Lecture 04
07.10.2024

Logistic regression (continued)

Feature engineering

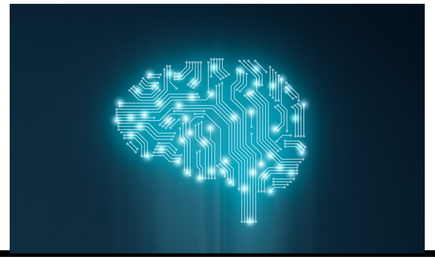
Outline

- Logistic regression
  - Performance metrics
  - Multinomial logistic regression

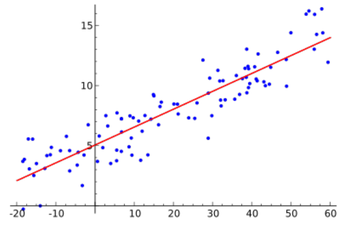
- Feature engineering
  - Defining features
  - Data statistics

- Announcements:
  - Exercise hours: problem sets and extra python exercises

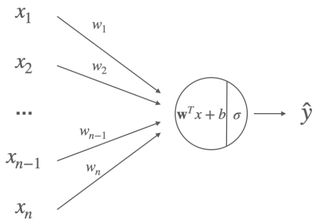
Introduction



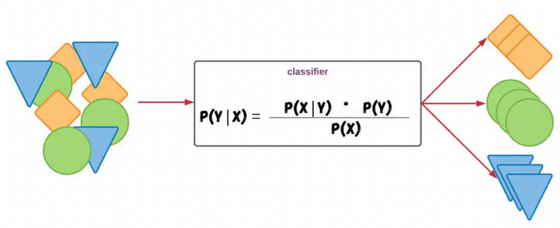
Linear regression



Logistic regression



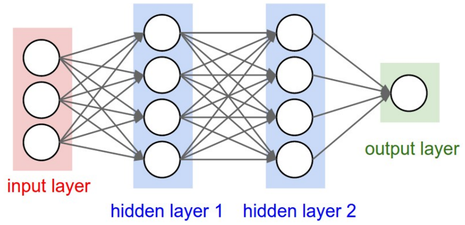
Naive Bayes



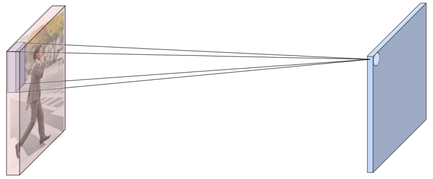
KNN

AI ethics

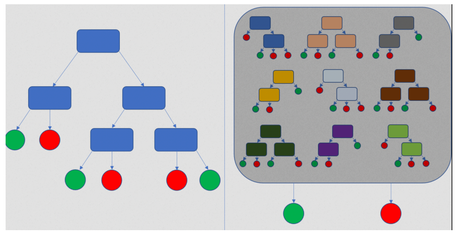
Neural networks



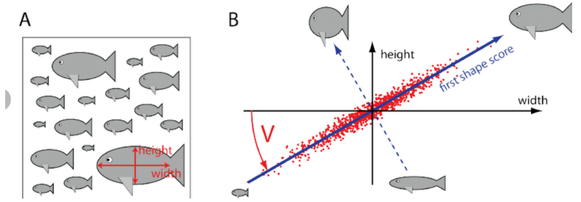
Convolutional neural networks



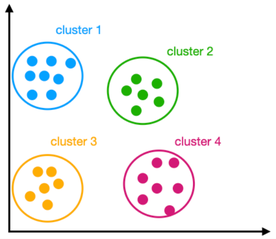
Decision-trees



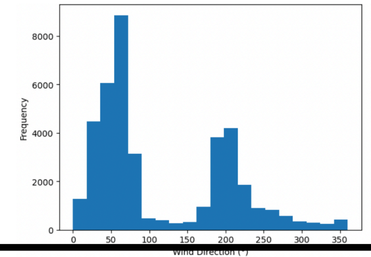
Dimensionality reduction



Clustering



Reinforcement learning



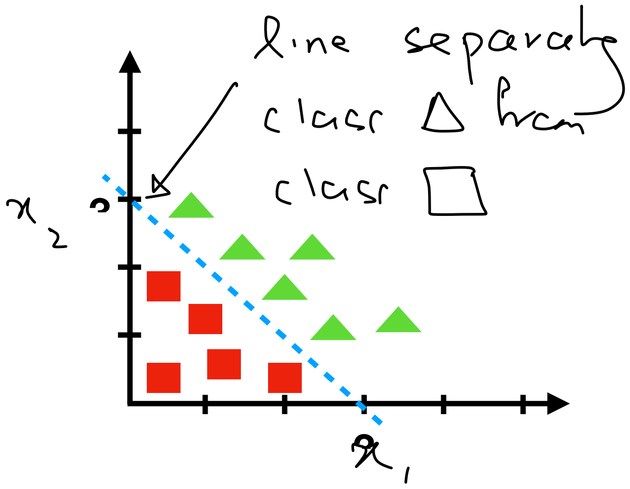
Logistic regression

Performance metrics

Multinomial logistic regression

Review of logistic regression

- Setting $\{x^i, y^i\}_{i=1}^N \qquad y^i \in \{0,1\}, \quad x^i \in \mathbb{R}^d$



- Approach $z = b + w_1 x_1 + \cdots + w_d x_d$

$z \geq 0 \rightarrow$ class 1

$< 0 \rightarrow$ class 0

- Loss function

$$J(w,b) = -\frac{1}{N}\left[\sum_{i=1}^{N} y^i \log\left(1+e^{-z^i}\right) + (1-y^i)\log\left(1+e^{+z^i}\right)\right]$$

$x^i \Longrightarrow z^i = b + w_1 x_1^i + \cdots + w_d x_d^i$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$ probability of class 1

$$1-\sigma(z) = \frac{e^{-z}}{1+e^{-z}} = \frac{1}{e^{z}+1}$$ prob. of class 0

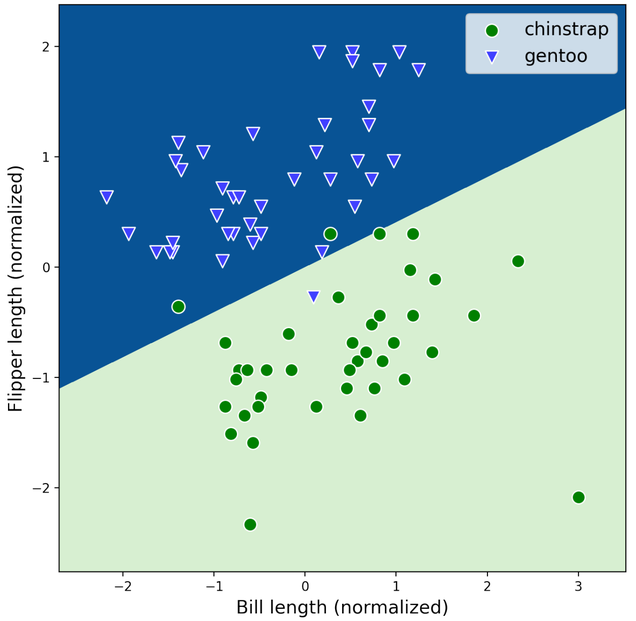
$$z = b + w_1 x_1 + \cdots + w_d x_d$$

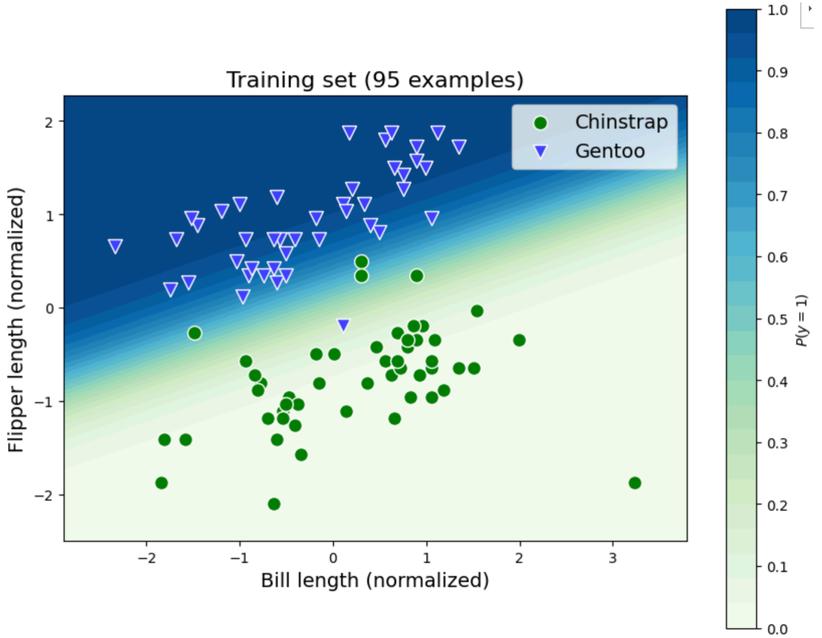
interpret the loss as the "cross-entropy" between the true probability & our estimated ones based on $\sigma(z)$.

Logistic regression on the penguin data

Palmer Penguins

	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Chinstrap	49.0	19.5	210.0	3950.0
1	Chinstrap	50.9	19.1	196.0	3550.0
2	Gentoo	42.7	13.7	208.0	3950.0
3	Chinstrap	43.5	18.1	202.0	3400.0
4	Chinstrap	49.8	17.3	198.0	3675.0

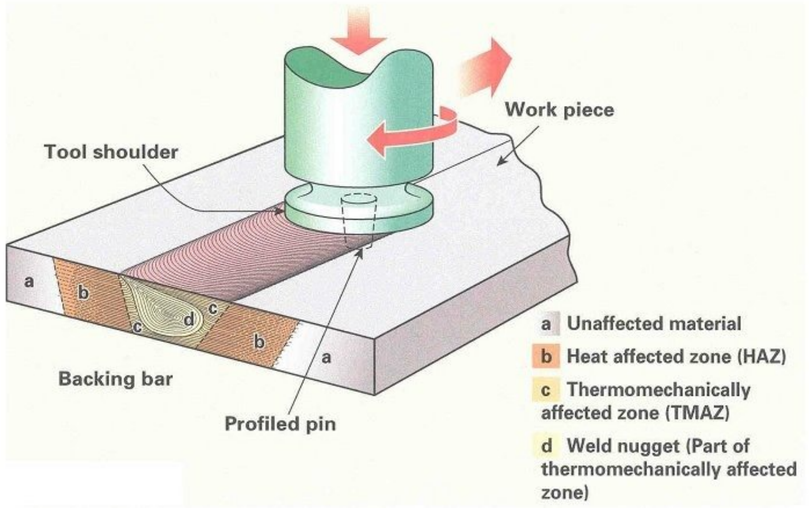


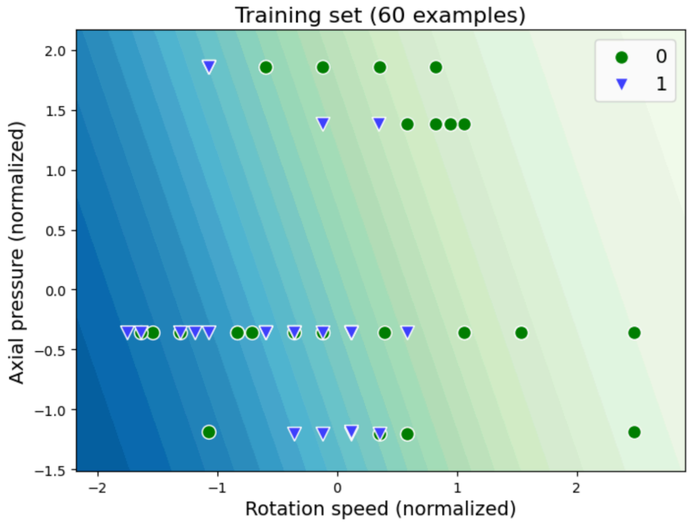


Based on dataset available here: https://www.kaggle.com/code/parulpandey/penguin-dataset-the-new-iris

Logistic regression exercise last week

- **Dataset 1:** Void Formation in Welding, based on the paper
- **Goal:** formation of voids in friction stir welding as a function of the operation conditions
  - Tool rotational speed, axial pressure
  - The label: void or not void





- **Dataset 2:** discriminate between sonar signals bounced off a mine (metal cylinder) and those bounced off a roughly cylindrical rock
- **Goal:** predict whether the object is mine or rock based on
  - The features (60 of them) are the energy within a particular frequency band, integrated over a certain period of time
  - The label: rock/mine

Performance metrics for binary classification

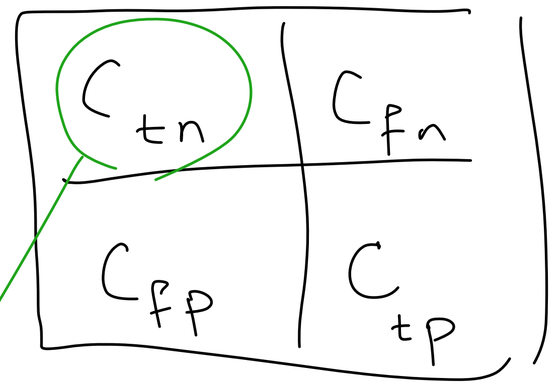
Confusion matrix, accuracy, error rate, recall

$y = 0, \quad \hat{y} = 0$ true negahr tn

$y = 0, \quad \hat{y} = 1$ fala posihe fp

$y = 1, \quad \hat{y} = 0$ fala negahe fn

$y = 1, \quad \hat{y} = 1$ true posihe tp



C_{tn} : # of true negahes

Accuracy $\frac{C_{tn} + C_{tp}}{N}$ =: total number of correct point we have classified

error rate $\frac{C_{fn} + C_{fp}}{N} = 1 - \frac{C_{tn} + C_{tp}}{N}$

recall $\frac{C_{tp}}{C_{tp} + C_{fn}}$

Exercise

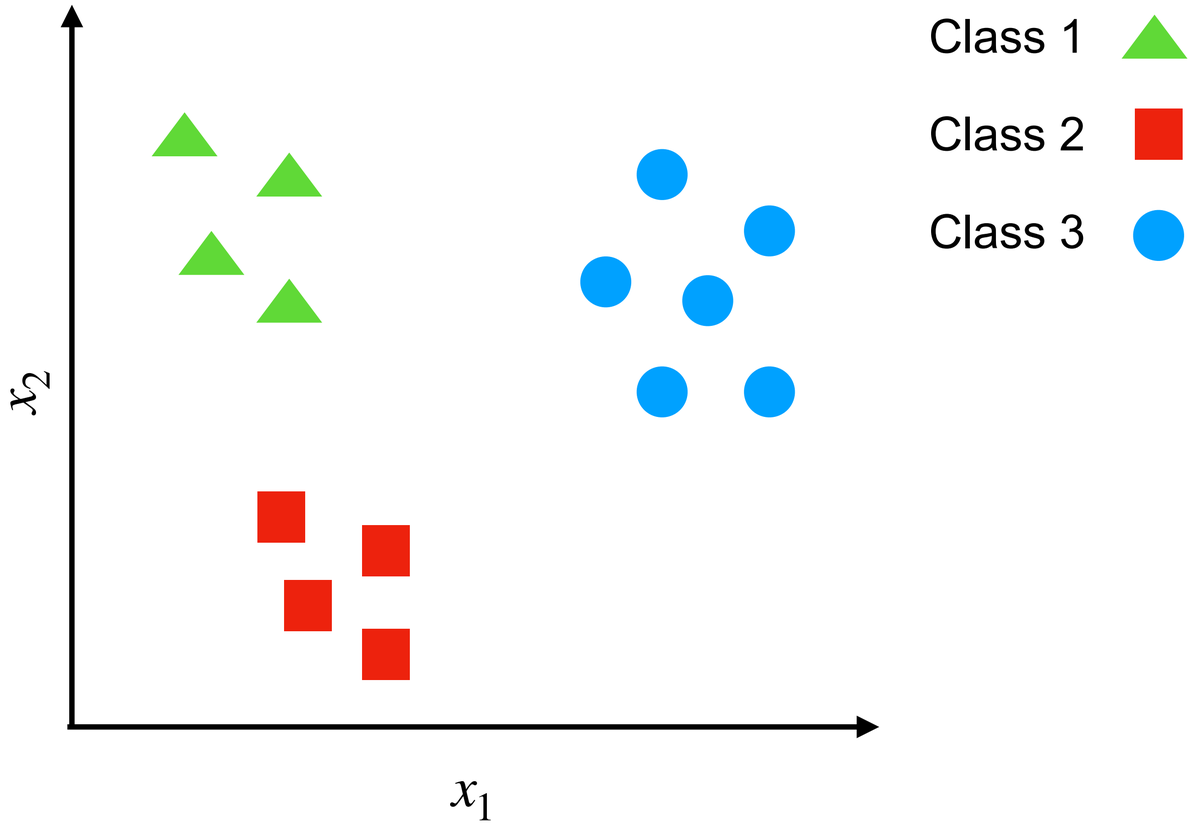
Performance metric for binary classification

- We have used two approaches to train classifiers for spam email detection: “non-spam” (class 0) and “spam” (class 1)
- Our test set has1000 emails, 900 of which were non-spam
- Approach 1: classified all data as non-spam
- Approach 2: classified 850 of non-spam emails as non-spam and 50 of spam emails as spam
- Write the confusion matrix of each approach
- Compute the error rate, accuracy and recall of each algorithm
- What do you conclude?

Multinomial logistic regression

How to deal with a multi-class (more than 2) classification problem ?

$y \in \{1, 2, \cdots, K\}$



Example: Medical diagrams:

Not ill (y = 1),Cold(y = 2), Covid(y = 3)

$$z_1 = \mathbf{w}_1^T \boldsymbol{x} + b_1$$
$$z_2 = \mathbf{w}_2^T \boldsymbol{x} + b_2$$
$$z_3 = \mathbf{w}_3^T \boldsymbol{x} + b_3$$

idem ...

$z_c \geq z_{\hat{c}}, \forall \hat{c} \in \{1, 2, \cdots K\}$

classify

as class c.

Multinomial logistic regression through probabilistic interpretation

Extend the logistic function to the multi-class setting by defining the softmax function

$$softmax(z) = \left(\frac{\exp(z_1)}{\sum_{j=1}^{K}\exp(z_j)}, \ldots, \frac{\exp(z_K)}{\sum_{j=1}^{K}\exp(z_j)}\right)$$

After applying softmax, each component will be in the interval $(0,1)$ and the component will add up to 1, so that they can be interpreted as probabilities

well-defined probably on K classes

Example: $softmax([1,5,2,3]) = [0.0152,0.8310,0.0414,0.1125] \in \mathbb{R}^4$

Softmax regression: extends the probabilistic interpretation of logistic loss function

Training multinomial logistic regression

Multinomial (Categorical) Cross-Entropy Loss

$$J(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} \mathbf{1}\{y^{(i)} = c\} \log\left(\frac{\exp(z_c^i)}{\sum_{j=1}^{K}\exp(z_j^i)}\right)$$

where $\mathbf{1}\{y^i = k\}$ is "indicator function", it work as: $\mathbf{1}\{\text{True statement}\} = 1$ and $\mathbf{1}\{\text{False statement}\} = 0$

$$z_c^i = b_c + w_{c,1}x_1^i + w_{c,2}x_2^i + \cdots + w_{c,d}x_d^i \qquad \forall c \in \{1, 2, \cdots, K\}$$

given x^i, $z_c^i = [1, 5, 2, 3]$

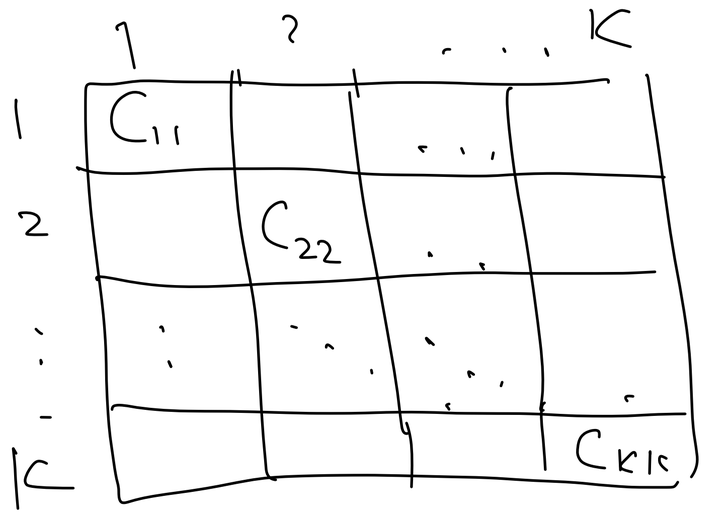
true label y^i : class 2

$$1\{y^i = 2\}\log(0.831) + 1\{y^i = 1\}\log(0.0152) + 1\{y^i \neq 3\}\log(0.0112) + 1\{y^i = 4\}\log(0.1125)$$

$1\{y^i = 1\} = 0$, $1\{y^i \neq 3\} = 0$

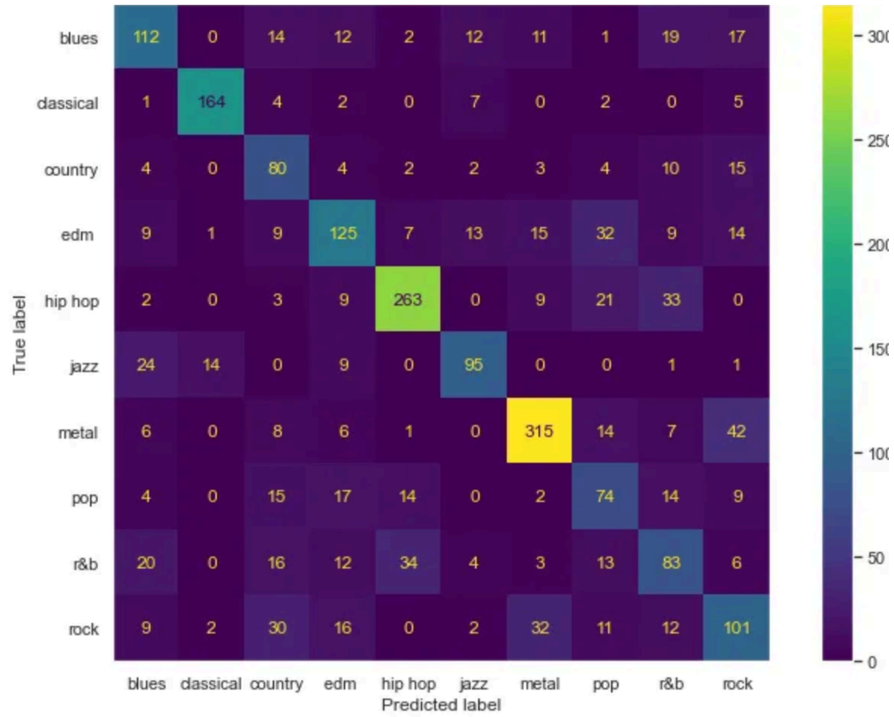
Performance metric

Confusion matrix



C_{11} : # of data point that were in class 1 & were correctly classified as class 1

C_{21} : # of data point in class 2 but classified as class 1



Example from genre classification based on music data

Accuracy

$$\frac{\sum_{i=1}^{K} C_{ii}}{N}$$

Error rate

$$1 - \frac{\sum_{i=1}^{K} C_{ii}}{N}$$

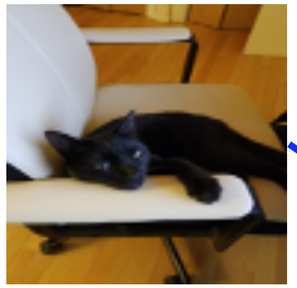
Feature engineering

What is an input representation in ML?

A representation is a mathematical form (*e.g.*, a vector)

- It describes an observation in the real-world (*e.g.*, an image, waveforms, signals, …)
- It is used for subsequent steps (*e.g.*, a classifier) to produce the outcome of interest (*e.g.*, recognizing objects)
- It is often more compact than the original observation (lower dimensionality)
- It is potentially more robust to nuisances
- With a good representation, subsequent steps should be easier

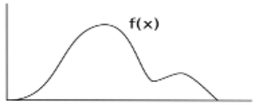
The machine learning pipeline

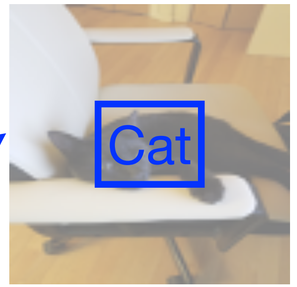


Representation

[81 20 84 64 58 39 17 54 72 15]

ML algorithm





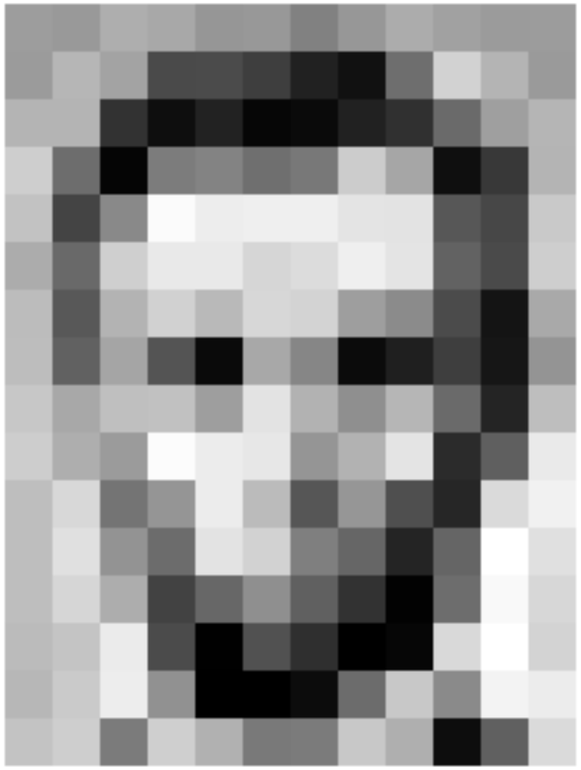
What is a Feature?

A feature vector is a representation,
***i.e.*, a mathematical form that describes an observation in the real-world…**

Examples of representations

Image, pixel values

The image is translated into features representing the value of each pixel in the image

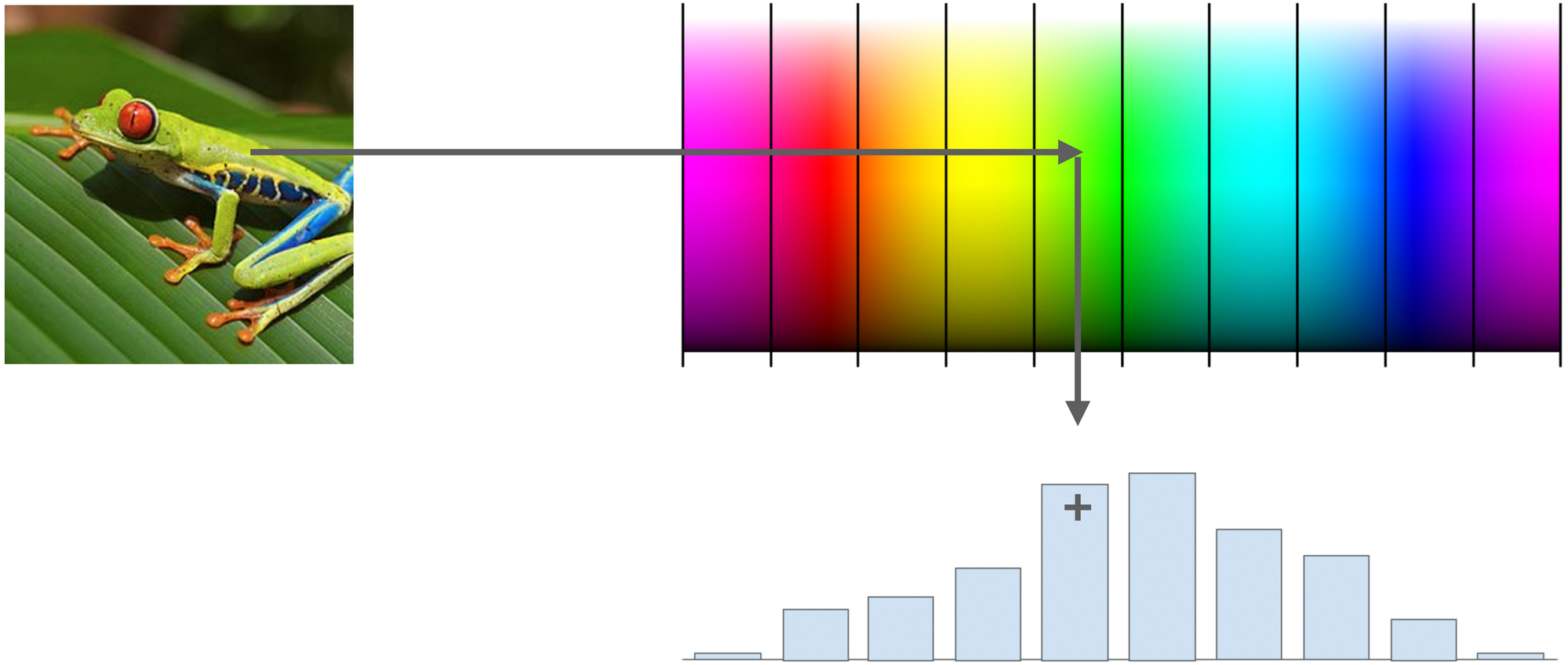


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

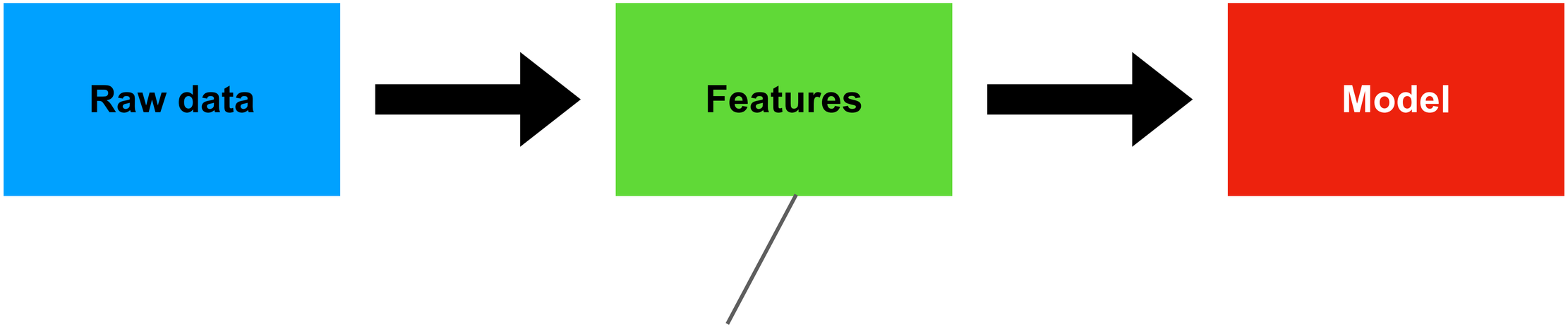
Examples of representations
Image, color histogram

number of pixels that have colors in each of a fixed list of color ranges



Feature engineering

transforming **raw data** into a **feature vector** that represents the underlying data well



Designing clever features is a key part of the machine learning pipeline

For simple models, most of the “heavy lifting” is done there

Features
Types of features

- **Numerical**
  - *e.g.,* height, temperature, price, …
- **Ordinal (an intrinsic order on the categories)**
  - *e.g.,* “like”, “somewhat like”, “neutral”, “somewhat dislike”, “dislike”
- **Categorical (no intrinsic order on the finite categories)**
  - *e.g.,* color, gender, species, …

Preprocessing: the process of transforming raw feature vectors into a representation that is more suitable for ML algorithms

Techniques differ depending on type of feature:

- Numerical, ordinal and categorical features need to be handled differently

Looking “into” a feature

Probabilistic view: data is generated from an unknown probability distribution

Probabilisty distribution

Ordinal/categorical data: Probability mass function

Numerical with continuous values: probability density function

$S = \{s_1, s_2, \cdots, s_6\}$

$p : S \longrightarrow \mathbb{R}$

$\sum_{i=1}^{6} p(s_i) = 1, \quad p(s_i) \geq 0$

$p(s_i)$: probability that the dice rolls i

Empirical distribution: distribution of observed data

roll dice 100 times

face	s_1	s_2	s_3	s_4	s_5	s_6
#	15	40	8	8	19	10

$\hat{p}(s_1) = \frac{15}{100}, \quad \hat{p}(s_2) = \frac{40}{100}, \quad \cdots$

Summary statistics for "looking at" a feature

Data: $\{x^i\}_{i=1}^N \qquad x^i \in \mathbb{R}$

Mean: average value $\frac{1}{N}\sum_{i=1}^{N} x^i = \mu$ sample mean

Variance: measures how far values are from mean $\frac{1}{N-1}\sum_{i=1}^{N}(x^i-\mu)^2 = \sigma^2$

Standard deviation: square root of variance $\sqrt{\sigma^2} = \sigma$ sample variance / std.

Quantile: k-th q-quantile, Note: data needs to be ordered from lowest value to highest : x_I

index of data as $I = \frac{Nk}{q}$, ex: $N=100, k=1, q=4, I = \frac{100\times1}{4} = 25$

- if I is not an integer round up to nearest integer $\Rightarrow x_I$.
- if I is an integer, choose x_I corresponds to I or $I+1$

Summary statistics for “looking at” a feature

Data: ordered

Mean: average value

Mode: most occurring value

Median: (2nd quartile or 50th percentile) $k=2, q=4$, $I = \frac{N}{4} \cdot 2 \Rightarrow x_I$

1st quartile (25th percentile) $k=1, q=4$, $I = N \cdot \frac{1}{4} \Rightarrow x_I$

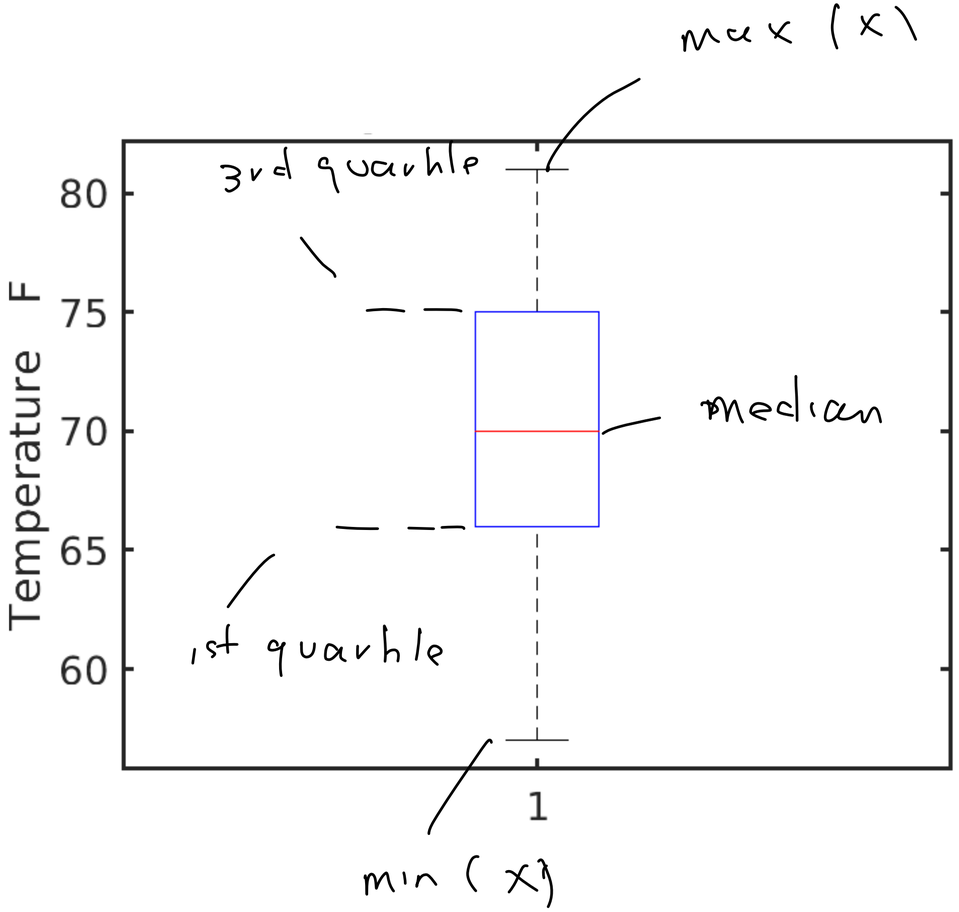
3rd quartile (75th percentile): $k=3, q=4$ $I = N \cdot \frac{3}{4} \Rightarrow x_I$

Variance: measures how far values are from mean

Standard deviation: square root of variance

Range: max value - min value

Interquantile ranges: difference between quantiles

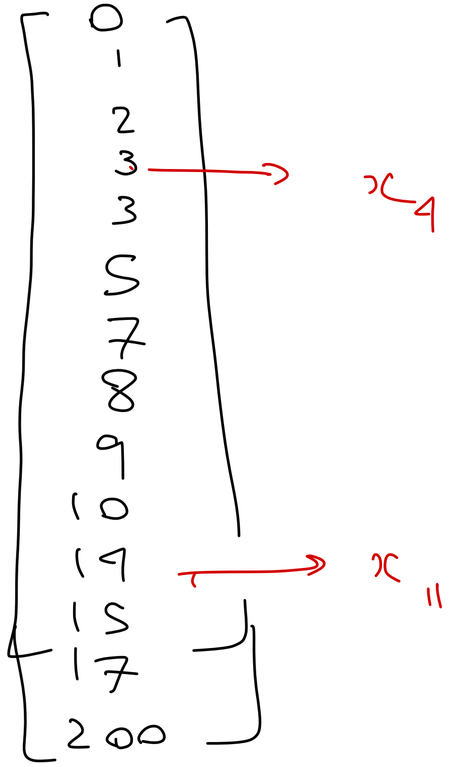


Example of summary statistics

Feature value:

$N = 14$

$x =$



Mean: 21

Median: 7.5

Mode: 3

1/4 Quantile: 3

3/4 Quantile: 14

$N \times \frac{1}{4} = 3.5 \rightarrow$ round up to 4 & return x_4

standard deviation: 51.79

Range: 200

Inter (1/4, 3/4) quantile range: 11

Numerical features
Data normalization

Data normalization / feature scaling: Normalize features (bring them all to the same scale)

Crucial step in preprocessing:

- Many classifiers (e.g. KNN that we will see in next lectures) rely on distance metrics
- Gradient descent will converge faster
- Coefficients are penalized appropriately (in the case where regularization is applied)

Numerical features
Data normalization - Example

Example:

KNN with Palmer Penguins dataset

Features: body mass & bill length

Q: Which feature matters the most for the distance metric?

A:

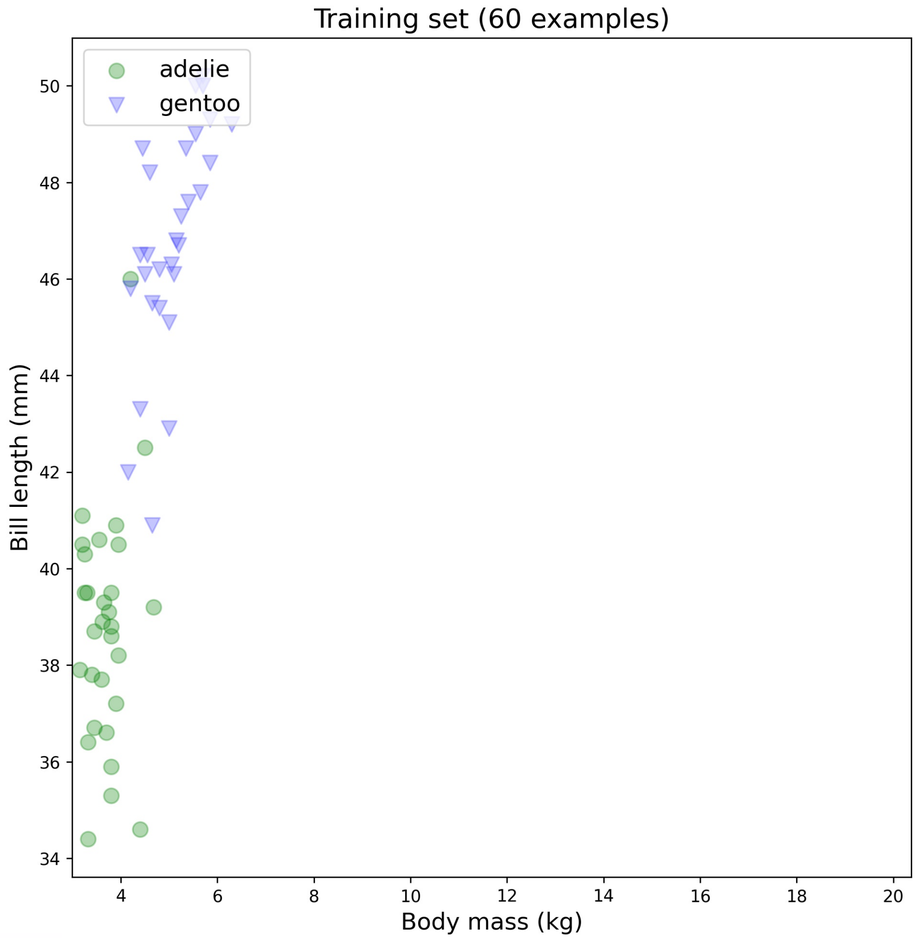
- If body mass in **kg** and bill length in **mm**

  → **bill length** matters more

- If body mass in **g** and bill length in **m**

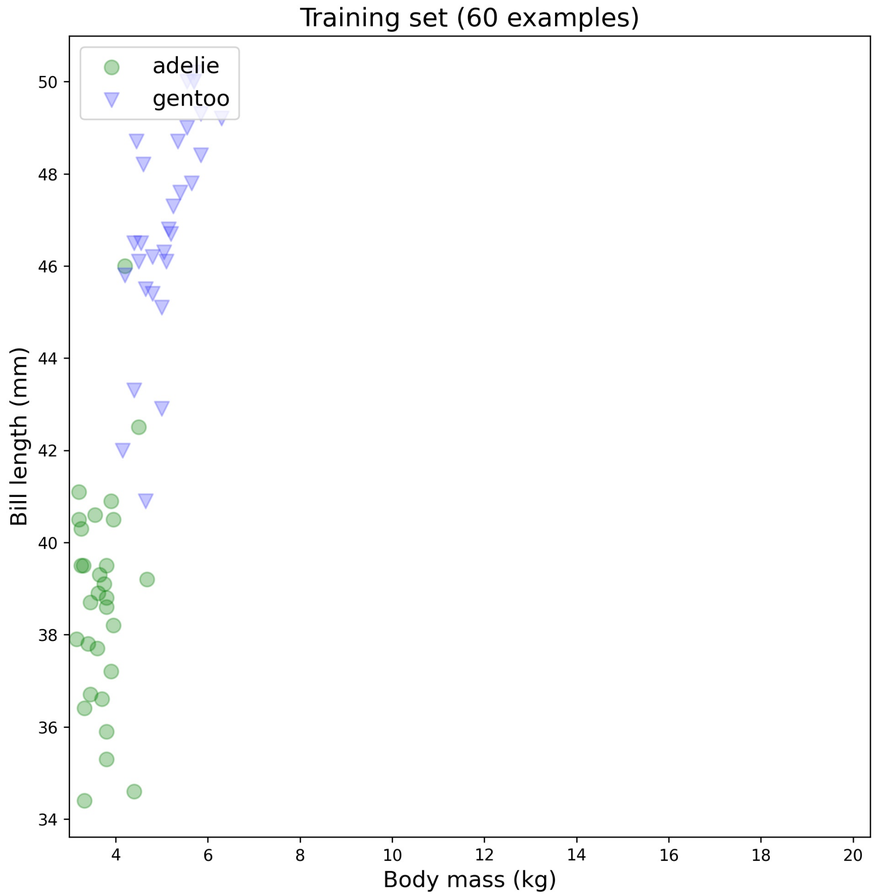
  → **body mass** matters more

With normalization, the units of the features stop playing an important role in the model accuracy

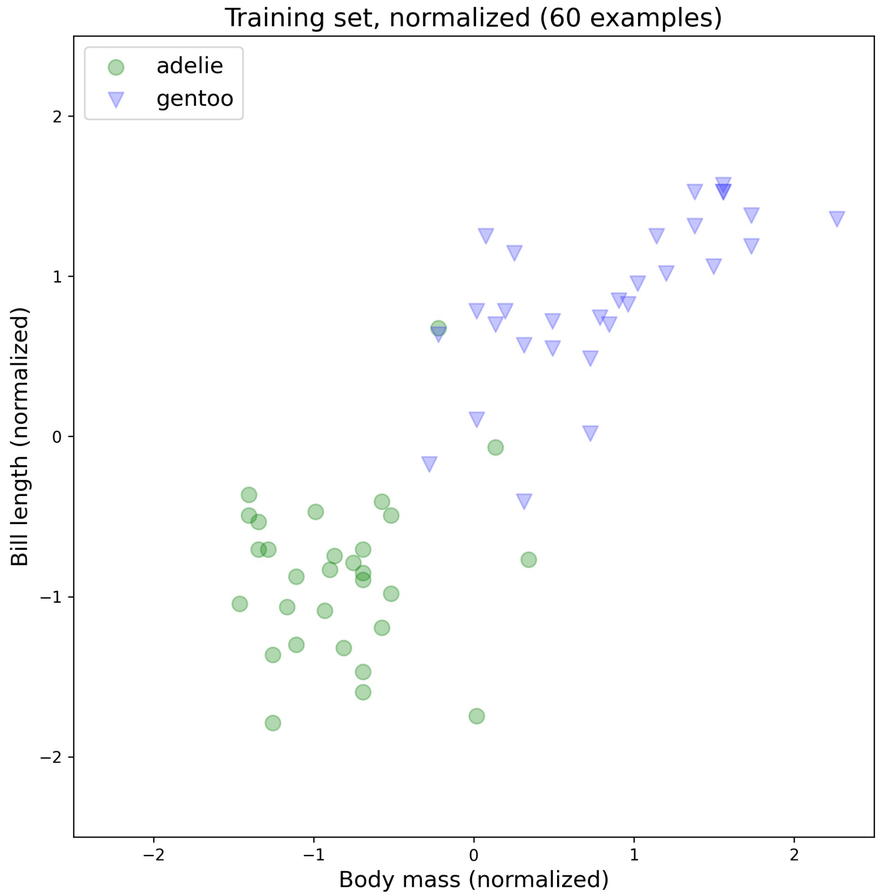


Numerical features

Data normalization - Example



Normalization



Numerical features

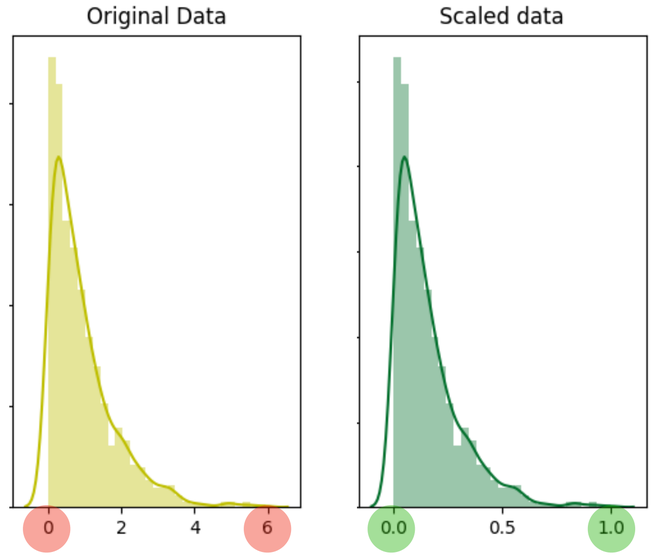
Data normalization - Methods

Min-max scaling:

Scale each dimension of data to the range [0, 1]

For each dimension:

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

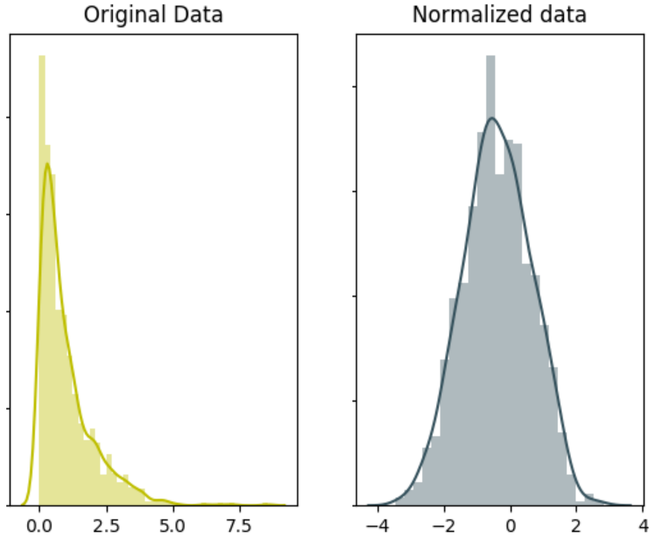


Z-Score Normalization / Standardization:

Set mean of each dimension (**μ**) to 0, standard deviation (**σ**) to 1

For each dimension:

$$x_{norm} = \frac{x - \mu_x}{\sigma_x}$$



Numerical features
Data normalization - Outliers

Features may have outliers or follow some heavy-tailed distribution

- e.g. **urban area population**:
  - most urban areas have a few thousands inhabitants
  - a handful of urban areas have tens of millions of inhabitants (New York, Tokyo, …)

- With **min-max** scaling:
  - typical values are scaled to a very small interval

→ you can use z-score normalization

Numerical features

If the value of a component (dimension/field) is positive and ranges over a wide scale

Logarithmic scaling: take the log of every value

- e.g. $\boldsymbol{x} = [20\ 22\ 120\ 1000\ 1110]$
  $\rightarrow \log(\boldsymbol{x}) = [3.0\ 3.1\ 4.79\ 6.91\ 7.00]$
- Interpretation
  - 20 and 22 are similar
  - 1000 and 1100 are similar
  - 20 and 120 are not similar

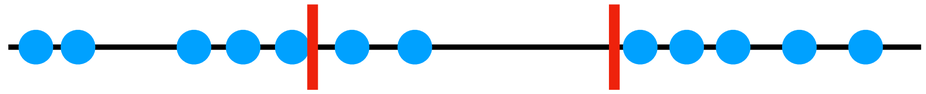
Numerical features
Binning

Binning / discretization: partition continuous features into discrete values

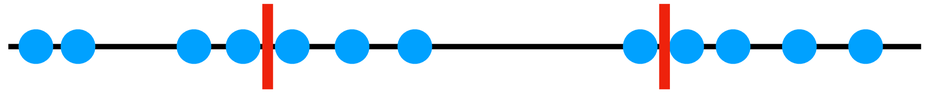
- *e.g.,* **age:** instead of using each person's age as a number, we may want to use age ranges instead: 0-9, 10-19, 20-35, …

Common types of binning:

- **Uniform:** all bins have identical widths



- **Quantile:** all bins have the same number of points



- **Clustered:** a clustering algorithm (seen later in this course) is used to separate values

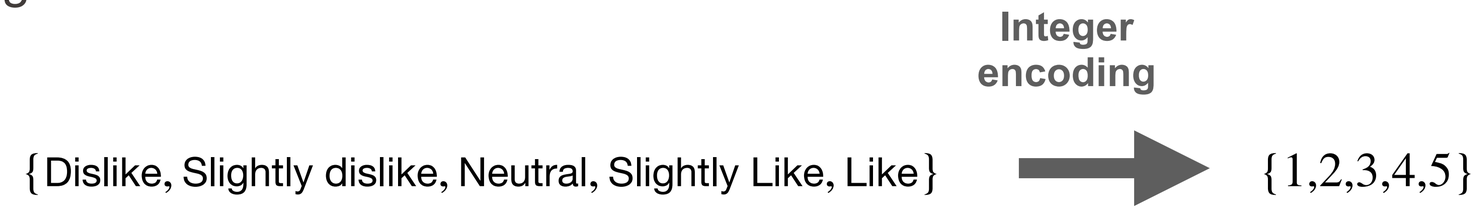
Ordinal features
Overview

Ordinal variables: **finite** set of discrete values, with a **clear ordering** between them

- Examples:
  - Satisfaction *(like, slightly like, neutral, slightly dislike, dislike)*
  - Education level *(high school, BS, MS, PhD)*

Integer encoding: map values to integers ranging from 1 to n

- n = number of unique values of the feature
- Mapping from 0 to n-1 is also used



Categorical features
Overview

Categorical variables: **finite** set of discrete values, with **no intrinsic ordering** between them

- Examples:
  - Animal species *(cat, dog, iguana, penguin, …)*
  - Eye color *(blue, green, brown, …)*
  - Sex *(male / female)*

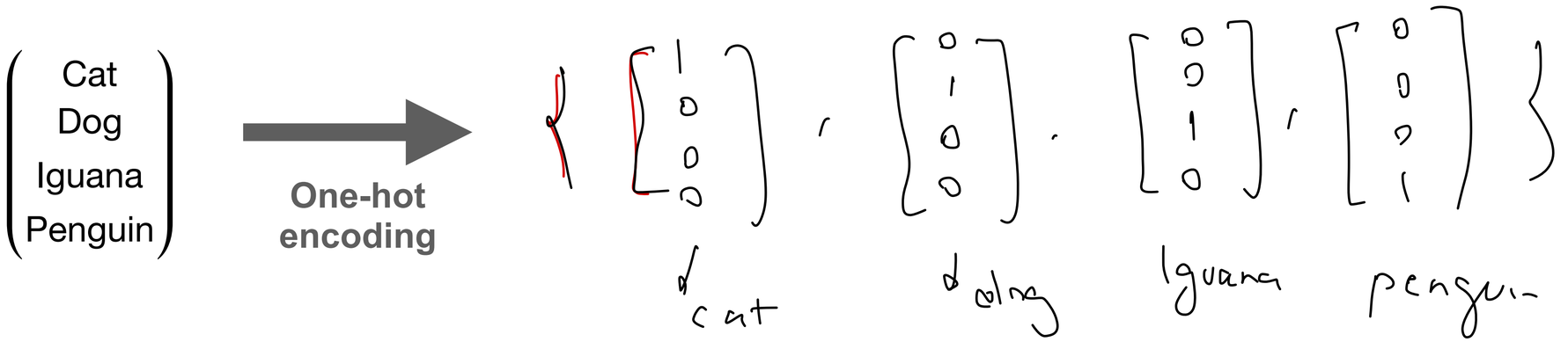
As there is no order between these values, integer encoding may not be appropriate

- e.g. assume [cat, dog, iguana, penguin] is encoded to [1, 2, 3, 4]
  - It wouldn't make sense to imply that a cat is less than an iguana, or that a penguin is closer to an iguana than it is to a cat

Instead, use **one-hot encoding**

Categorical features
One-hot encoding

One-hot encoding: Each category is represented by a binary variable



- One-hot encoding removes any assumption of order between categories
- If there are many categories, the resulting feature vector can be very large and sparse (e.g. suppose you one-hot encode every word in the dictionary)

Missing values
Introduction

Raw data isn't always clean, it is quite common for some values to be missing

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	NaN	41.0	6.984127	NaN	322.0	2.555556	37.88	-122.23
1	NaN	21.0	NaN	0.971880	2401.0	NaN	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	NaN	1.073059	558.0	NaN	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
5	4.0368	52.0	NaN	1.103627	413.0	NaN	37.85	-122.25
6	3.6591	52.0	4.931907	0.951362	1094.0	2.128405	37.84	-122.25
7	NaN	52.0	4.797527	1.061824	1157.0	1.788253	37.84	-122.25

Each row is block group of population
Each column is defined as:

- MedInc median income in block
- HouseAge median house age in block
- AveRooms average number of rooms
- AveBedrms average number of bedrooms
- Population block population
- AveOccup average house occupancy
- Latitude house block latitude
- Longitude house block longitude

Missing values
Overview

Many ML algorithms cannot work with missing values

→ handling missing values properly is very important

How to handle missing data?

Deletion (simplest solution)

- If only a few features have missing values, delete these features (i.e. delete the column)
- If only a few rows have missing values, delete these rows

Imputation

- Replace missing values by another value
- Many different data imputation techniques exist

Missing values
Imputation - Numerical

How to impute data?

For **numerical** data**:**

- **Constant imputation:** Impute with constant value different from all other values in the feature (such as 0)
- **Mean imputation:** Impute with mean of the data along that feature
- **KNN imputation:** Impute with mean of k nearest neighbors to data point (we will discuss KNN approach)

Missing values
Imputation - Numerical

Example: Mean imputation for housing dataset

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup
0	NaN	41.0	6.984127	NaN	322.0	2.555556
1	NaN	21.0	NaN	0.971880	2401.0	NaN
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260
3	5.6431	52.0	NaN	1.073059	558.0	NaN
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467
5	4.0368	52.0	NaN	1.103627	413.0	NaN
6	3.6591	52.0	4.931907	0.951362	1094.0	2.128405
7	NaN	52.0	4.797527	1.061824	1157.0	1.788253

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup
0	4.88852	41.0	6.984127	1.045183	322.0	2.555556
1	4.88852	21.0	6.256710	0.971880	2401.0	2.291188
2	7.25740	52.0	8.288136	1.073446	496.0	2.802260
3	5.64310	52.0	6.256710	1.073059	558.0	2.291188
4	3.84620	52.0	6.281853	1.081081	565.0	2.181467
5	4.03680	52.0	6.256710	1.103627	413.0	2.291188
6	3.65910	52.0	4.931907	0.951362	1094.0	2.128405
7	4.88852	52.0	4.797527	1.061824	1157.0	1.788253

Missing values
Imputation - Ordinal

How to impute data?

For **ordinal** data**:**

- **Constant imputation:** Impute with constant value different from all other values in the feature (such as 0, if encoding starts at 1)
- **Median imputation:** Impute with median of the feature
- **KNN imputation:** Impute with median of k nearest neighbors to data point

Missing values
Imputation - Categorical

How to impute data?

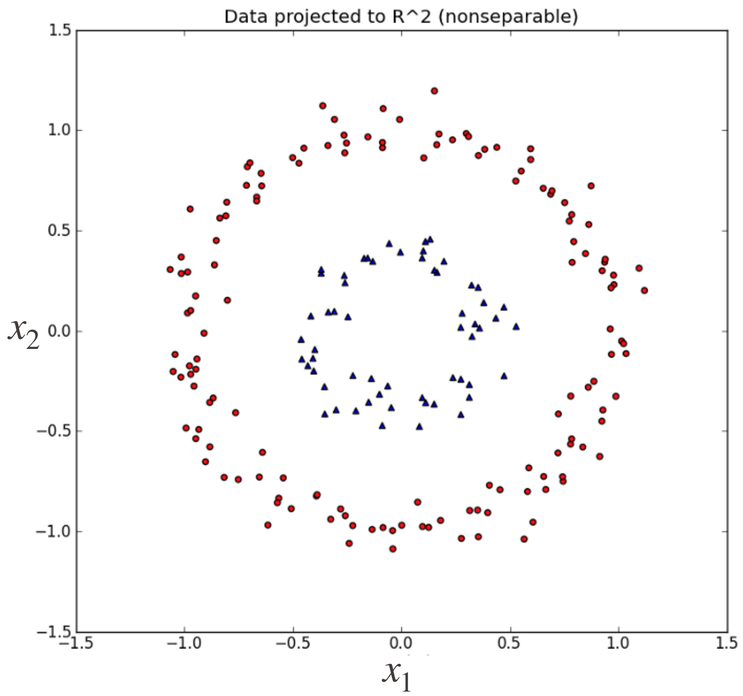
For **categorical** data**:**

- **Add new category:** Add a new category corresponding to “missing value”
- **Mode imputation:** Impute with the mode of the feature (most common category)
- **KNN imputation:** Impute with mode of k nearest neighbors to data point

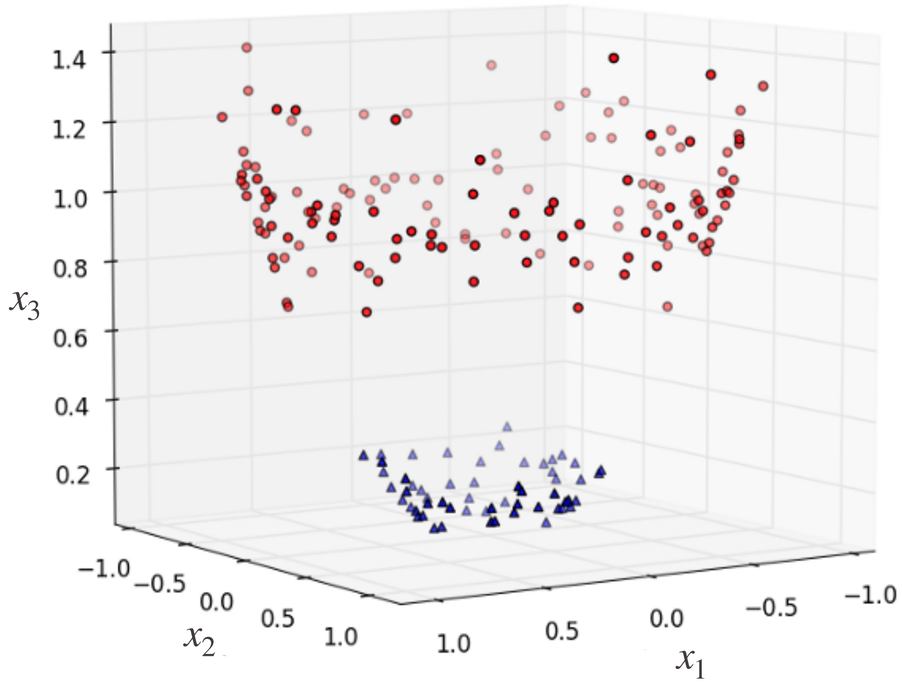
Feature expansion

Creating new features based on existing ones

Example: Let's add a synthetic feature: $x_3 = x_1^2 + x_2^2$



$$\boldsymbol{x}' = (x_1, x_2, x_3)$$
$$= (x_1, x_2, x_1^2 + x_2^2)$$



x_3 encodes a non-linearity in the feature space

By adding x_3, the data becomes linearly separable!

Feature expansion
Overview

Feature expansion is the process of creating derived features from the input data

- Adds complexity at the benefit of
  - Reduces underfitting
  - Can significantly improve performance

Popular feature expansion techniques:

- **Feature crosses:** Multiply features together
- **Binning:** Transform a numerical feature into several categorical or ordinal features
- **Applying a non-linear function to each feature** (*e.g.,* sin, log, polynomials)

Summary and exercise hour announcements

Logistic regression for classification

- Determine a hyperplane/hyperplanes for separating the classes
- Probabilistic interpretation
- Loss function different than performance metric

Feature engineering

- How to best represent your numerical/ordinal/categorial data for the computer
- Need some insight into the data to address it

Exercise hour

- This week: problem sets
- Next week: quizzes