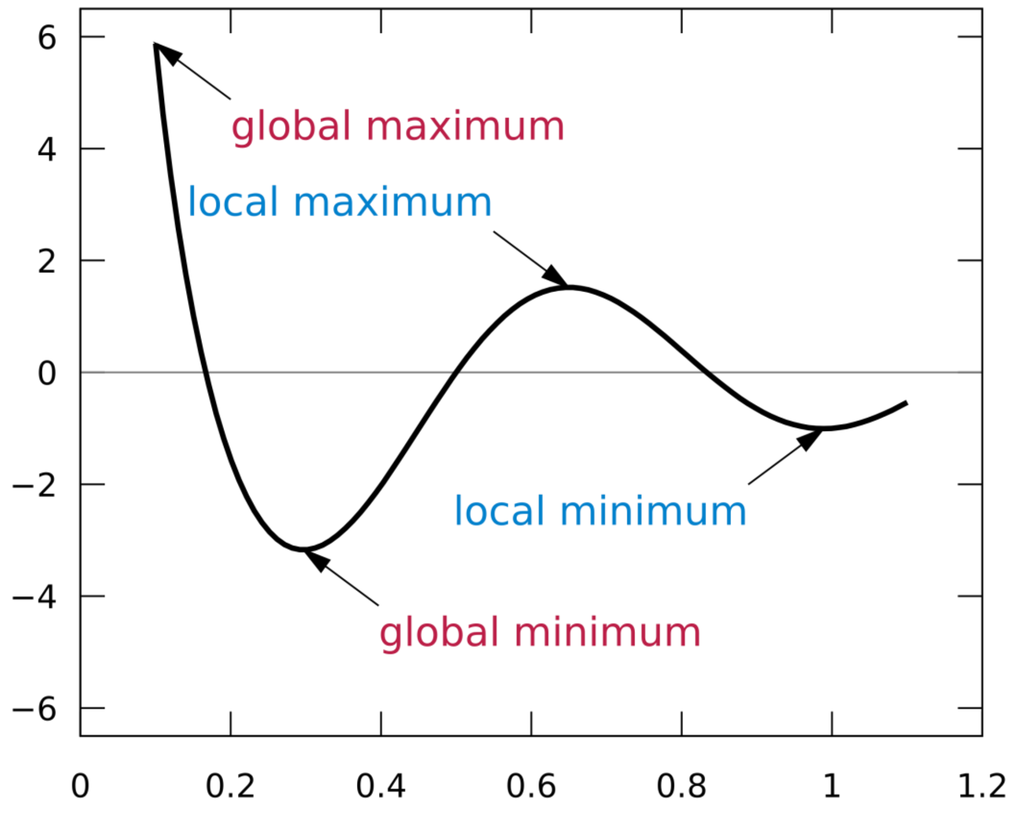
$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}+\boldsymbol{y})$$

  for all $\|\boldsymbol{y}\| < \epsilon$.

- $\boldsymbol{x}$ is called a global minimum of function $f(\boldsymbol{x})$ if

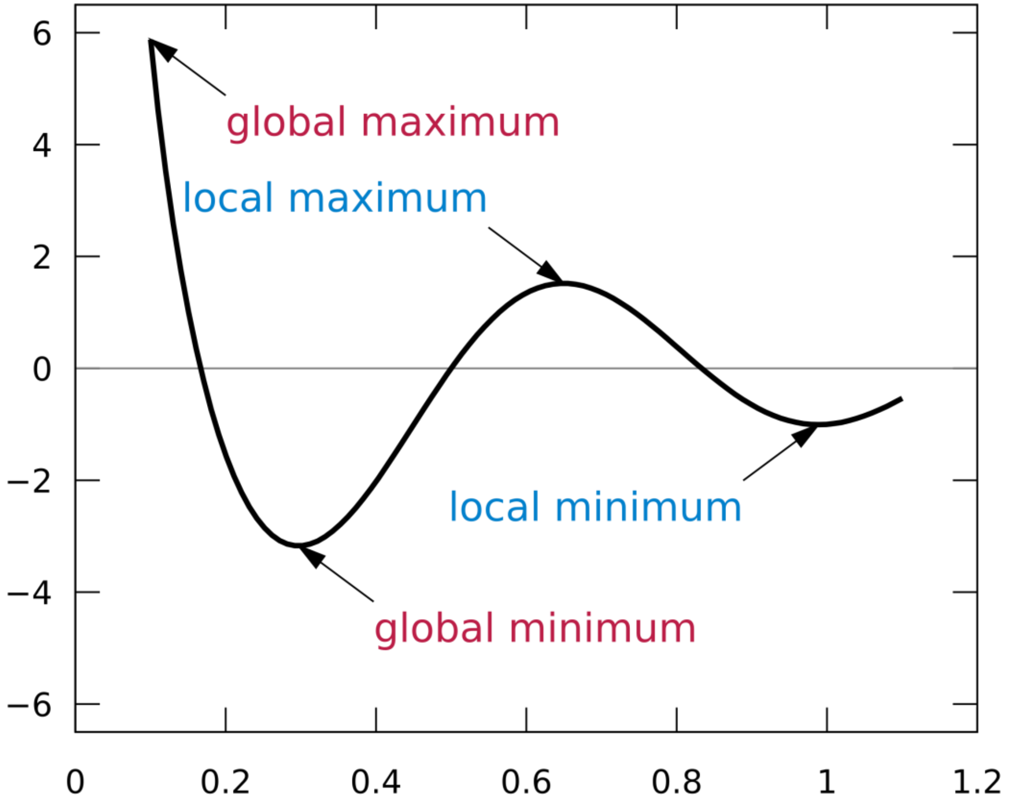
$$f(\boldsymbol{x}) \leq f(\boldsymbol{y})$$

  for all $\boldsymbol{y}$ in the domain of $f(\boldsymbol{x})$



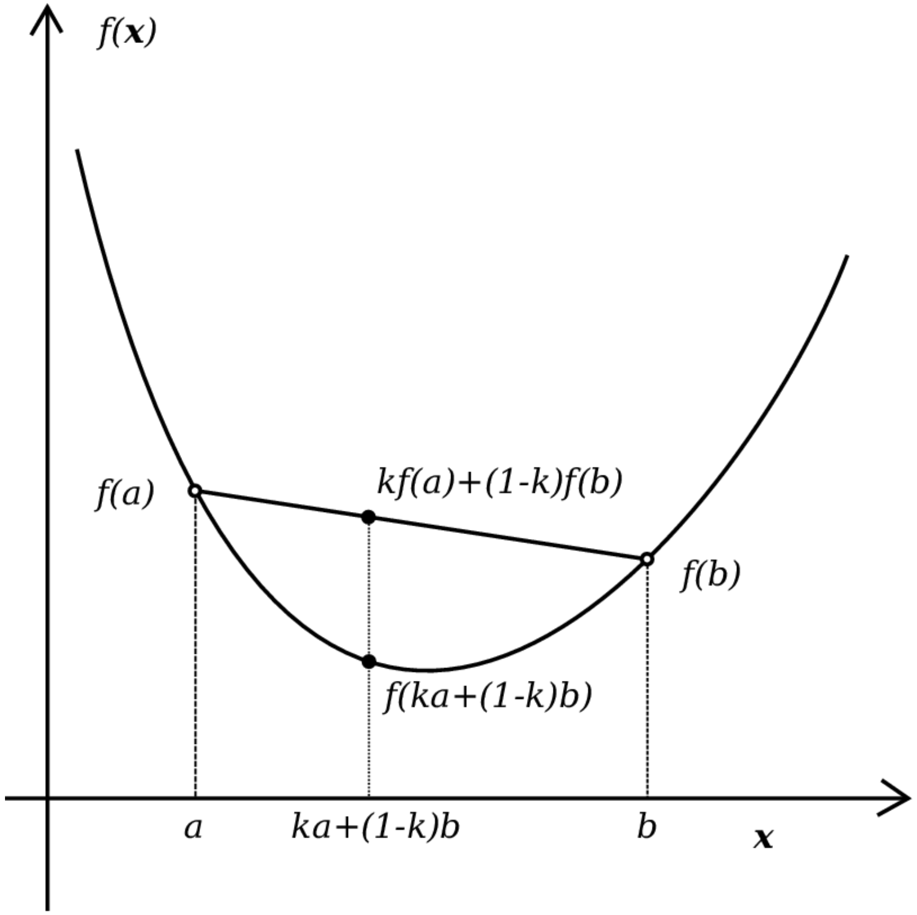
What is a Minimum?

- A global minimum must be a local minimum, but a local minimum may not be a global minimum.
- Multiple local minima cause problems for optimization.



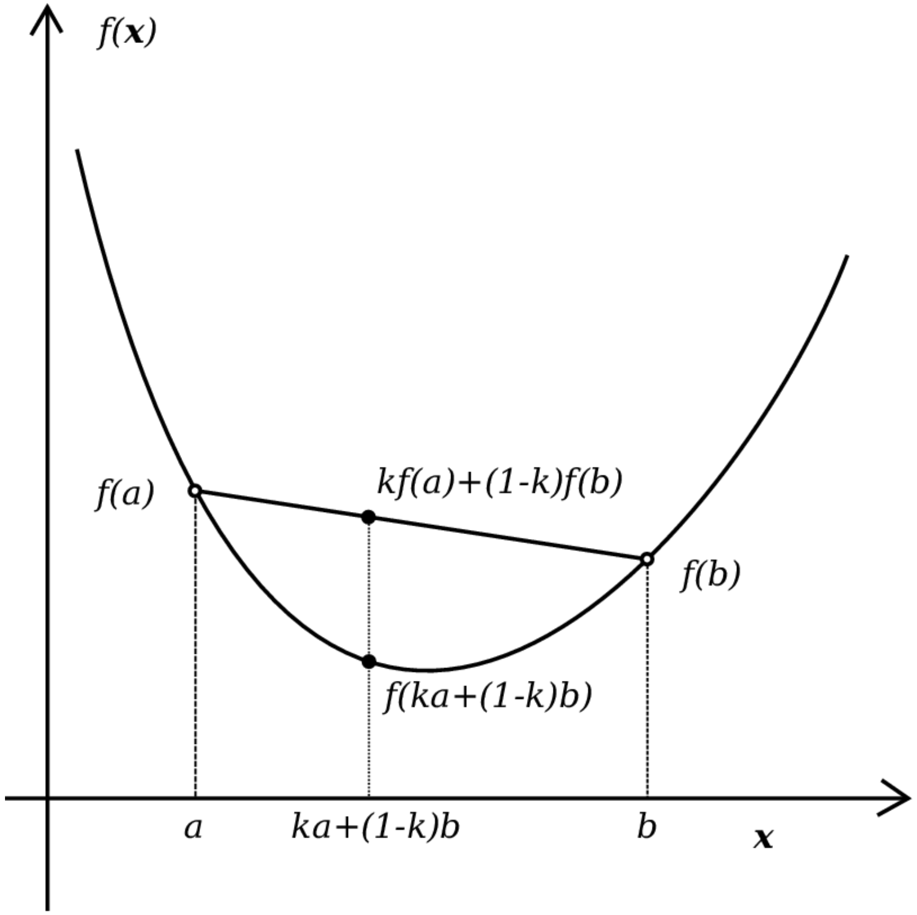
Optimization: Convex Functions

- The line segment connecting $f(\boldsymbol{a})$ and $f(\boldsymbol{b})$ always lies above the function between $\boldsymbol{a}$ and $\boldsymbol{b}$.
- We can understand a convex function as a function where any local minimum is also a global minimum.
- Easy optimization!



Optimization: Convex Functions

- Logistic regression has a convex loss function.
- Deep neural networks usually have non-convex loss functions that are difficult to optimize.
- We will worry about that later!

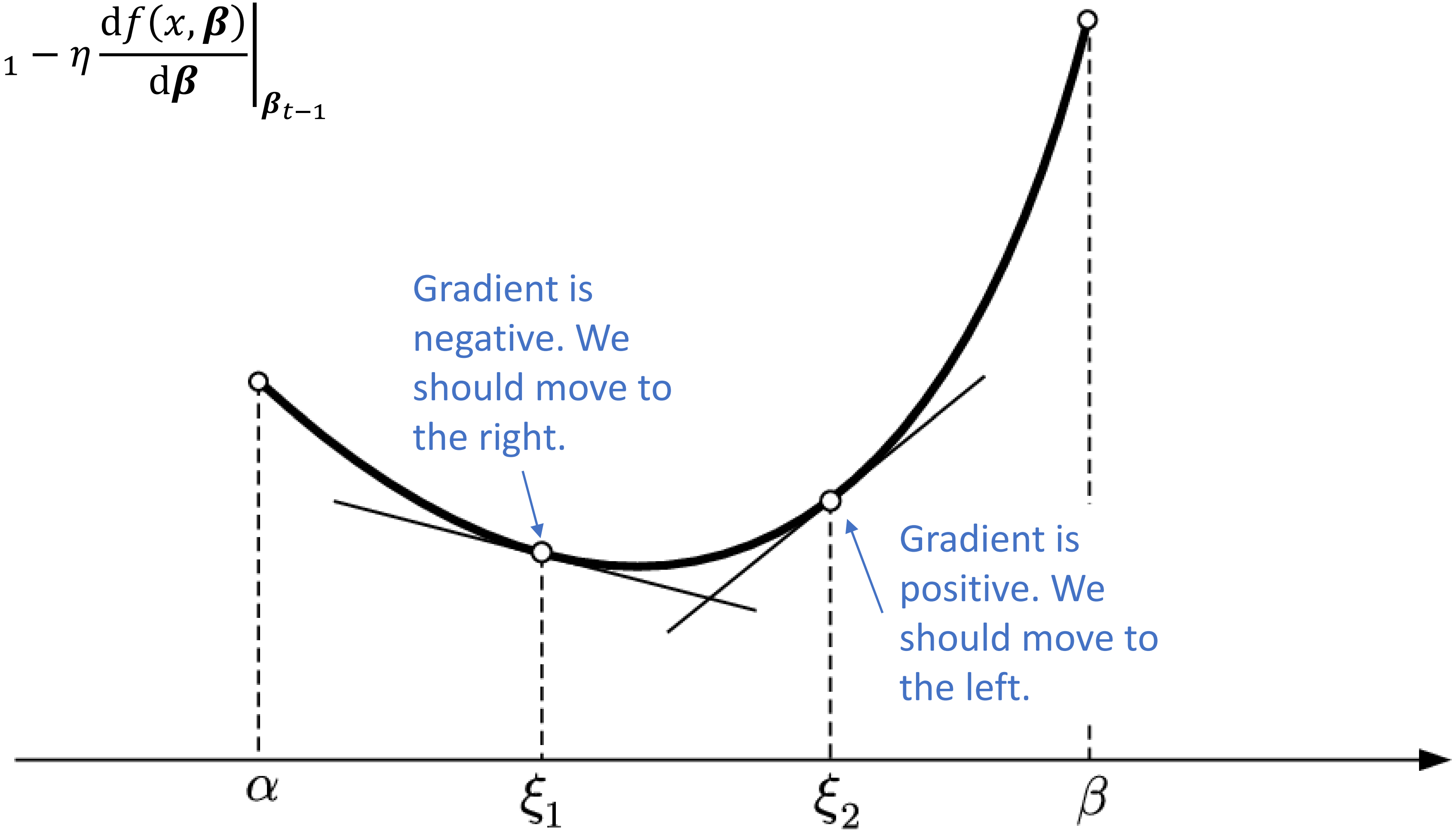


The Gradient Descent Algorithm

- Input: loss function $L(\boldsymbol{\beta})$ and initial position $\boldsymbol{\beta}_0$
- Repeat for a predefined amount of time (or until convergence)
  - Move in the direction of negative gradient
  - $\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta \left.\frac{\mathrm{d}L(\boldsymbol{\beta})}{\mathrm{d}\boldsymbol{\beta}}\right|_{\boldsymbol{\beta}_{t-1}}$
- η is a small constant called the learning rate

Gradient Descent

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta \left. \frac{\mathrm{d}f(x, \boldsymbol{\beta})}{\mathrm{d}\boldsymbol{\beta}} \right|_{\boldsymbol{\beta}_{t-1}}$$

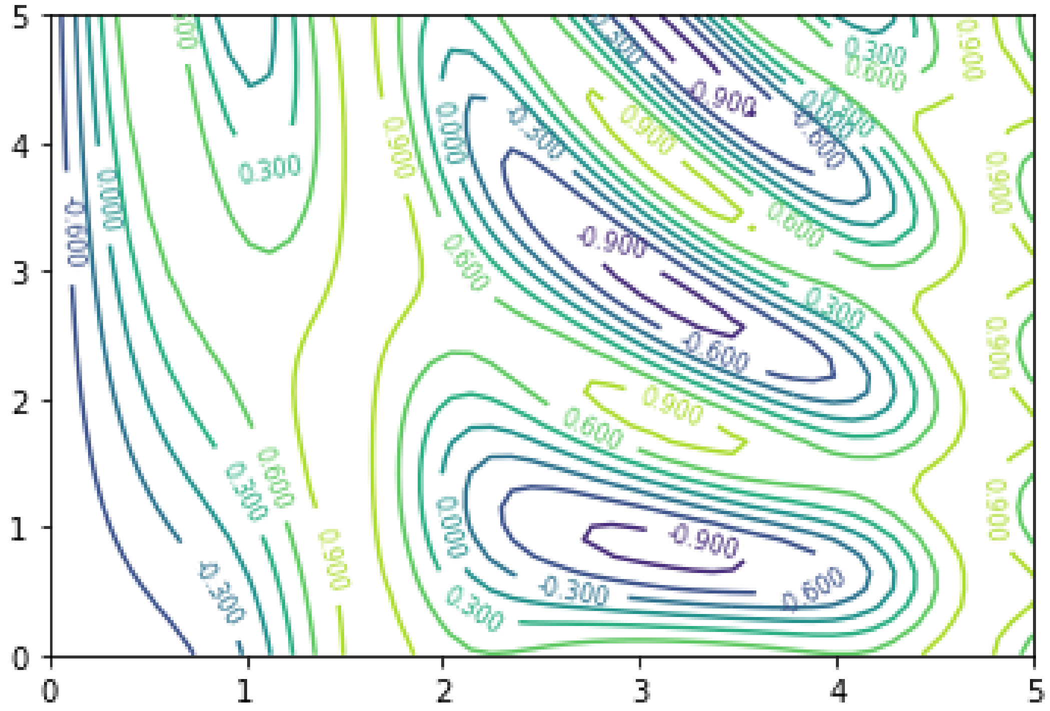


Optimization: Gradient Descent

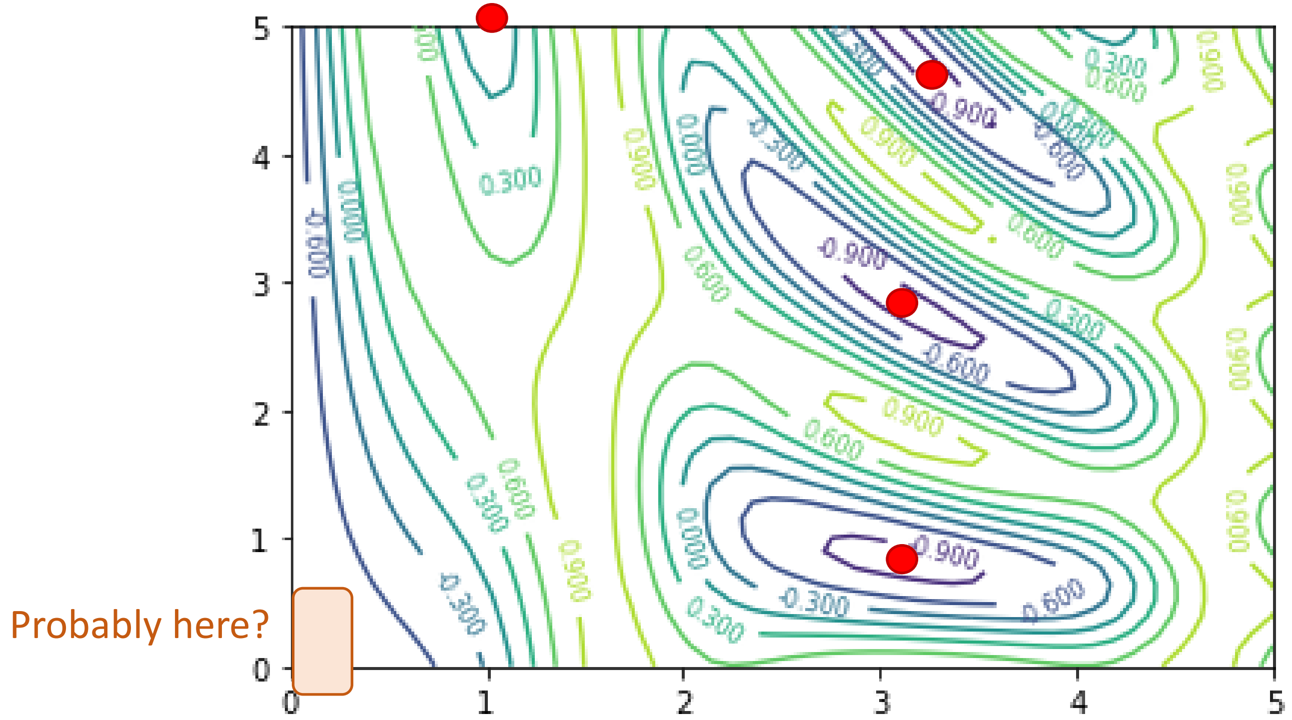
- Starting from a given initial position $\boldsymbol{\beta}_0$
- Repeat for a predefined amount of time (or until convergence)
  - Move in the direction of negative gradient
  - $\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta \frac{\mathrm{d}L(x,\boldsymbol{\beta})}{\mathrm{d}\boldsymbol{\beta}}\Big|_{\boldsymbol{\beta}_{t-1}}$
- This produces a sequence of $\boldsymbol{\beta}$
- $\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{\mathrm{T}}$
- That goes increasingly closer to the optimum value $\boldsymbol{\beta}^*$
- If, as $\mathrm{T} \to \infty$, $\boldsymbol{\beta}_{\mathrm{T}} \to \boldsymbol{\beta}^*$, we say that the sequence converges to $\boldsymbol{\beta}^*$.

2D Functions

- Loss surface in 2D = contour diagrams / level sets $L_a(f) = \{\boldsymbol{x} | f(\boldsymbol{x}) = a\}$

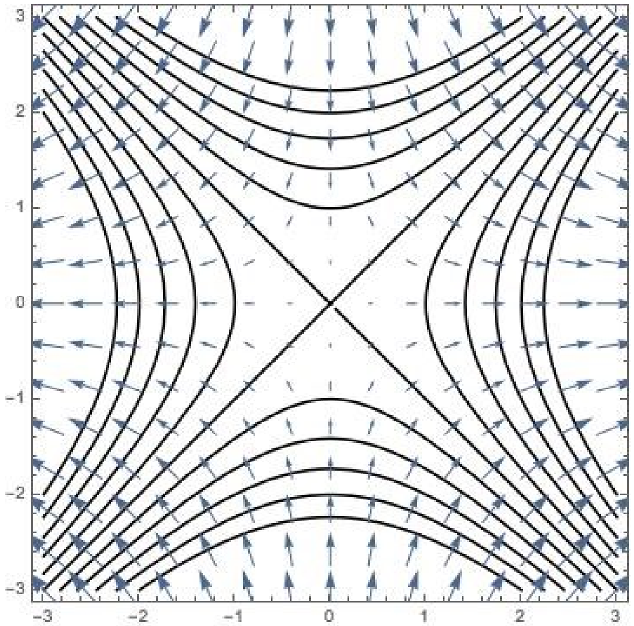


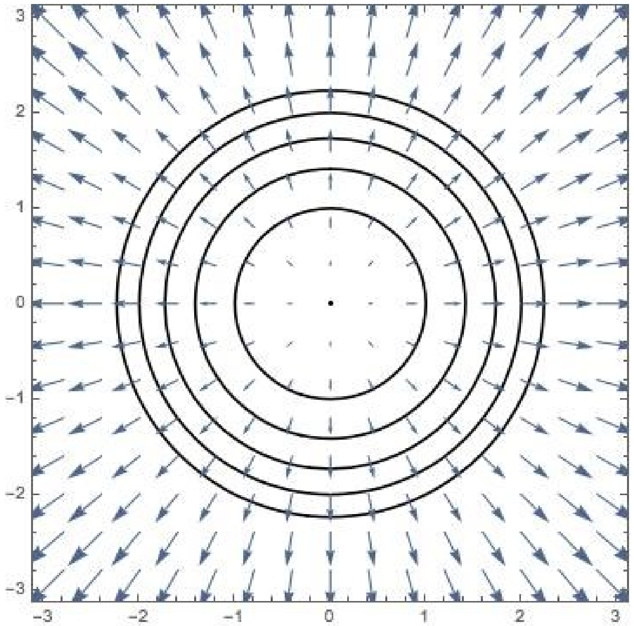
All local minima on the diagram



2D Functions

- Loss surface in 2D = contour diagrams / level sets $L_a(f) = \{\boldsymbol{x} | f(\boldsymbol{x}) = a\}$



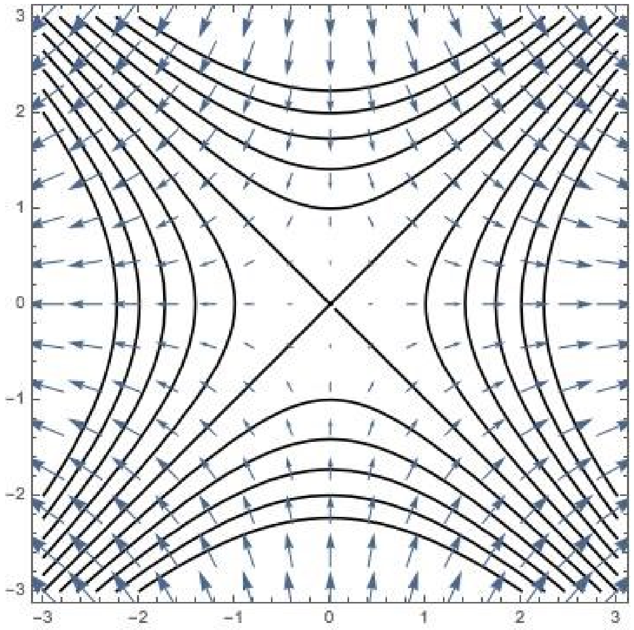


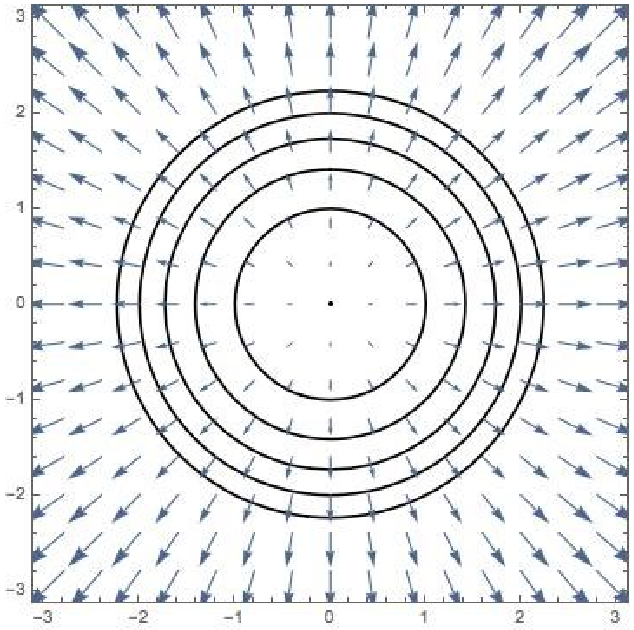
The gradient direction is the direction along which the function value changes the fastest (for a small change of $\boldsymbol{x}$ in Euclidean norm).

Along the level set, the function value doesn't change.

2D Functions

- Loss surface in 2D = contour diagrams / level sets $L_a(f) = \{\boldsymbol{x} | f(\boldsymbol{x}) = a\}$





For a differentiable function $f(\boldsymbol{x})$, its gradient of $\frac{\mathrm{d}f}{d\boldsymbol{x}}$ at any point is either zero or perpendicular to the level set at that point.

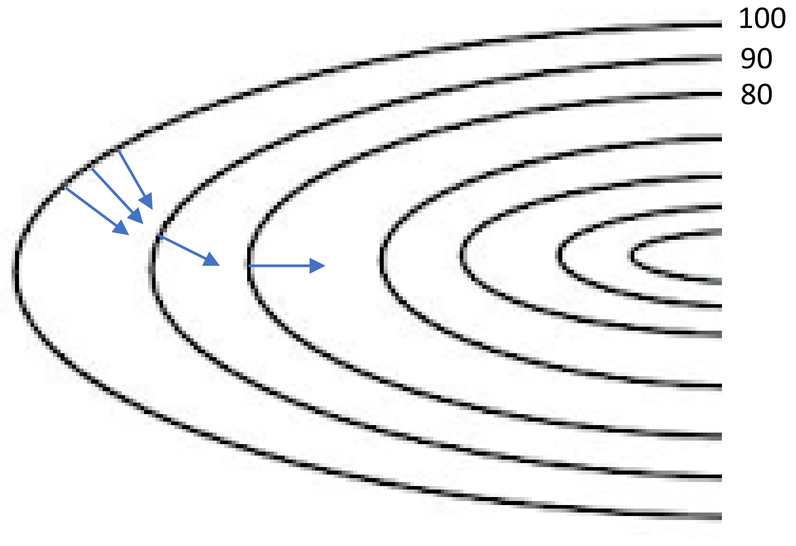
Gradient Descent on Convex Functions

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta \left.\frac{\mathrm{d}f(x,\boldsymbol{\beta})}{\mathrm{d}\boldsymbol{\beta}}\right|_{\boldsymbol{\beta}_{t-1}}$$

The learning rate η determines how much we move at each step.

We cannot move too much because the gradient is a local approximation of the function.

Thus, the learning rate is usually small.



Contour diagram / Level sets

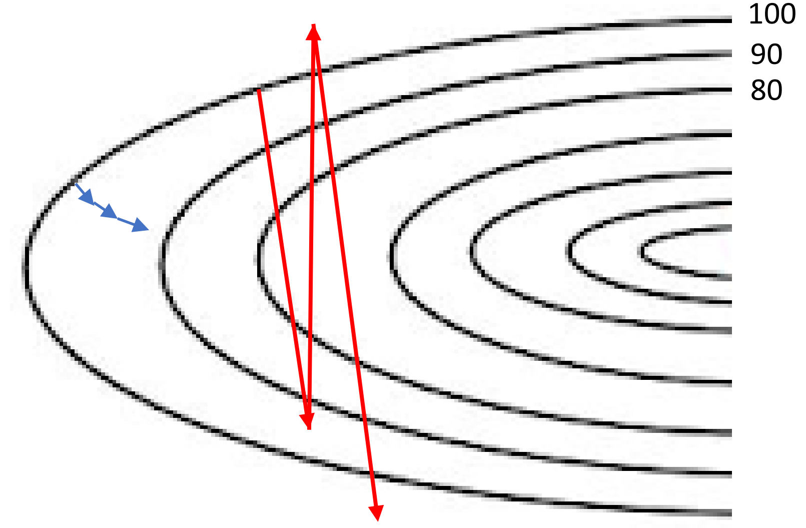
Gradient Descent on Convex Functions

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta \left.\frac{\mathrm{d}f(x, \boldsymbol{\beta})}{\mathrm{d}\boldsymbol{\beta}}\right|_{\boldsymbol{\beta}_{t-1}}$$

The learning rate η determines how much we move at each step.

Too small a learning rate η: slow convergence

Too large a learning rate η: oscillation, overshooting



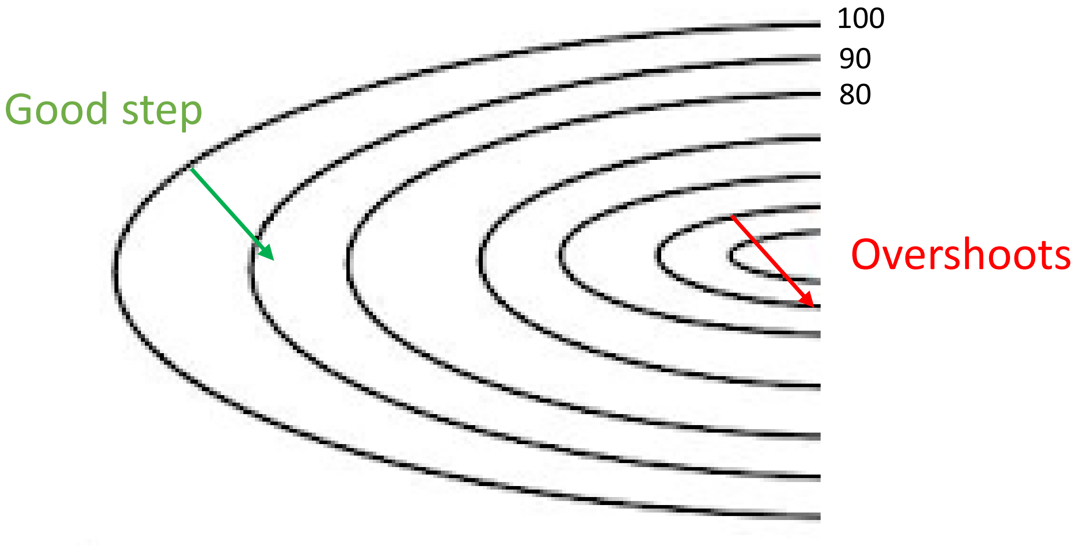
Contour diagram / Level sets

Gradient Descent on Convex Functions

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta \frac{\mathrm{d}f(x, \boldsymbol{\beta})}{\mathrm{d}\boldsymbol{\beta}}\bigg|_{\boldsymbol{\beta}_{t-1}}$$

The learning rate η determines how much we move at each step.

As we move closer to the minimum, we often decrease η so that we do not overshoot.



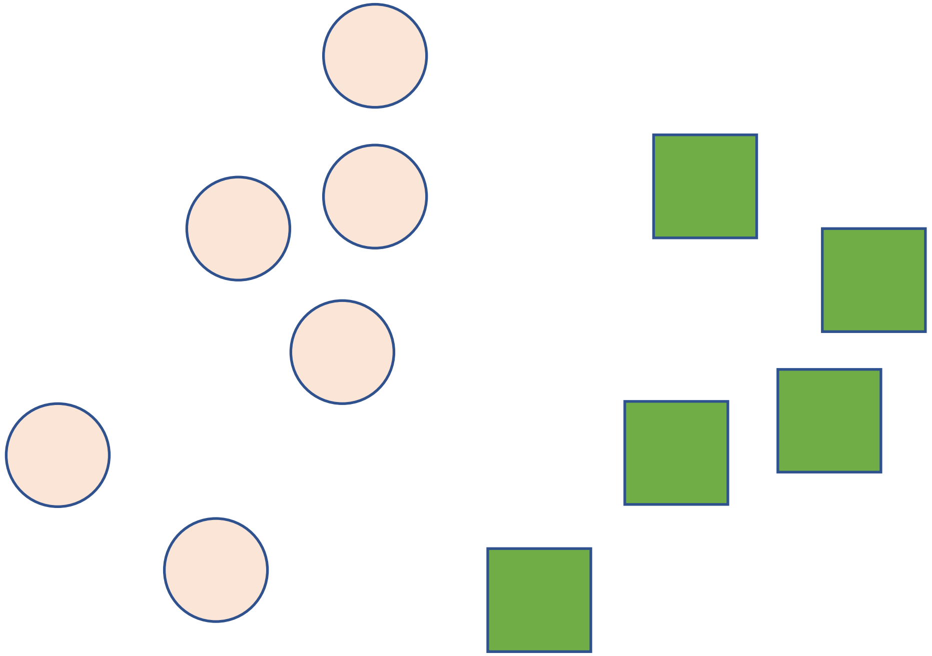
Contour diagram / Level sets

Logistic Regression

- Collect some data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \dots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
- Specify a model $\hat{y}^{(i)} = \sigma\left(\boldsymbol{\beta}^{\top}\boldsymbol{x}^{(i)}\right)$
- Define a loss function: cross entropy
- Find the parameters $\boldsymbol{\beta}$ that minimizes the loss function

A single neuron is still VERY limited

- It only works well when there is a straight line that can separate two classes.



A single neuron is still VERY limited

- Perceptron (similar to logistic regression) infamously fails to represent the XOR function (Minsky & Papert, 1969).