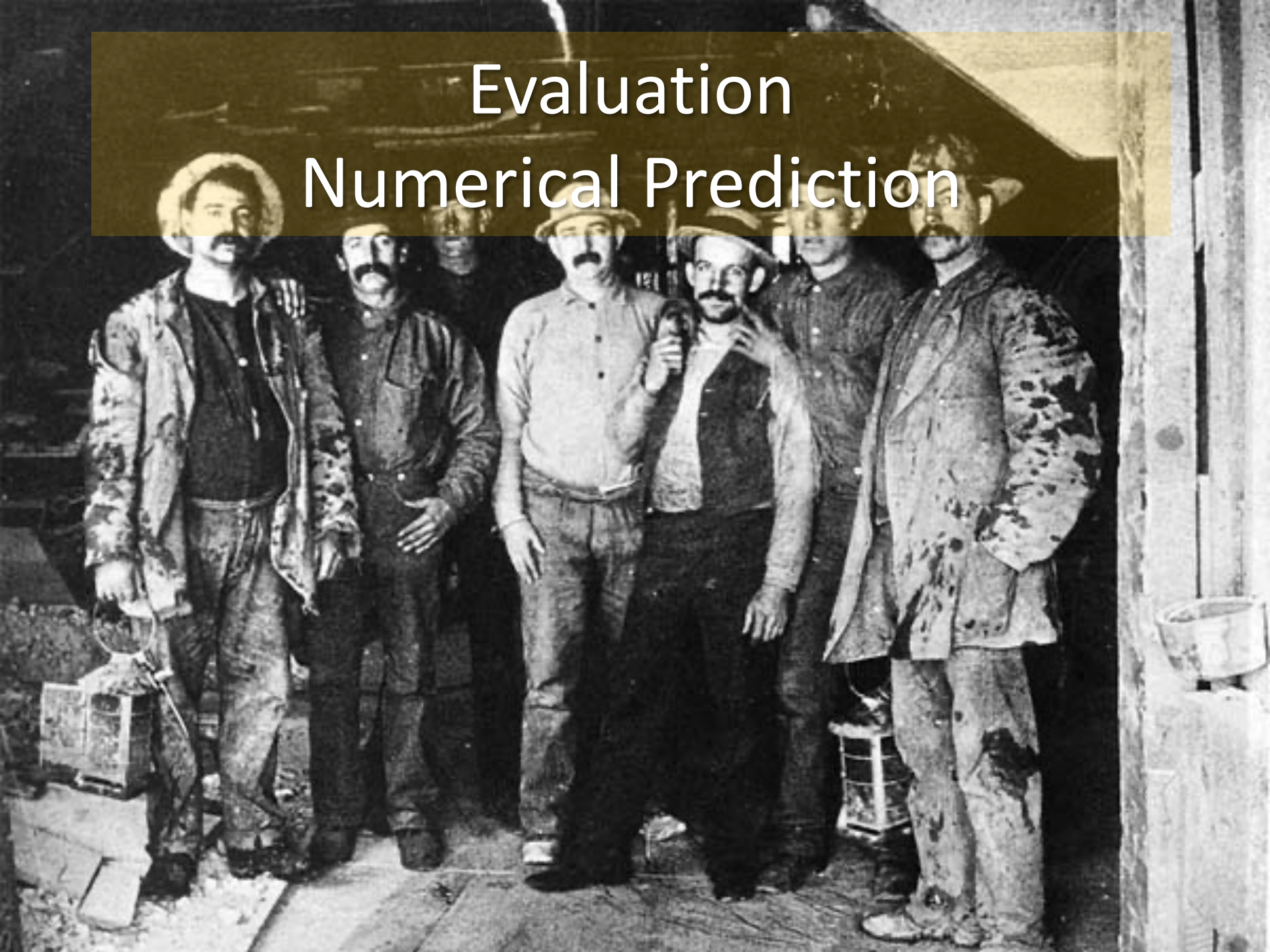


Evaluation
Numerical Prediction

Evaluation

- Why do we evaluate performance?
  1. Evaluate generalizability
  2. Hyperparameter optimization
  3. Algorithm selection
- We need to:
  - Split our data
  - Define performance metrics

Splitting the data

- Methods:
  - Without replacement:
    - train, validation and test sets
    - k-fold cross-validation
    - leave-one-out cross-validation
  - With replacement:
    - 0.632 bootstrap

Training & test sets

- Training and testing on the same data → bias!
- Main idea is that models are
  - built on one part of the data (**training set**), and
  - validated/optimised on another part (**validation set**)
  - tested on the rest (**test set**)
- Training set
  - Usually around 2/3 of the data (of which we can use part as validation set)
- Test set
  - 1/3 of the data
  - Gives us some ideas about the performance of our model for new data

K-fold cross validation

- Typical k values: 5 and 10
- Idea:
  1. Select $^1/_5$ of the data and set it aside for evaluation
  2. Train the classifier on the rest ($^4/_5$)
  3. Test on the selected $^1/_5$
  4. Remember the evaluation scores
  5. Set aside another $^1/_5$ for testing, and repeat the above
  6. Altogether, there will be 5 rounds of training+testing
  7. Take the average of the eval scores

	Attrib 1	Attrib 2	Class
5	3	R	Y
	4	T	Y
4	5	R	Y
	4	R	N
3	3	E	Y
	4	R	N
2	5	R	Y
	5	T	N
1	5	R	N
	3	E	Y

Leave-one-out cross-validation

- Extreme case of k-fold cross-validation
- **K = n**, number of examples in the dataset

Train the classifier on **all but one** examples

Test the prediction on **one** example

Repeat this n times

Record the percentage of correctly classified instances

- Costly: $O(n^2)$ steps

A frequent student scenario

(based on historical data)

Lecturer: *"How did you evaluate your classification algorithm?"*

Student: *"I used 5-fold cross validation."*

Lecturer: *"Good! Could you tell what your model looks like?"*

Student: *"Hm...for which fold?"*

	Attrib 1	Attrib 2	Class
5	3	R	Y
	4	T	Y
4	5	R	Y
	4	R	N
3	3	E	Y
	4	R	N
2	5	R	Y
	5	T	N
1	5	R	N
	3	E	Y

The final model is always trained on the **whole** dataset!

See textbook, page 146, paragraph 2, sentence 1

0.632 bootstrap: with replacement

- Let us consider a dataset of ***n* instances**
- Take ***n* samples** *with replacement*
- **Picked 0 times** at the end → testing
- 1/n → 1-(1/n) → $(1-(1/n))^n$ → e^{-1} = 0.368 → 1 - 0.632!
- Training/testing – 63.2%/36.8%

Sampling

- How do we select data into the folds?
- We sample from the full dataset
  - Can sample in multiple ways
  - Can also sample in a stratified way
    - Guarantee certain ratio in the sample
    - Can also be used to amplify certain groups

Sampling strategies (examples)

- X-val: Suppose we have 100 instances (rows):

1. Rows 1-10 → ***fold 1***, 11-20 → ***fold 2***, ...
2. Random 10 → ***fold 1***,...
   - What goes into ***fold 2***???
3. Sample so that
   - **DISTRIBUTION** of data in **training set** = **DISTRIBUTION** of data in **test set**
   - i.e., Distr. in ***fold 1*** = distr. of remaining 9/10th of data
   - Stratified sampling

Am I good? Am I better than you?

- My own performance
  - Suppose my eval. score is 0.3415
  - Can I trust this number?

Confidence intervals

- My performance vs. Yours
  - Your eval. score is 0.3511
  - Are you really better than me?

Significance tests

Partly based on Raschka, S., Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

Confidence intervals

- How certain are we about a performance measurement?
  - Discuss performance metrics in more detail later
- We can provide confidence intervals
  1. Measure performance on the test set once and estimate confidence interval
  2. Bootstrap and estimate the confidence interval

Estimating confidence interval

- For classification:
  - Each instance we predict in the test set is a Bernoulli trial
  - Predictions follow a binomial distribution distribution $X \sim (n, p)$
    - n: number of test instances
    - p: probability of success
  - For example: classifier is 50% correct (p = 0.5) on 1000 cases

Confidence at 95%

pop. size	lower	upper
freq = 0.03:		
100	0.0062	0.0851
1000	0.0203	0.0425
10000	0.0267	0.0335
100000	0.0289	0.0310
freq = 0.10:		
100	0.0490	0.1762
1000	0.0821	0.1202
10000	0.0941	0.1060
100000	0.0981	0.1018
freq = 0.50:		
100	0.3983	0.6016
1000	0.4685	0.5314
10000	0.4901	0.5098
100000	0.4968	0.5031

Estimating confidence interval

- Can also approximate it through a normal distribution, we focus on accuracy (ACC_s)

$$ACC_S \pm z\sqrt{\frac{1}{n}ACC_S\,(1-ACC_S)},$$

- For 95% CI, z = 1.96

Bootstrapping for CI

- We select different training/test sets

- We again assume a normal distribution

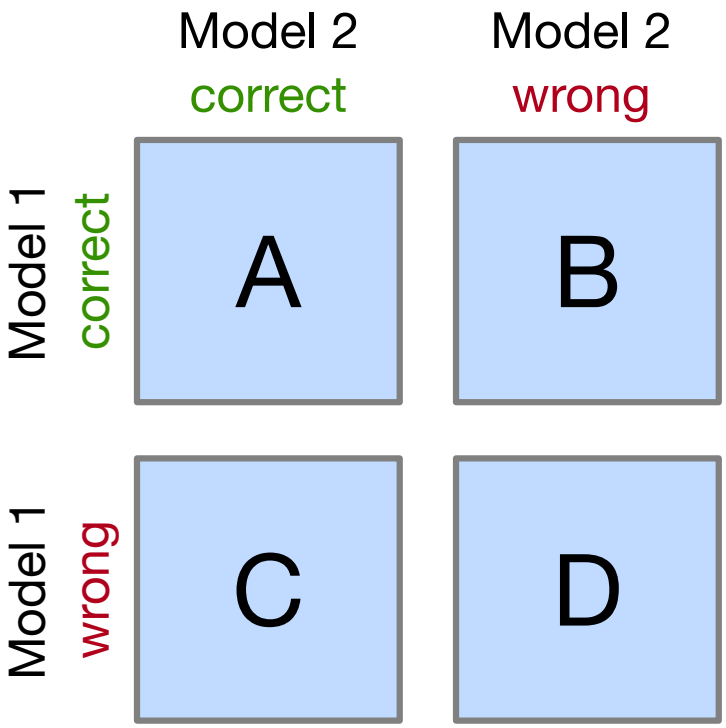
Bootstrapping for CI

- The accuracy over the *b* bootstraps (50-200 recommended) is: $\mathrm{ACC}_{boot} = \frac{1}{b}\sum_{i=1}^{b}\mathrm{ACC}_i$
- The standard error is: $\mathrm{SE}_{boot} = \sqrt{\frac{1}{b-1}\sum_{i=1}^{b}(\mathrm{ACC}_i - \mathrm{ACC}_{boot})^2}$
- And the CI is: $\mathrm{ACC}_{boot} \pm t \times \mathrm{SE}_{boot}$
- t depends on the sample size (i.e. *b*)

Which one is better?

- We have computed the CI's and can exploit them (do they overlap)
  - Assumptions on normality should hold
  - Shown to have a high false positive rate (showing a difference while there is none)
- Alternative: McNemar test (or within subject chi squared test)
- And there are many more……
  - Also based on folds in x-val

Make a table like this



McNemar test

- We compute the test statistic:

$$\chi^2 = \frac{(B - C)^2}{B + C}$$

- Set a confidence value (e.g. $\alpha = 0.05$), compute the p-value using the chi-square distribution
- p-value below observed value: reject hypothesis that performances are equal

Regression problems

- What if we do not have an accuracy?
- We collect prediction errors for each instance
  - Results in means and standard deviation of errors
  - Can exploit this for CIs and significance tests
- Can still use the bootstrap method

Moving on...

- Now we have a frame for evaluation, and
- A way to determine significance
- Let us see what and how we can measure
  - For classification problems
  - For regression problems

Classification: the simplest ones

- You know them of course
  - Number of correctly classified instances
  - Classification rate:

    #correct/#all_cases (aka. *success rate* or *accuracy*)
  - Error rate:

    #errors/#all_cases = 1-classification rate

True/false positives/negatives

- Make the right kind of mistake

	Predict **Plane**	*Predict* **Bird** (not plane)	
It is really a **Plane**	**True positive** (The enemy is attacking, and you dispatch the fighters)	**False negative** (7/12/1941)	Positive reference TP + FN
It is a **Bird** (not plane)	**False positive** (False alarm)	**True negative** (yes, it's just a bunch of *Nēnē* flying)	Negative reference FP + TN
	Positive response TP + FP	Negative response FN + TN	Total TP+ TN + FP + FN

What might describe quality

- Count TP, FP, TP+FP, etc.
- Familiar? → correct/total
- Recall
- Precision
- F-measure

$$\frac{TP + TN}{TP + TN + FP + FN}$$

$$R = \frac{TP}{TP + FN}$$

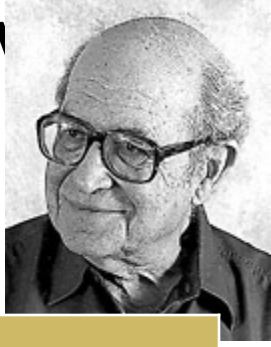
$$P = \frac{TP}{TP + FP}$$

$$\frac{(1 + \beta^2)PR}{\beta^2 P + R}$$

+ many others, and not only for two classes...

Have a look: https://scikit-learn.org/stable/modules/classes.html#classification-metrics

(Jacob) Cohen's Kappa



- Developed to measure agreement
- Predicted vs. Actual *is* a sort of agreement
- *Kappa = (Observed_agr. – Expected_agr.)/(1-Expected_agr.)*

Kappa index	Agreement
-1	Complete disagreement
0.00	Expected by chance
0.00 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost perfect

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$

$$\Pr(a) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\Pr(e) = \frac{TP + FP}{TP + TN + FP + FN} \cdot \frac{TP + FN}{TP + TN + FP + FN} + \frac{TN + FN}{TP + TN + FP + FN} \cdot \frac{TN + FP}{TP + TN + FP + FN}$$

Non-binary classes

- Confusion matrix
- Higher numbers in the diagonal ➔ good

A Weka confusion matrix for the Iris dataset

a	b	c	<-- classified as
50	0	0	| a = Iris-setosa
1	45	4	| b = Iris-versicolor
0	4	46	| c = Iris-virginica

Kappa = 0.91

Costs in evaluation

- If costs of TP/TN/FP/FN can be measured then...
  - 1 false negative (Plane vs. Bird) can be counted as 1000, etc.

Predicted (right) Actual (down)	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

Predicted (right) Actual (down)	A	B	C	D
A	0	0	4	12
B	1	0	3	5
C	2	0	0	6
D	1	1	9	0

Costs in training

- Predict class with lowest expected cost

Predicted (right) Actual (down)	A	B	C	D	
A	0	0	4	12	$P_A = 0.1$
B	1	0	3	5	$P_B = 0.2$
C	2	0	0	6	$P_C = 0.3$
D	1	1	9	0	$P_D = 0.4$

1.2	0.4	4.6	4

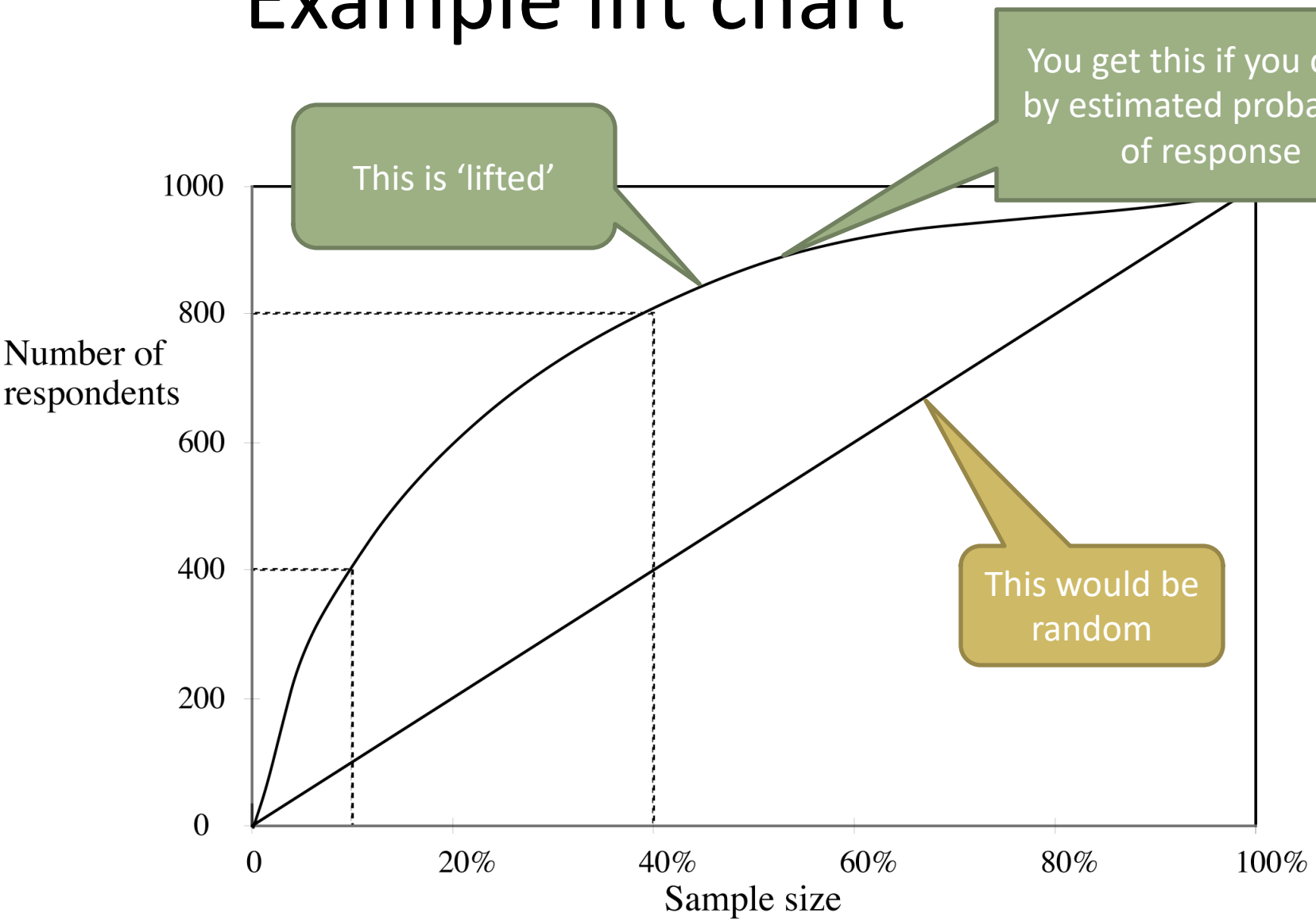
Cost-trick in training

- Suppose
  - **TP** is when you say to a Plane that it is a Plane, and that
  - **FN** *(Plane when you say it's a bird)* costs 500 times as much as **FP** *(false alarm)*
- Then
  - You increase the Plane cases 500-fold in the dataset (e.g. by duplication)
  - Your model will avoid **FN**-s very much
  - You can just pick the highest probability *(if applicable)*
  - You can just use a binary table for evaluation

Lift charts and ROC curves

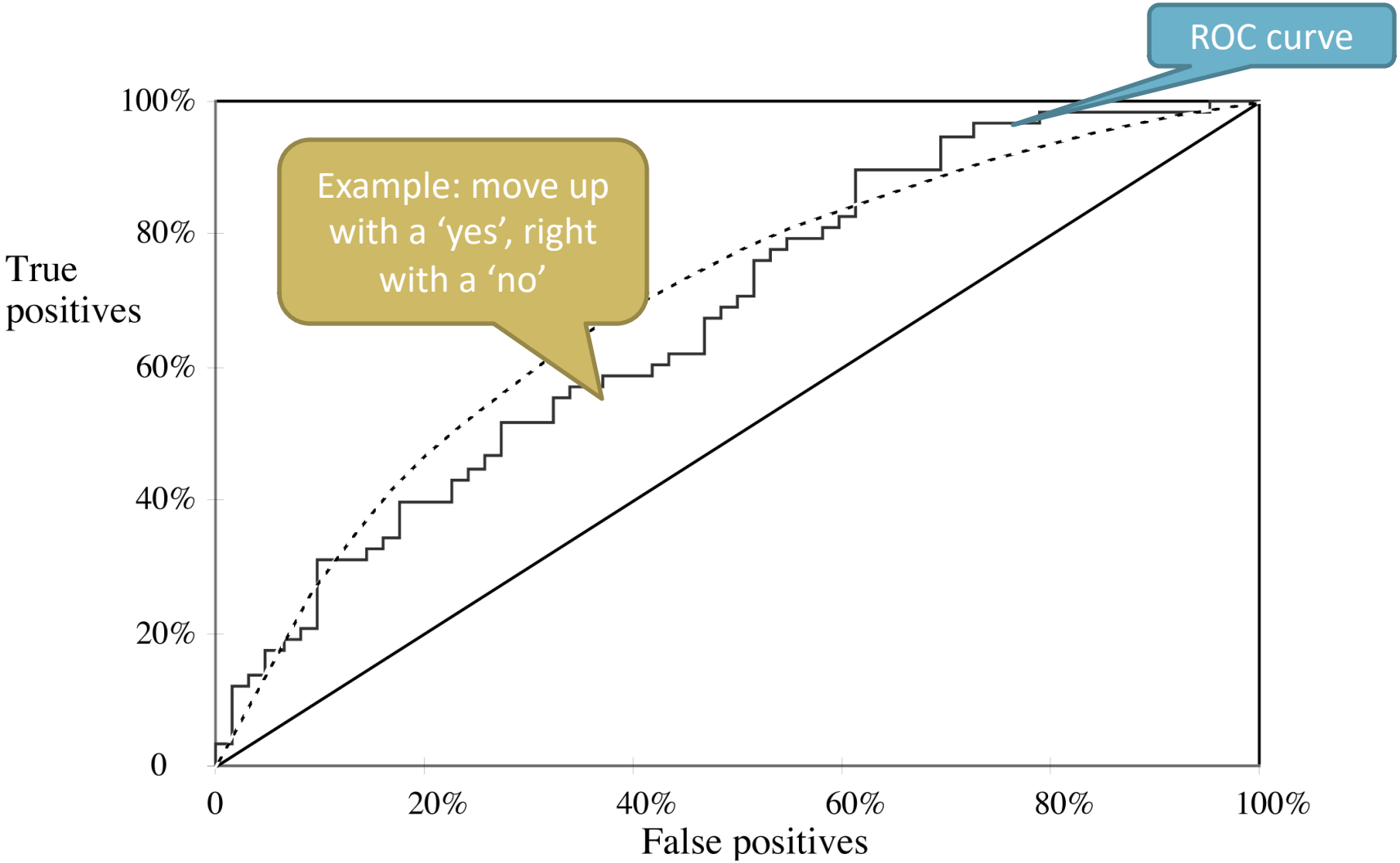
- **Scenario:**
  - Direct mailing
  - 1.000.000 households
  - 0.1% – average response rate
- **Your task:**
  - increase response rate by ***targeting***
- **Client's task:** to decide which one is better:
  - 0.4% response rate for 100.000 households or
  - 0.2% response rate for 400.000 households?
- **Your task:** help them to decide!

Example lift chart



ROC curve

(Receiver Operating Characteristic)

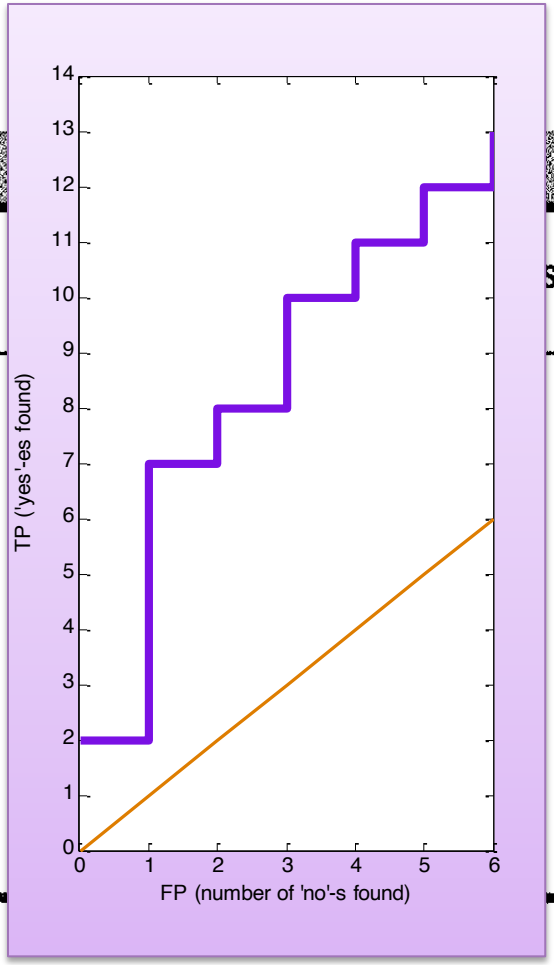


Create a curve from these...

Of a 'yes'

Table 5.6 Data for a lift chart.

Rank	Predicted probability	Actual class	Rank	ss
1	0.95	*yes*	11	
2	0.93	*yes*	12	
3	0.93	*no*	13	
4	0.88	*yes*	14	
5	0.86	*yes*	15	
6	0.85	*yes*	16	
7	0.82	*yes*	17	
8	0.80	*yes*	18	
9	0.80	*no*	19	
10	0.79	*yes*	. . .	



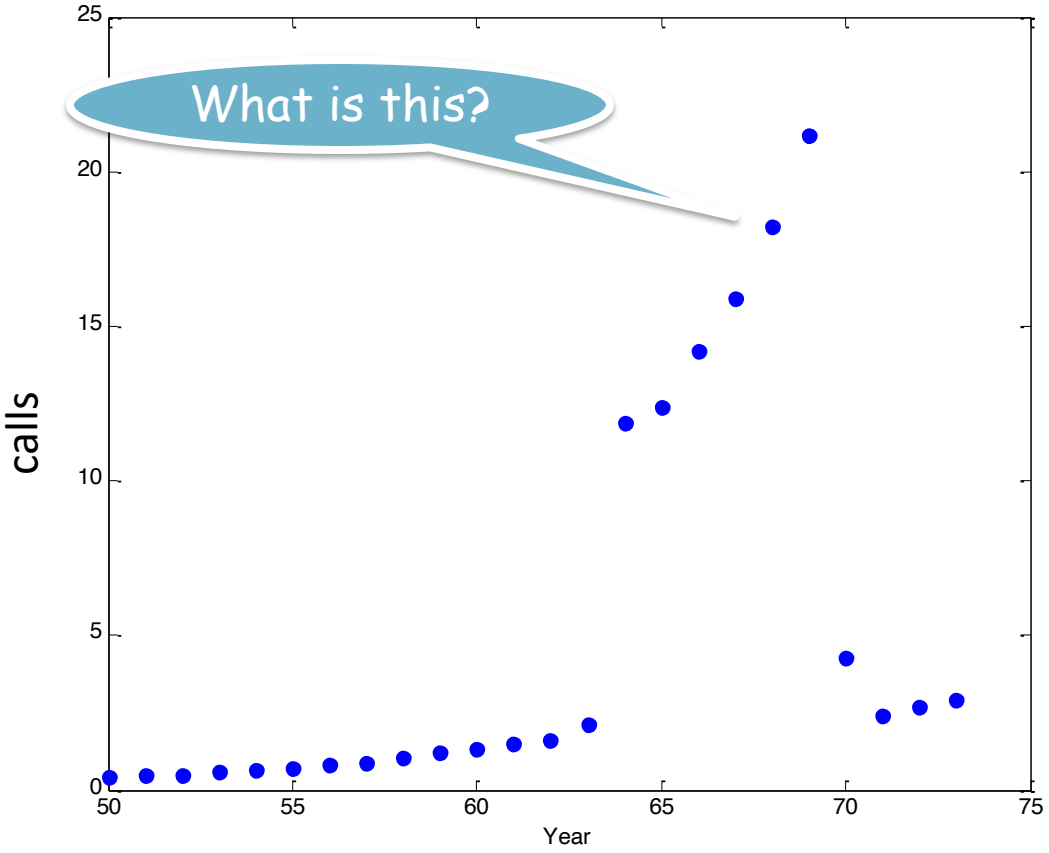
Numeric Prediction

Table 5.8 Performance measures for numeric prediction*

Performance measure	Formula
mean-squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}}$
mean absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{n}$
relative squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}$, where $\bar{a}=\frac{1}{n}\sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}}$
relative absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{\lvert a_1-\bar{a}\rvert+\ldots+\lvert a_n-\bar{a}\rvert}$
correlation coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}}$, where $S_{PA}=\frac{\sum_i (p_i-\bar{p})(a_i-\bar{a})}{n-1}$, $S_p=\frac{\sum_i (p_i-\bar{p})^2}{n-1}$, and $S_A=\frac{\sum_i (a_i-\bar{a})^2}{n-1}$

* p are predicted values and a are actual values.

Belgian International Phone calls (1950-1973)

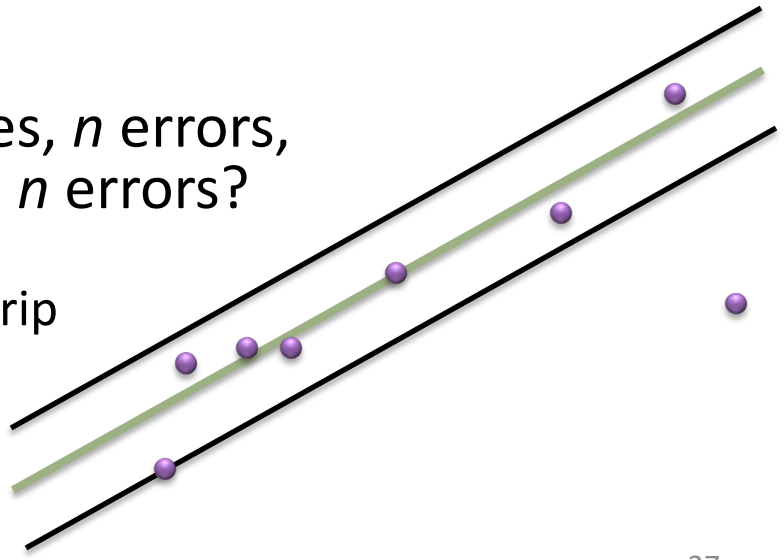


Robust regression – options

1. Minimize Mean Absolute Error instead of Mean Squared Error

2. Detect and Ignore Outliers:
   - build a model, remove 1% most deviating cases (outliers)
   - build another model, remove 1% most deviating cases
   - etc., till, say 50% of cases left

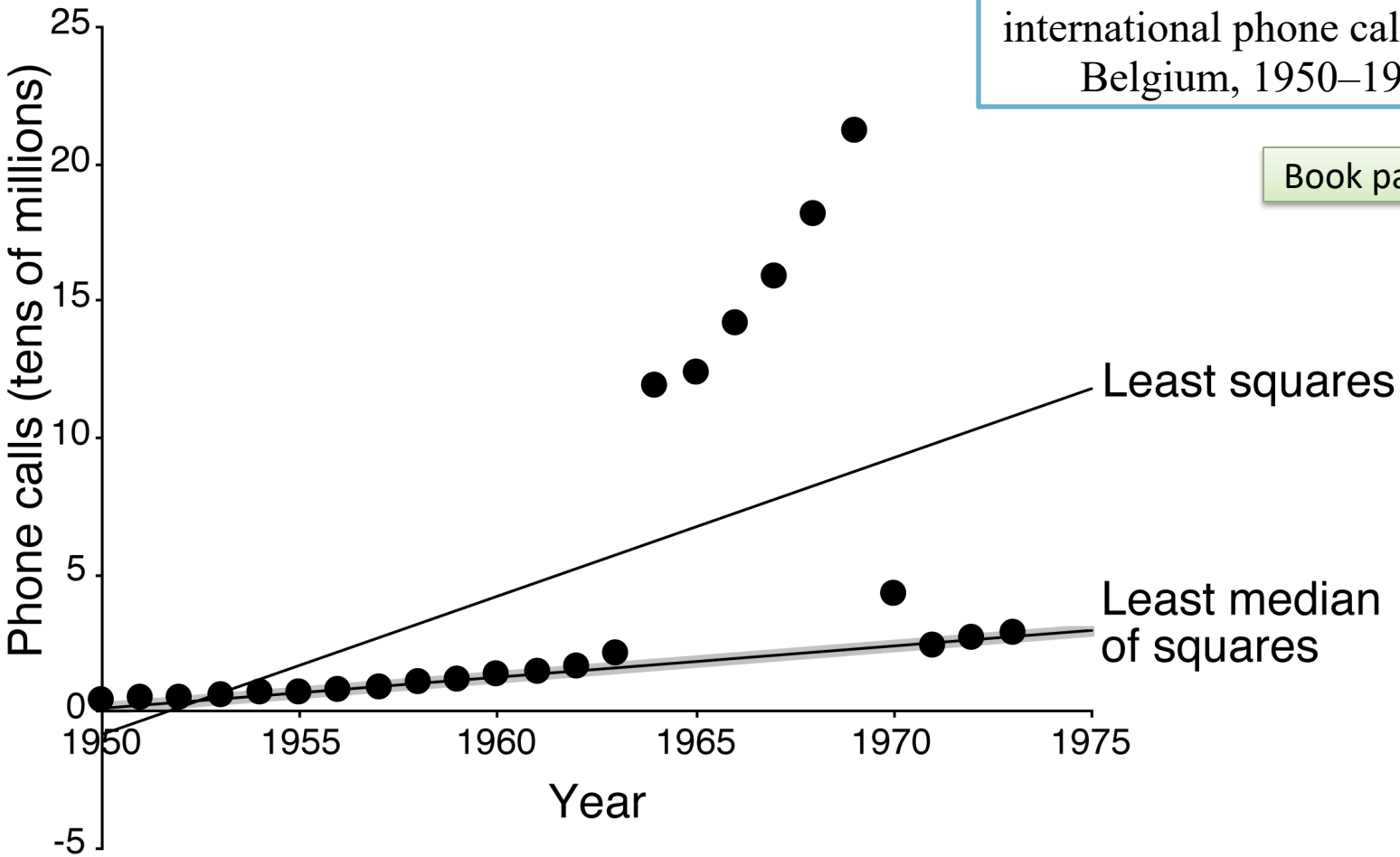
3. Minimize the median error: *n* cases, *n* errors, what is the median value of these *n* errors?
   - geometrical interpretation:
     the line in the narrowest strip that covers 50% of points
   - works very well



What happened in Belgium?

Is it really? – Number of international phone calls from Belgium, 1950–1973

Book page 313



You can follow this link to learn more

Or download the data: http://www.uni-koeln.de/themen/statistik/data/rousseeuw/

Mean vs. Median

Given *n* numbers: $x_1, x_2, \ldots, x_n$

Question 1:

which value y minimizes the Mean Squared Error

$$MSE(y) = ((x_1-y)^2+(x_2-y)^2+ \ldots + (x_k-y)^2)/n$$

Not good with skewed data

Answer:

the mean: $y = (x_1+ x_2 + \ldots + x_n)/n$

Question 2:

which value y minimizes the Mean Absolute Error

$$MAE(y) = (|x_1-y| + |x_2-y| + \ldots + |x_k-y|)/n$$

Answer:

the median: any y that lies "in the middle of $x_1, x_2, \ldots, x_n$"

Curves and numbers

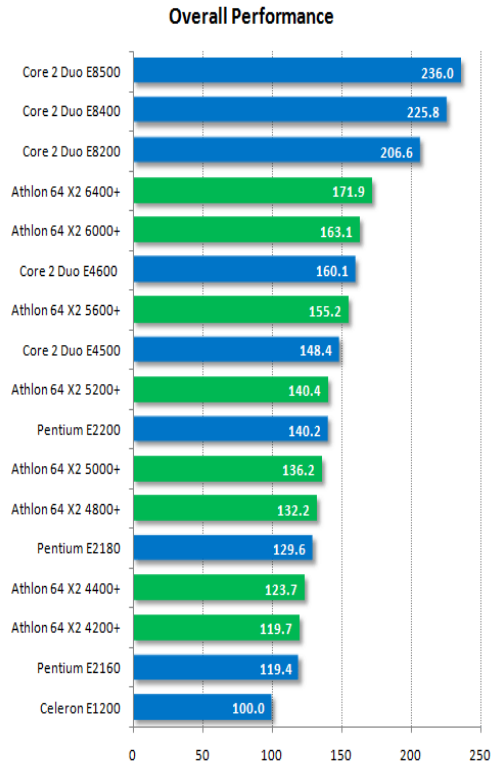
- You have seen a lot now
  - Performance for classification
  - Performance for numerical prediction
- Many others exist of course….
  - Select one that fits your problem best

Remember? A regression problem

- CPU performance

- Computer's CPU performance (PRP) depends on a number of factors:
  - cycle time (MYCT)
  - main memory (MMIN, MMAX)
  - cache (CACH)
  - number of channels (CHMIN, CHMAX)

- Problem:
  - express PRP as a function of all these factors



Linear regression idea

n – number of attributes used
k – number of training examples

- Assume a linear function:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

- Find values for $w_0, \ldots, w_n$ for which the squared error

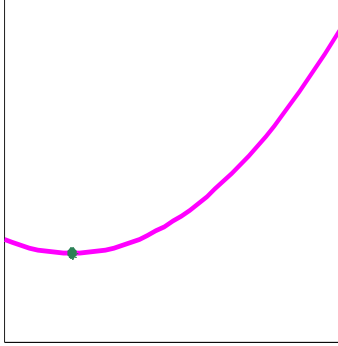
$$\text{error} = (a_1 - y_1)^2 + (a_2 - y_2)^2 + \ldots + (a_k - y_k)^2$$

is minimized

$a_1, \ldots, a_k$ are actual values of the target variable,

$y_1, \ldots, y_k$ are predictions

Finding the weights



- Error function: quadratic, with $n+1$ variables:

$$\mathbf{error(w) = (a_1 - y_1(w))^2 + (a_2 - y_2(w))^2 + \ldots + (a_k - y_k(w))^2}$$

- All partial derivatives = 0 → error is minimal

- Partial derivatives are linear functions of ***w***
  - **n+1 partial derivatives**
  - **n+1 variables for w**

- Finding optimal w reduces to the problem of solving a system of $(n+1)$ linear equations with $(n+1)$ variables

The CPU dataset

n = 6
k = 209

MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
50	2000	16000	24	6	16	65
1500	768	1000	0	0	0	12
70	4000	12000	8	6	8	75
52	4000	16000	32	4	12	130
90	256	1000	0	3	10	17
143	2300	6200	0	6	64	61
1100	768	2000	0	1	1	13
125	2000	8000	0	2	14	52
23	16000	64000	64	16	32	636
50	4000	32000	112	52	104	397
…	…	…	…	…	…	…

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

$$error(w) = (a_1 - y_1(w))^2 + (a_2 - y_2(w))^2 + \ldots + (a_{209} - y_{209}(w))^2$$

Result

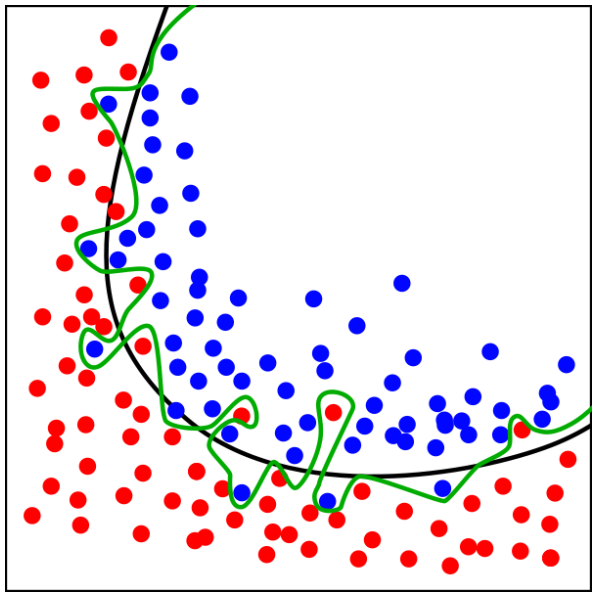
$$
\begin{aligned}
PRP = \\
&- 56.1 \\
&+ 0.049\ MYCT \\
&+ 0.015\ MMIN \\
&+ 0.006\ MMAX \\
&+ 0.630\ CACH \\
&- 0.270\ CHMIN \\
&+ 1.46\ CHMAX
\end{aligned}
$$

Questions

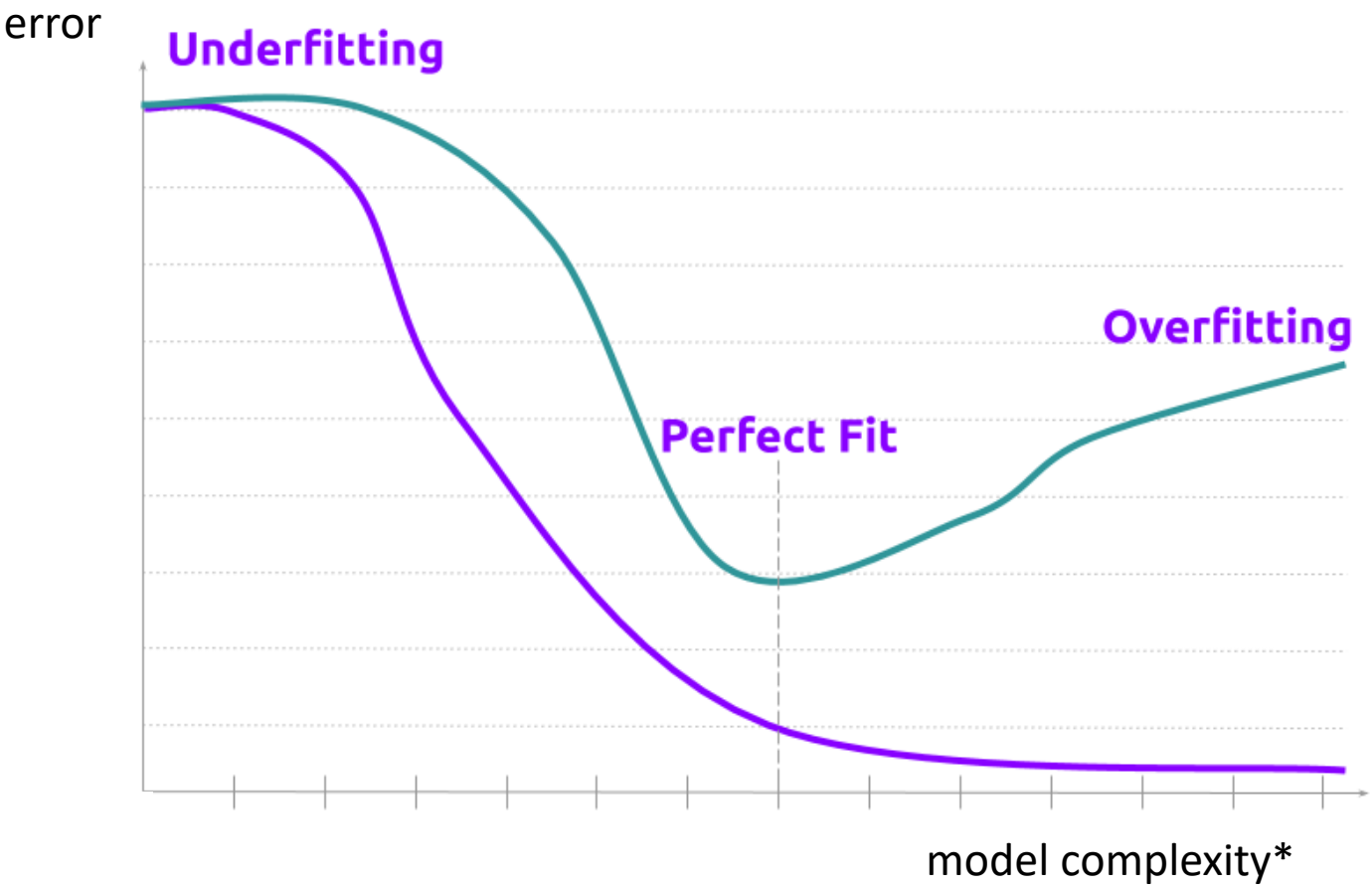
A. How to avoid overfitting?
B. Are there other ways to measure the error?
C. Is a single linear regression model enough?
D. Is a linear function not too simple?

A. Overfitting

- Overfitting is when...
  - You choose to memorise and not to understand
  - When you learn too much about the training data but are unable to generalise the idea
  - When your results on the training set are good, but on the test set, poor.

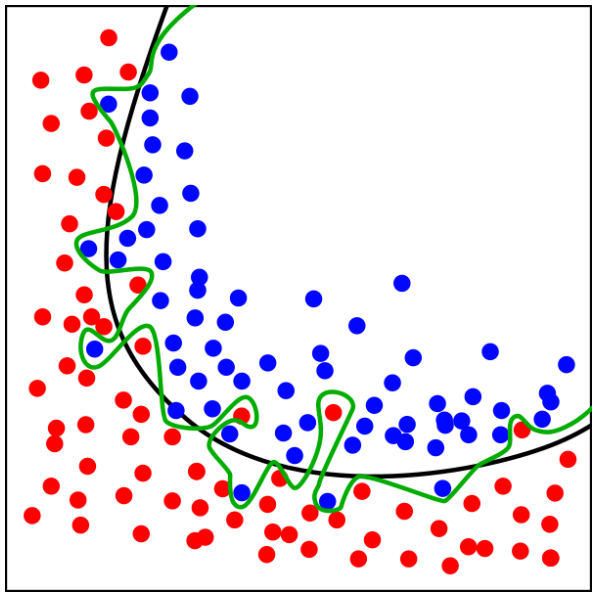


Overfitting: train vs test performance



A. Overfitting

- Overfitting is when...
  - You choose to memorise and not to understand
  - When you learn too much about the training data but are unable to generalise the idea
  - When your results on the training set are good, but on the test set, poor.



- One solution (of many)
  - Use less attributes
  - All kinds of norms can help (L1/L2)

Removing attributes *(variables)*

- **Backward selection heuristic**
  - remove variable after variable using the following ***estimate*** of the test error:

$$error_{test} = error_{train} \cdot (n+v)/(n-v),$$

where n=number of cases; v=number of variables

- Example:

$error_{test8} = error_{train8} \cdot (50+8)/(50-8) = error_{train8} \cdot 1.38$

$error_{test7} = error_{train7} \cdot (50+7)/(50-7) = error_{train7} \cdot 1.32$

Today we had...

- Sampling
- Bootstraps
- Confidence
- Significance tests
- Some brief start of regression