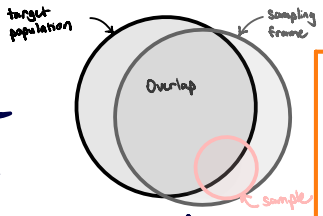
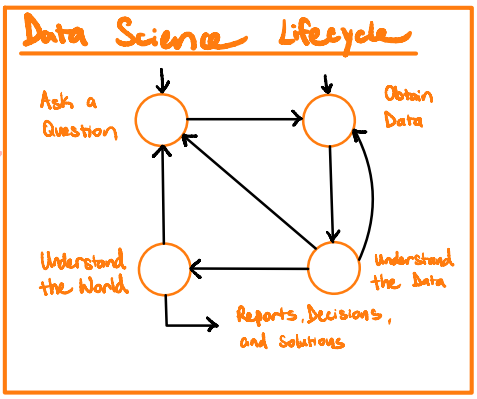
Data 100 Midterm 1 Study Guide — Jeffrey Shen

1 Sampling





Population: The group that you want to learn something about

Sampling Frame: List from which the sample is drawn

Sample: Who you actually end up sampling

Biases

Selection Bias: Systematically excluding particular groups
- ↳ **Avoid by:** sampling frame and method of sampling

Response Bias: People don't always respond truthfully
- ↳ **Avoid by:** nature of questions & method of surveying

Non-Response Bias: People don't always respond, people who don't aren't like ppl who do
- ↳ **Avoid by:** keep surveys short, be persistant

Samples

Simple Random Sample (SRS): sample drawn uniformly at random w/o replacement
Every individual, pair, triple... has same chance of being selected

---

2 Random Variables & Distributions

Random Variables: Variable that can take numerical values w/ particular probabilities

Expectation: weighted avg of values where weights are probabilities

$$\mathbb{E}(X) = \sum_{x \in X} x \, \mathbb{P}(X = x)$$

Linearity of Expectation: Linear transformations apply to expectations

$$\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$$

Distributions

	Expectation	Variance
Bernoulli(p): takes on value 1 w/ probability p & 0 w/ prob $1-p$	$E(X) = p$	$Var(X) = p(1-p)$
Binomial(n, p): # of 1s in n independent Bernoulli(p) trials	$E(X) = np$	$Var(X) = np(1-p)$

Binomial:

$$P(k \text{ successes}) = \binom{n}{k} p^k (1-p)^{n-k}$$

Multinomial n times, proportion p_i:

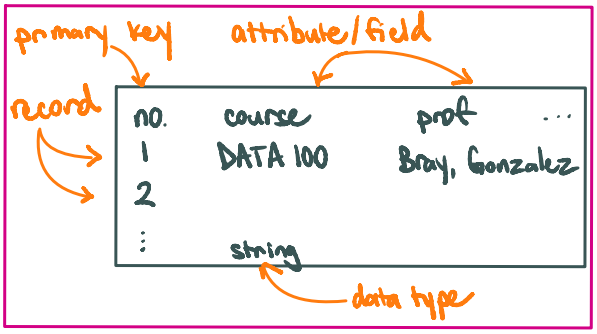
$$P(k_1, k_2, k_3) = \frac{n!}{k_1! \, k_2! \, k_3!} p_1^{k_1} p_2^{k_2} p_3^{k_3}$$

Uniform: probability of each value is $\frac{1}{\text{size of set}}$

③ Databases & SQL

Database Management System (DBMS) vs. CSV

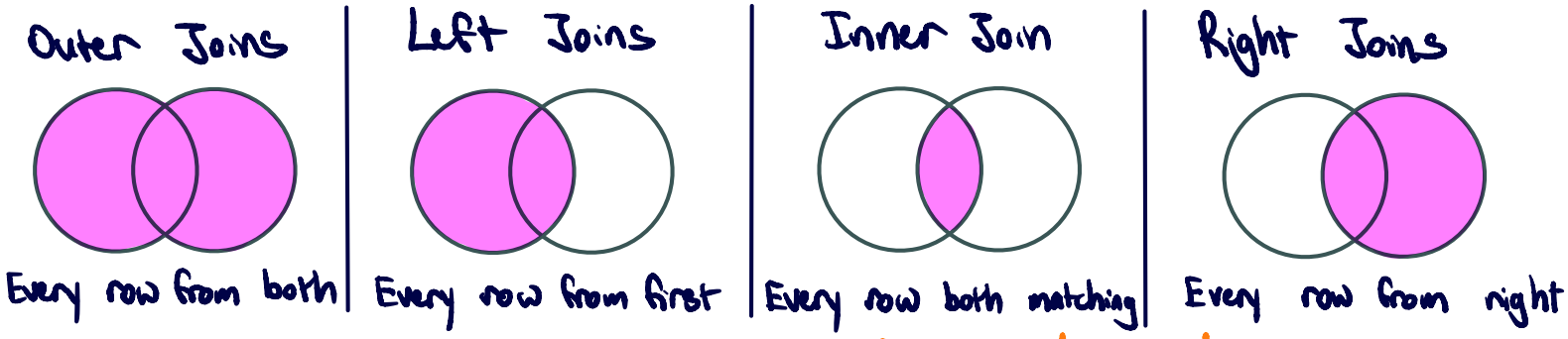
Pros: Reliable storage, optimized, performance
logically organized, safe concurrent operations



SQL

SELECT [DISTINT] <column expression list> FROM <list of tables>
[WHERE <predicate>] [GROUP BY <column list>]
[HAVING <predicate>] [ORDER BY <column list>]
[LIMIT <number of rows>]

Joins



Ex) SELECT * FROM s JOIN t ON s.id = t.id

④ Pandas

Data Structures

Index: row labels

0
1
2
⋮

Series: 1D data

0	Obama
1	McCain
⋮	

Name: Candidate, dtype: obj

Data Frame: 2D tabular data

	Candidate	Party	...
0	Obama	D	
1	McCain	R	
⋮			

[ ] operator

<dataframe>[<column name>] ⟶ Series (column selection)

<dataframe>[< list >] ⟶ Data Frame (column selection)

<dataframe>[<numeric slice>] ⟶ Data Frame (row selection)

Boolean Array Selection

Boolean mask with boolean array for each index in df

<dataframe>[<dataframe>[<column>] <condition>] ⟶ filtered Dataframe

loc / iloc

Note: loc/iloc is inclusive on both ends, [start : stop]

loc: label based accessing

<dataframe>[< row indexes >, <column labels >]

iloc: index based accessing

<dataframe>[< row indexes >, <column indexes >]

Other helpful functions

function	description	return type
<dataframe>.index	range of indexes	Index
<dataframe>.columns	names of cols besides key	Index
<dataframe>[<col>].value_counts()	counts of each value	Series
<dataframe>.sort_values(<col>)	sort by <col>	DF
<dataframe>[<col>].unique()	list of unique items	np array

Groupby / Aggregate

groupby: separates data by given column

<dataframe>.groupby(<col>)

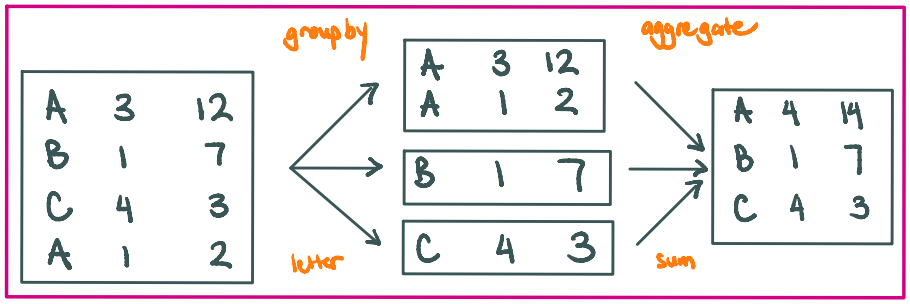
aggregate: condenses sub dataframe back into single row

<groupby>.agg(<function>)

<groupby>.sum()

<groupby>.max()

<groupby>.median()



Pivot Table

<dataframe>.pivot_table(index = <col>, columns = <col>, values = <col>, aggfunc=<f>)

- index: rows of new table
- columns: column values
- values: fields to process in each group
- aggfunc: group operation

Pandas Joins

<dataframe>.merge(<df2>, how=<str>, left_on=<df_col>, right_on=<df2_col>)

join type : 'inner' / 'outer' / 'left' / 'right'

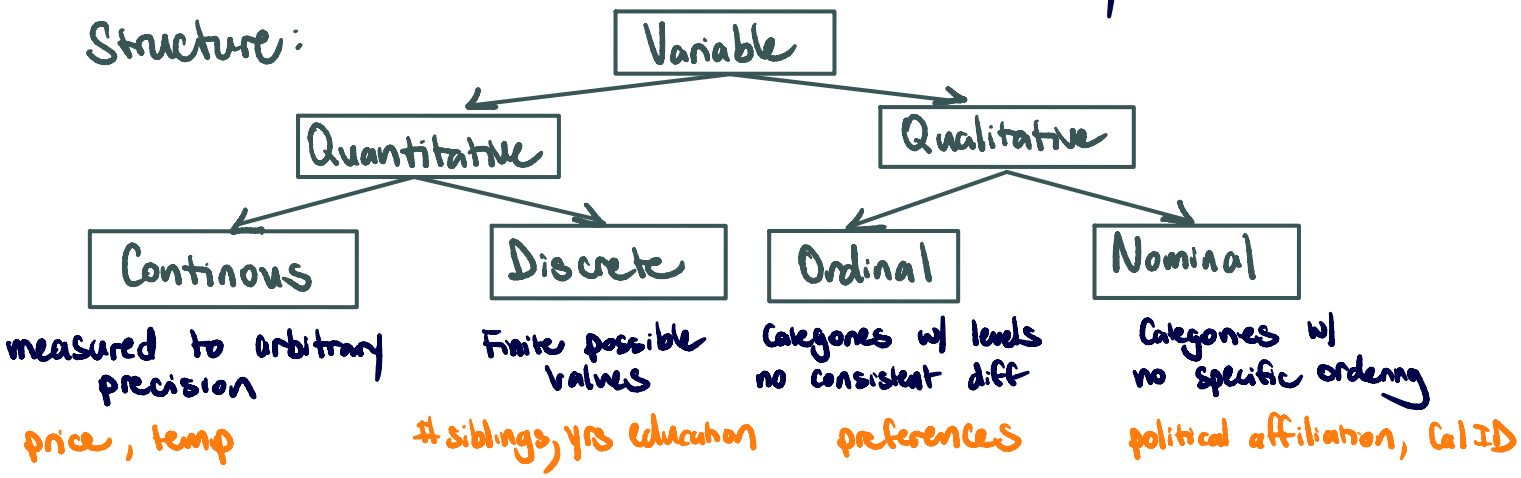
---

⑤ Exploratory Data Analysis (EDA)

Key Properties in EDA

1. Structure — "shape" of a data file (csv, JSON)
2. Granularity — how fine/coarse is data (individual vs. group)
3. Scope — how (in)complete is data (cover area of interest, too expensive)
4. Temporality — how is data situated in time (data changes over time)
5. Faithfulness — how well does data capture reality (unrealistic, missing)

Structure:



---

⑥ Regular Expressions (RegEx) / Strings

Canonicalization

<str>.replace(<old_str>, <new_str>)

<str>.lower()

RegEx

Python re library for regex

re.findall(<regex pattern>, <text>)	return list of all instances
re.sub(<regex pattern>, <text>	substitutes pattern
r"<regex>"	reduces backslashes

Regex Reference

^	^ark	match beginning of string
$	ark$	match end of string
?	joh?n	match 0 or 1 times
+	jo+hn	match at least 1 time
*	AB*A	match 0 or more times
.	.U.U.	match any character
|	AA|BA	match left or right pattern
[ ]	[a-z]	match any of chars inside
( )	(AB)*A	group patterns
\b		boundary between words
\w		"word" (letters, digits, underscore)
\s		whitespace
\d		digits
[^]	[^a-z]	character class negation
\	cow\.com	escape character

---

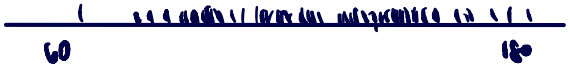
⑦ Visualizations

Accuracy of our judgements depend on type of marking

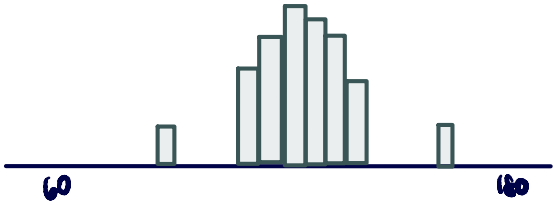
Generic ← color hue, color intensity, angle, volume, area, slope, length → Accurate

Graphs

Rug Plot: Shows every quantitative variable



Histograms: smoothed rug plot, areas are proportions, x axis divided into bins

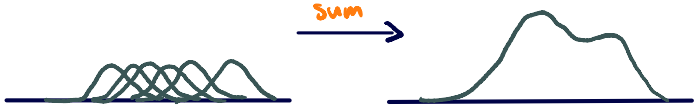


Bin width $= 2\frac{IQR(x)}{\sqrt[3]{n}}$

Kernels: valid density function

1. Place kernels at each data point
2. Normalize kernels so area = 1
3. Sum all kernels

α: hyperparameter ↑ α ↑ smooth $f_\alpha(x) = \frac{1}{n}\sum_{i=1}^{n} K(x, x_i)$

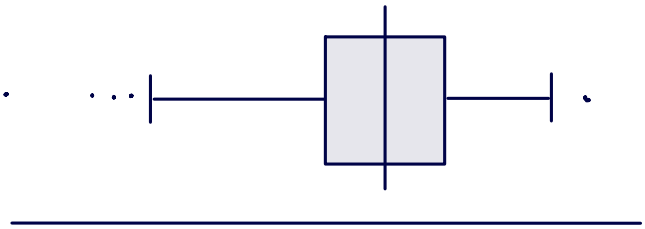


Gaussian: $K_\alpha(x, x_i) = \frac{1}{\sqrt{2\pi\alpha^2}} e^{-\frac{(x-x_i)^2}{2\alpha^2}}$

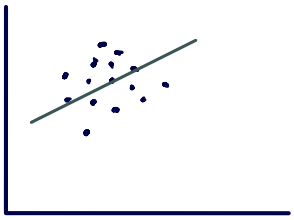
Box Plots: Summarize several characteristics

whiskers, lower quartile, median, upper quartile, outliers

↑ 1.5 * IQR ± Q1/Q3

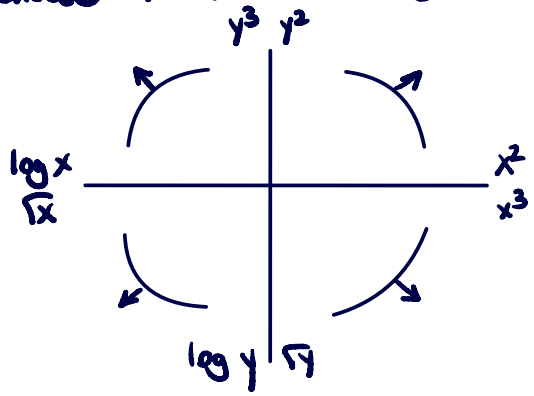


Scatter Plots: Plotting 2 quantitative variables on same plot



Tukey-Mosteller Bulge Diagram

Help choose transformations to linearize



8 Modeling

Model: useful simplification of reality allows us to understand the world we live in & predict the value of unseen data

Modeling Process

1. Choose a Model (constant, linear, non-linear)
2. Choose an Objective Function (loss function)
3. Fit model by optimizing Objective Function (analytical, numerical)

Loss function

L_2 squared loss: $(y - \hat{y})^2$ ← mean of dataset minimizes

L_1 absolute loss: $|y - \hat{y}|$ ← median of dataset minimizes

Want to reduce average loss across all points

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i)$$

with L_2: Mean Squared Error (MSE) — smooth

with L_1: Mean Absolute Error (MAE) — robust to outliers

Model

Constant Model: $\hat{y} = \theta$

$\hat{\theta}_0 = \bar{y} - \hat{\theta}_1 \bar{x}$

Simple Linear Regression: $\hat{y} = \theta_0 + \theta_1 x$

Multiple Linear Regression: $\hat{y} = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p = \theta_0 + \sum_{j=1}^{p} \theta_j x_j$

$\hat{\theta}$ is value that minimizes avg loss

1. Can take derivative of avg loss w/ respect to $\theta / \hat{y}$
2. set to 0, solve for $\theta / \hat{y}$

When model has intercept term:
1) Sum (mean) of residuals is 0
2) Avg true y value = avg predicted y

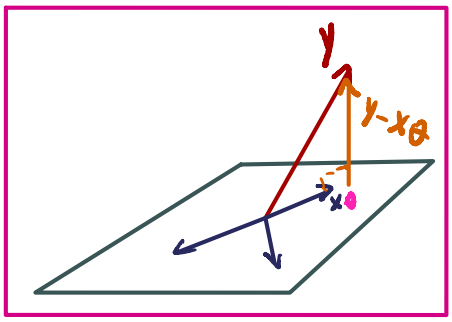
$$\hat{Y} = X\theta$$

vector w/ dimensions $n \times 1$

matrix w/ rows as observations, cols as features

$$\begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$$

vector w/ optimal $\hat{\theta}$ dimension $(p+1) \times 1$



Least Squares Regression: want θ where residual is orthogonal to span(X),

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

if $X^T X$ is full rank

Predicted response in span(X), orthogonal to residuals $\hat{y}^T e = 0$

Using optimal parameter vector, $X^T e = 0$

Data 100 Final Study Guide

Jeffrey Shen

1 Feature Engineering

transform raw features to more informative features

capture domain knowledge, express non-linear relationships, encode non-numeric features

One-hot Encoding: transform categorical feature into binary features

State	AK	...	CA	...	NY	...	WA	...	WY
NY	0	...	0	...	1	...	0	...	0
CA	0	...	1	...	0	...	0	...	0

Bag-of-words: encode as vector of word counts

words	a	...	fun	...	learning	...	z
vector	0	...	1	...	2	...	0

N-gram encoding: condition on previous N-1 words

2 Bias and Variance

Variance: expected squared deviation from expectation of X

$$Var(X) = E((X-E(X))^2) = E(X^2) - (E(X))^2 = SD(X)^2$$

Linear Transformations: $Var(aX+b) = a^2 Var(X)$, $SD(aX+b) = |a| SD(X)$

Standardization: $X_{su} = \frac{X-E(X)}{SD(X)}$

Covariance: expected product of deviations

$$Cov(X,Y) = E((X-E(X))(Y-E(Y))) = E(XY) - E(X)E(Y)$$

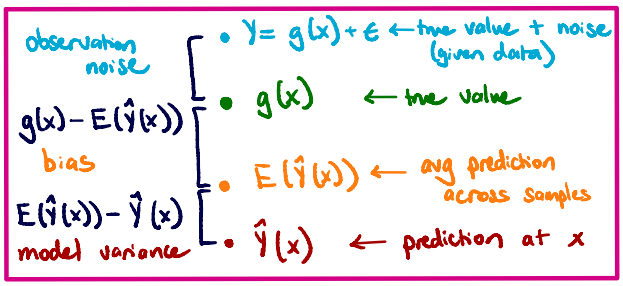
Correlation: avg product in standard units, covariance scaled by the 2 SD

$$r(X,Y) = \frac{Cov(X,Y)}{SD(X)SD(Y)}$$

Bias: non-random error, due to model different from underlying function g

Chance Error: due to randomness

Model Risk



$$E[(Y-\hat{Y}(x))^2] = \underbrace{\sigma^2}_{\text{noise}} + \underbrace{[E(\hat{Y}(x)) - g(x)]^2}_{(\text{bias})^2} + \underbrace{E[(\hat{Y}(x) - E(\hat{Y}(x)))^2]}_{\text{model var}}$$

3 Cross Validation and Regularization

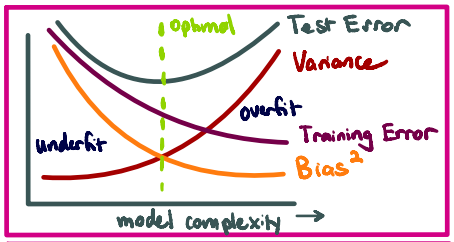
Training data: used to fit model

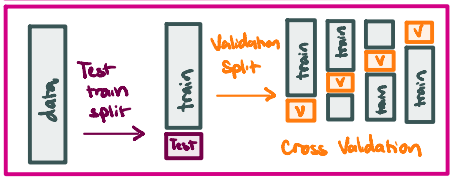
Validation set: test generalization while training

Regularization: control complexity

$$\theta = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} Loss(y, f_\theta(x_i)) + \lambda R(\theta)$$

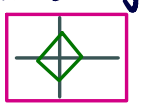
For each dimension: $z_k = \frac{x_k - \mu_k}{\sigma_k}$





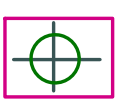
L1: Lasso Regression: choose specific feature weights

min avg squared loss + L1 penalty



L2: Ridge Regression: balance features

min avg squared loss + L2 penalty



4 Gradient Descent

optimization algorithm to find min/max

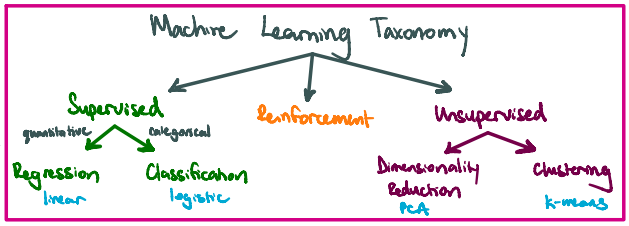
Gradient Descent: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \nabla_\theta L(\theta, X, \vec{y})$ where α is learning rate

Stochastic Gradient Descent: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \left(\frac{1}{|B|} \sum_{i \in B} \nabla_\theta L_i(\theta) \Big|_{\theta=\theta^{(t)}} \right)$ where B is rand subset of indicies

5 Logistic Regression

used for binary classification, est probability w/ linear func

Sigmoid: $\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t}$ $\quad \frac{d}{dt}\sigma(t) = \sigma(t)(1-\sigma(t))$



Mean Cross-Entropy Loss: empirical risk for logistic func

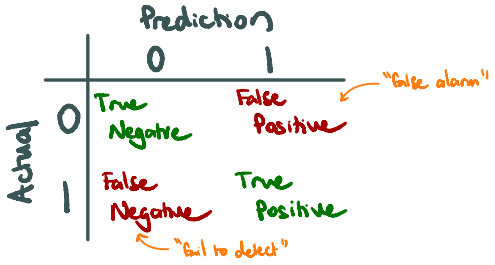
$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \log(\sigma(x_i^T\theta)) + (1-y_i)\log(1-\sigma(x_i^T\theta))\right)$$

$$\text{loss} = \begin{cases} -\log(1-\hat{y}) & y=0 \\ -\log(\hat{y}) & y=1 \end{cases}$$

Want to min cross-entropy loss

Linear Separability: separable if can draw degree-1 hyperplane to separate

Classification Errors



Accuracy: classified correctly

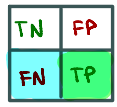
$$\text{accuracy} = \frac{TP+TN}{n}$$

Precision: penalize false positive
↑threshold ↑precision

$$\text{precision} = \frac{TP}{TP+FP}$$

Recall: penalize false negative
↓threshold ↑recall

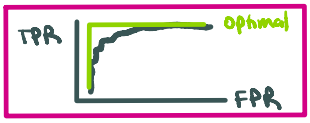
$$\text{recall} = \frac{TP}{TP+FN}$$



$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN}$$

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP+FN}$$

Reciever Operating Characteristic (ROC): plot TPR vs. FPR



6 Decision Trees

simple nonlinear classifier but susceptible to over-fitting

Fits perfectly on training data, with unclear boundary, becomes messy & overfit

Entropy: uncertainty about variable ↑entropy ↑uncertainty

$$S(V) = -\sum_k P(V_k)\log_2 P(V_k)$$ k: class

Weighted entropy as loss func: $L = \frac{N_1 S(X) + N_2 S(Y)}{N_1 + N_2}$

Random Forest: prevent overfitting by building multiple trees and voting

Bagging: Bootstrap AGGregatING, generating bootstrap resamples of training data

↑individual tree variance
↓overall forest variance

7 Inference for Modeling Classification

Inference: drawing conclusions about population parameters given a sample

Parameter θ^*: function of population

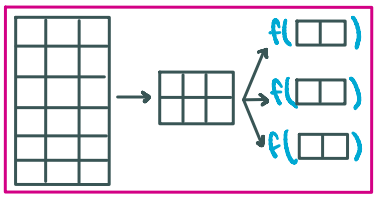
Estimator $\hat{\theta}$: function of sample, want to estimate a population parameter

Sampling Distribution: distribution of estimator values, across all samples, unknown

Bootstrap resampling: estimate sampling distribution by sampling from sample distr.

Confidence Intervals: takes sample returns interval contain θ^* for p% of samples

Multicollinearity: a feature that can fairly accurately be predicted from linear combination of others



---

8 Principal Component Analysis (PCA)

unsupervised learning dimensionality Reduction

Use singular value decomposition (SVD) to extract eigenvalues & eigenvectors X^TX

Principal Components: lines to project data onto, new features **score** is transformed coord

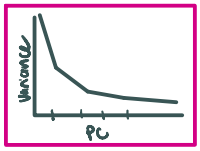
Singular Value Decomposition (SVD):

$$X = U\Sigma V^T$$

U: orthonormal
V: orthonormal
Σ: diagonal, singular values

PCA for EDA

1) Recenter data — `data - np.mean(data, axis=0)`
2) Use SVD — `np.linalg.svd(centered_data, full_matrices=False)`
3) Find PCA matrix $P = U\Sigma$ — `u @ np.diag(s)`



Scree plot

---

9 Clustering

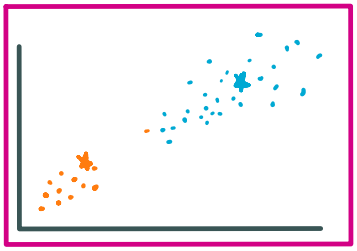
unsupervised learning to identify patterns in unlabeled data

K-means Clustering assign point to one of k clusters

1) Pick k and place k centers

Repeat until convergence

2) Color points according to closest center
3) Move center for each color to center of points w/ color



K-means minimizes **inertia**: sum of squared distances from each data pt to center
or **distortion**: weighted inertia

Silhouette Score: metric for single point $S = \frac{B-A}{\max(A,B)}$ A,B: avg dist to other/own cluster

Agglomerative Clustering: k-means optimized for distance not blobbiness

1) Start w/ every data point in own cluster
2) Combine two closest clusters until k clusters remaining, based on min/max/avg

---

10 Big Data

large datasets beyond typical SW tools to manage, use parallel computing

Extract, Transform, Load (ETL) for dealing with data

Extract and Loading provides snapshot of data
Transform involves cleaning, preparing data for analytics, hard

Fact Table: minimize redundant info, reduce data error
store labels for data in other tables

Dimension Table: Summarize encoded info through multidimensional table

Data Lakes: Store copy of all data in original form
- lots of noise, dirty data

Fact Table

pid	locid	sales
11	1	25
12	1	8
13	3	15

Products

pid	pname	price
11	corn	25
12	Galaxy 1	18

Locations

locid	city	State
1	Omaha	NE

Fault-Tolerant Distributed File Systems: split file into smaller parts and keep multiple copies

Map Reduce programming model for distributed data processing

Map: separates out function locally to compute
- deterministic: reexecution on failure

Reduce: combines values of one key
- commutative, associative: allows for reorder & regrouping of operations

Apache Hadoop → Apache Spark 100x faster

	Hadoop	Spark
Storage	Disk-only	Disk or In-memory
Operation	map reduce	map-reduce, join, sample
Execution Model	Batch	Batch, interactive, streaming
Programming env	Java	Scala, R, Python