CPSC 340: Machine Learning and Data Mining

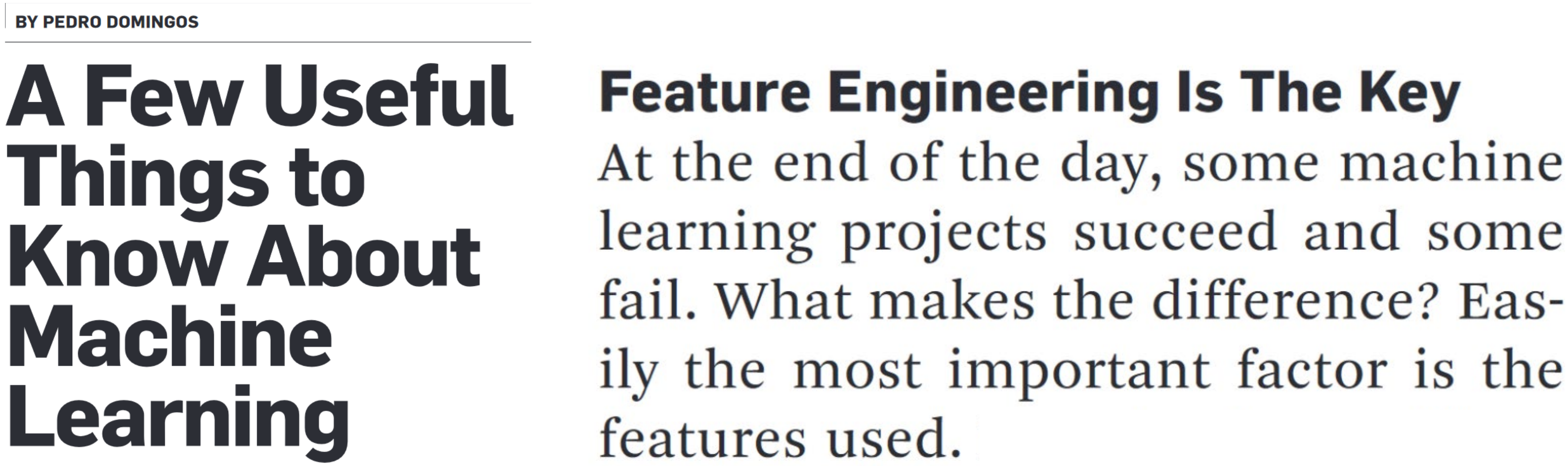
Feature Engineering

Summer 2021

In This Lecture

1. Feature Engineering Intro (10 minutes)
2. Review of Feature Transforms (10 minutes)
3. Representing Text Data (20 minutes)
4. Global and Local Features (10 minutes)

Why Feature Engineering?



BY PEDRO DOMINGOS

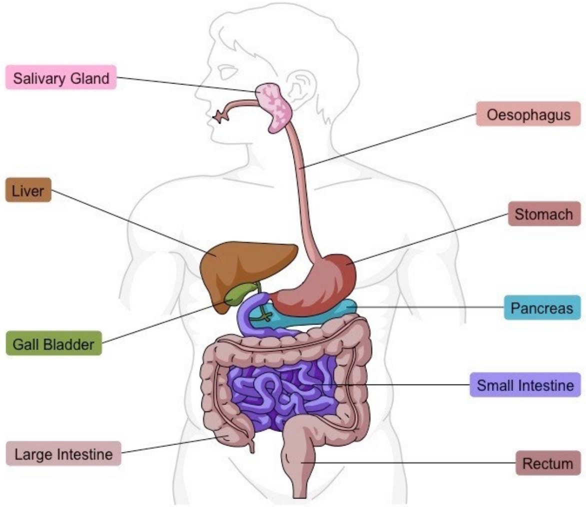
A Few Useful Things to Know About Machine Learning

Feature Engineering Is The Key

At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.

- “Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.”
  - Andrew Ng

Recall: “Process”



Digestive process

- Data is generated from a process
- “Process” := mapping of input → output based on State of the world ____________________.
- e.g. state of the world includes “I’m lactose-intolerant”
  => digestive process determines milk > 50ml → “sick”.

“State”

- “State” := sufficient and relevant information for a perfect prediction
  - Assumption: labels are consequences/parts of states
  - e.g. know the “state” of my body
    => perfectly predict acid reflux
  - e.g. know the “state” of the stock market
    => perfectly predict stock price
  - e.g. know the “state” of an email
    => perfectly predict spam-ness

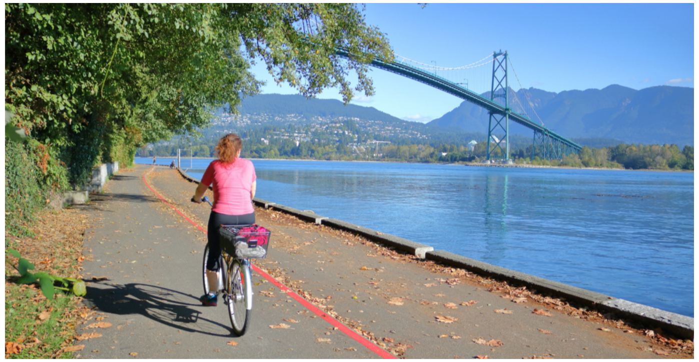
- 2 Fundamental Problems:
  1. observing states is hard
  2. representing. states is hard

“Observation”

- “Observation” := rendered state information

```
{
  “bike_position”: (10, 25, 30),
  “bike_velocity”: (2, 2, 0),
  “bridge_position”: (21, 60, 30),
    ....
}
```

Relevant state of the world
(a dictionary)



Observation of this state
(rendered version of the dictionary)

Q: Can we predict how long it takes to get to the bridge?

Q: Can we predict how long it takes to get to the bridge?

- Usually imperfect (lossy and noisy)
  - e.g. velocity is lost
  - e.g. irrelevant information is introduced (e.g. colour of shirt)
  - Not much we can do about it!

“Features”

- “Features” := quantities representing observations

Observation

shirt_colour	sky_colour	bike_x	bridge_x	...
pink	blue	10	21	

Features

- Good features should ideally reconstruct state information:
  - Expressive: represent complex observations without losing information
  - Relevant: given a task, present useful information

Feature Engineering

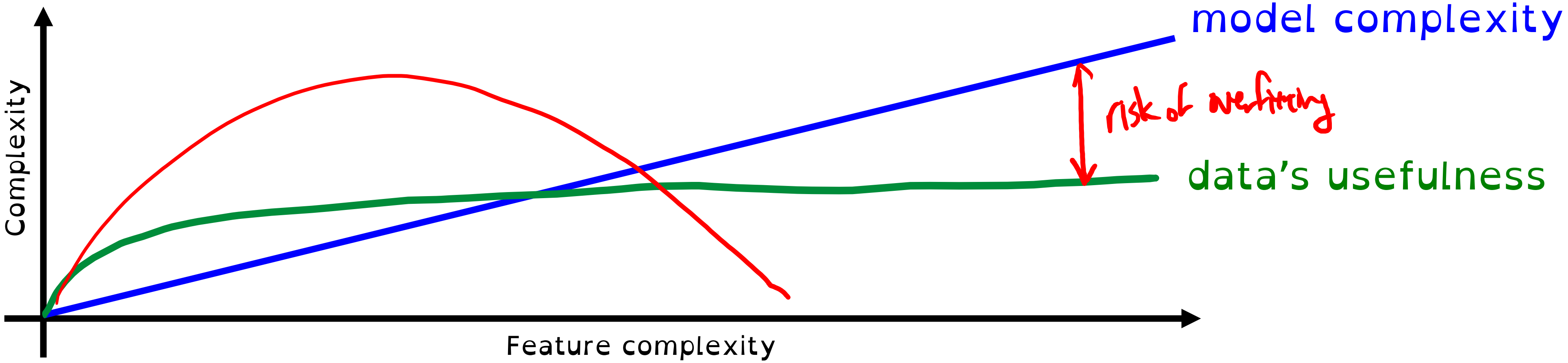
- Exact same observations can be represented with different features
  - Depending on context, some features are “good” and some are “bad”.

- Better features usually help more than a better model.
  - Why is “deep learning” popular? Deep learning learns good features.
  - Allow learning with few examples, be hard to overfit with many examples.

- “Feature engineering”: manually extract good features from observations
  - Context-specific: dependent on the model you use.
  - Capture most important aspects of problem.
  - Reflect invariances (generalize to new scenarios).
  - If given some default features, transform into better features.

Are Complex Features Always Good?

- A complex feature represents a complex signal in observations
  - But observation can add irrelevant information!
  - Model's complexity ∝ feature complexity
    - e.g. KNN picks up complicated patterns in features
  - What if data's usefulness is NOT ∝ feature complexity?
    - Maybe the feature is complex because of irrelevant information



- There is a trade-off between simple and expressive features:
  - With simple features overfitting risk is low, but accuracy might be low.
  - With complicated features accuracy can be high, but so is overfitting risk.

Feature Engineering

- The best features may be dependent on the model you use.

$O(K^d)$ examples needed

- For counting-based methods like naïve Bayes and decision trees:
  - Need to address coupon collecting, but separate relevant “groups”.

- For distance-based methods like KNN:
  - Want different class labels to be “far”.

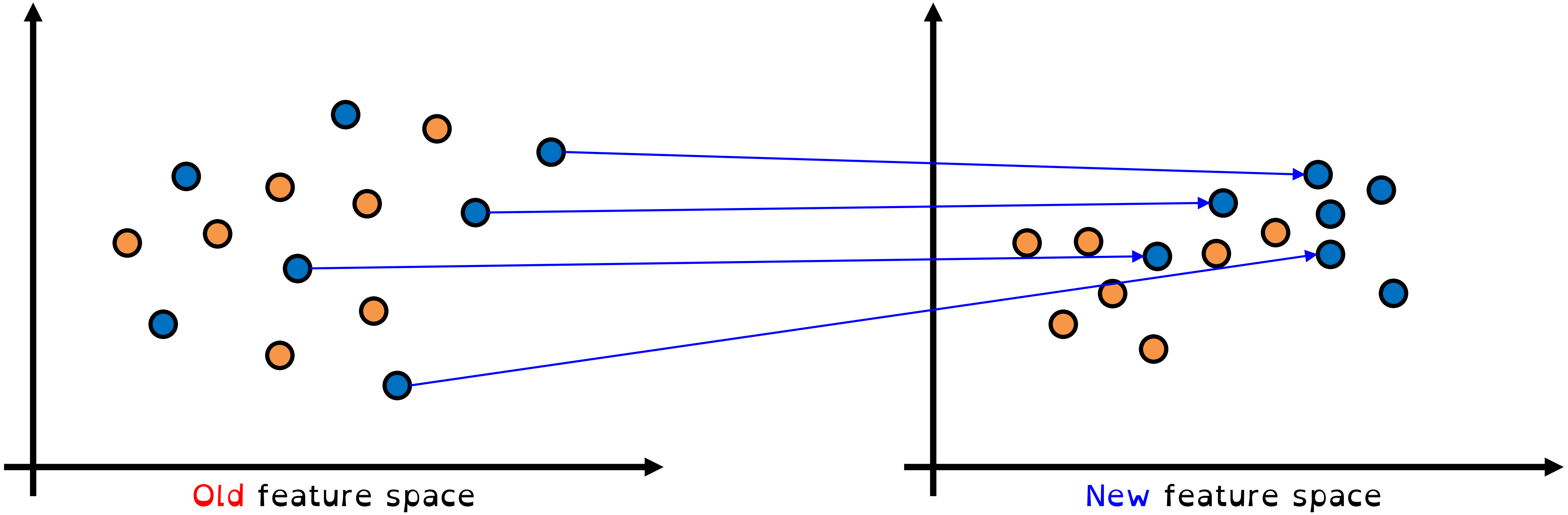
- For regression-based methods like linear regression:
  - Want labels to have a linear dependency on features.

Coming Up Next

FEATURE TRANSFORMS REVIEW

Feature Transform is Part of Training

- Important: when you do feature transform, you are already in training pipeline
  - i.e. you should not let test data inform your feature transform
  - i.e. which method of feature transform you use is a hyper-parameter
- Feature transform: a mapping of old feature space → new feature space
  - (hand-crafted, not learned)



Note: these two feature spaces
may have different dimensionalities!

Recall: Discretization

- For counting-based methods:
  - Discretization: turn continuous into discrete.

Age
23
23
22
25
19
22

< 20	>= 20, < 25	>= 25
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0

  - Counting age “groups” could let us learn more quickly than exact ages.
    - But we wouldn’t do this for a distance-based method.

Recall: Standardization

- Consider features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard ‘unit’?
  - It doesn’t matter for counting-based methods.
- It matters for distance-based methods:
  - KNN will focus on large values more than small values.
  - Often we “standardize” scales of different variables (e.g., Z-score).
  - Also need to worry about correlated features.

Recall: One-Hot Encoding

- How do we use categorical features in regression?
- Standard approach is to convert to a set of binary features:
  - "1 of k" or "one hot" encoding.

Age	City	Income
23	Van	22,000.00
23	Bur	21,000.00
22	Van	0.00
25	Sur	57,000.00
19	Bur	13,500.00
22	Van	20,000.00

→

Age	Van	Bur	Sur	Income
23	1	0	0	22,000.00
23	0	1	0	21,000.00
22	1	0	0	0.00
25	0	0	1	57,000.00
19	0	1	0	13,500.00
22	1	0	0	20,000.00

0 0 0

new city

Q: What if you get
a new city in the test data?

Digression: Linear Models with Binary Features

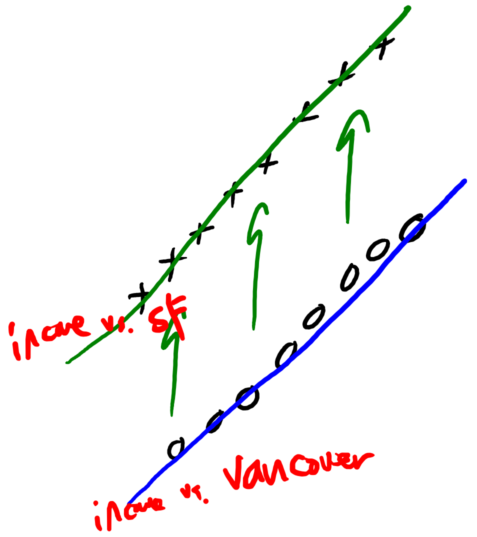
Q: What is the effect of binary features on linear regression?

- Suppose we use a bag of words:
  - With 3 words {"hello", "Vicodin", "340"} our model would be:

$$\overset{\text{spam-score}}{\hat{y}_i} = w_1 \underbrace{\overset{1}{x_{i1}}}_{\text{whether "hello" appears}} + \cancel{w_2}\, \overset{0 \to 0}{x_{i2}} + w_3 \underbrace{\overset{1}{x_{i3}}}_{\text{whether "340" appears}}$$

  - If email only has "hello" and "340" our prediction is:

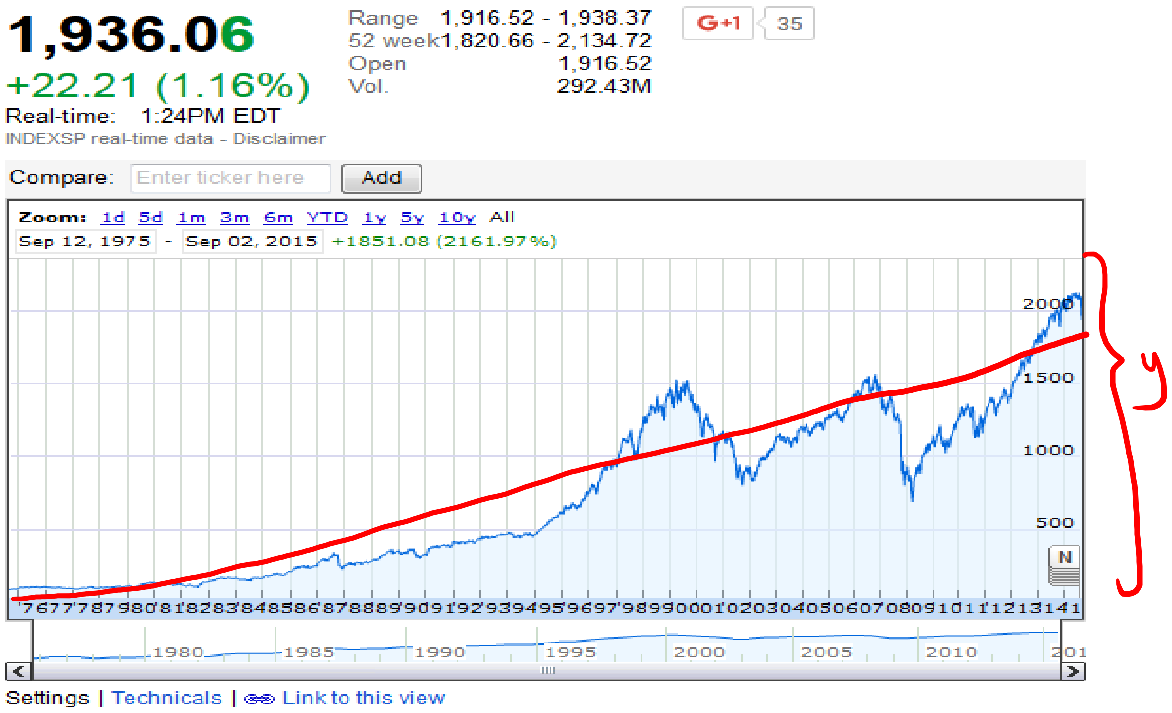
$$\hat{y}_i = \underbrace{w_1}_{\text{"hello" weight}} + \underbrace{w_3}_{\text{"340" weight}}$$



- So having the binary feature 'j' increases $\hat{y}_i$ by the fixed amount w_j.
  - Predictions are a bit like naïve Bayes where we combine features independently.
  - But now we're learning all w_j together so this tends to work better.

Non-Linear Transformations for Features/Labels

- Non-linear feature/label transforms can make things more (linear/non-linear):
  - Polynomial, exponential/logarithm, sines/cosines, RBFs.



1,936.06
+22.21 (1.16%)
Real-time: 1:24PM EDT
INDEXSP real-time data - Disclaimer

Range 1,916.52 - 1,938.37
52 week 1,820.66 - 2,134.72
Open 1,916.52
Vol. 292.43M

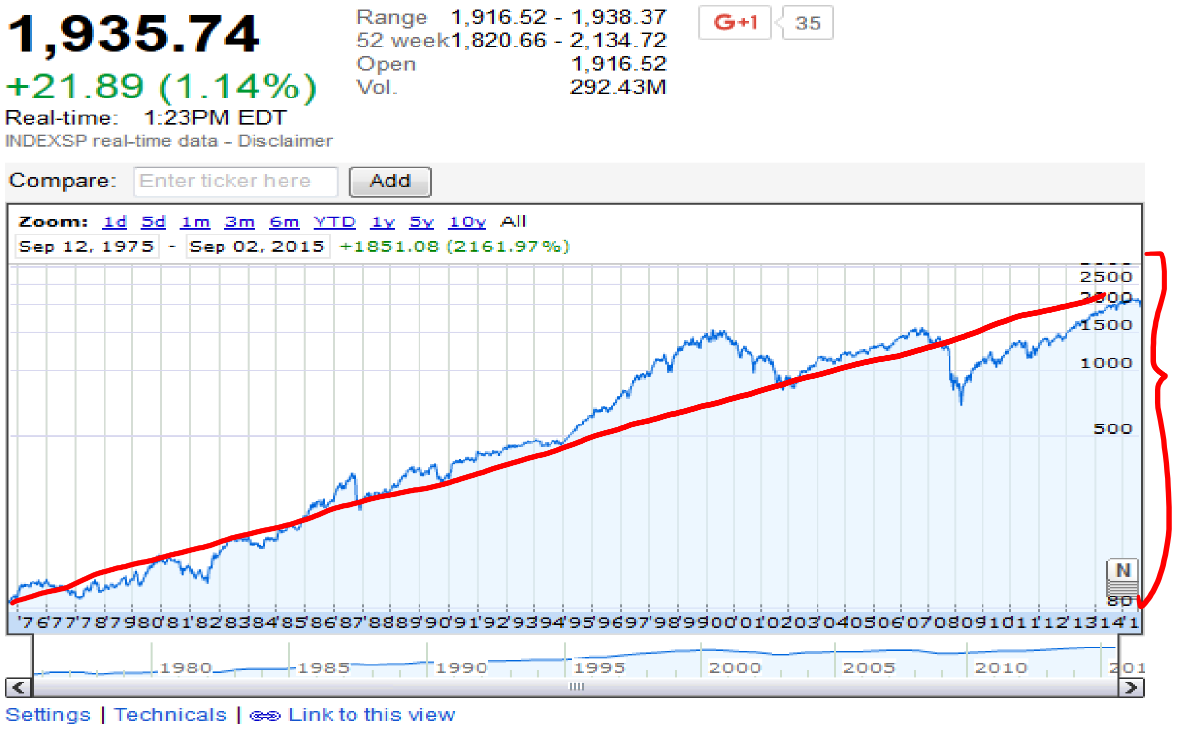
G+1 35

Compare: Enter ticker here Add

Zoom: 1d 5d 1m 3m 6m YTD 1y 5y 10y All
Sep 12, 1975 - Sep 02, 2015 +1851.08 (2161.97%)

Settings | Technicals | Link to this view

y



1,935.74
+21.89 (1.14%)
Real-time: 1:23PM EDT
INDEXSP real-time data - Disclaimer

Range 1,916.52 - 1,938.37
52 week 1,820.66 - 2,134.72
Open 1,916.52
Vol. 292.43M

G+1 35

Compare: Enter ticker here Add

Zoom: 1d 5d 1m 3m 6m YTD 1y 5y 10y All
Sep 12, 1975 - Sep 02, 2015 +1851.08 (2161.97%)

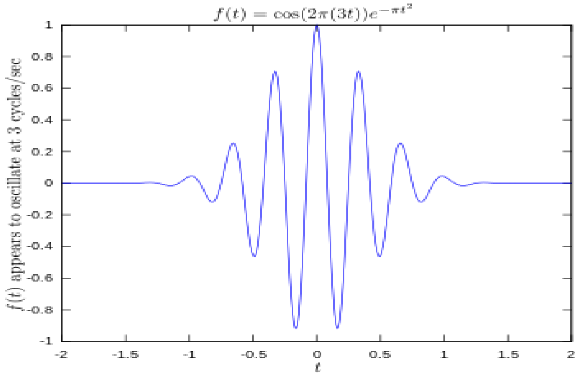
Settings | Technicals | Link to this view

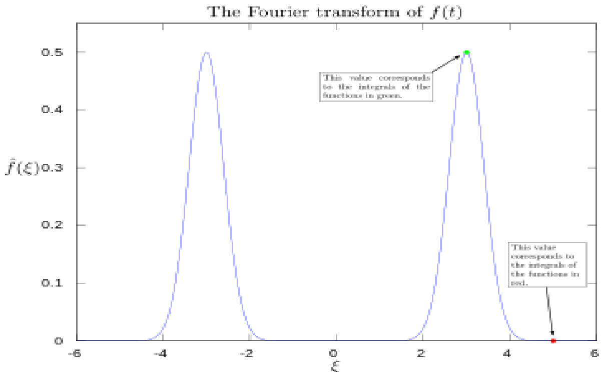
log y

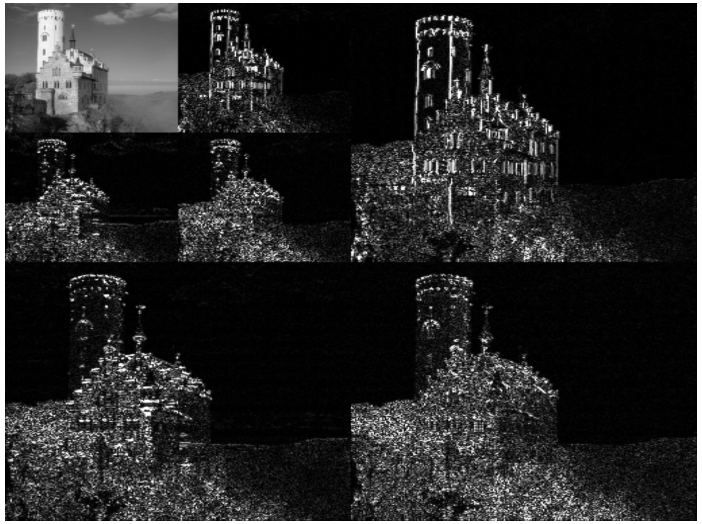
www.google.com/finance

Domain-Specific Transformations

- In some domains there are natural transformations to do:
  - Fourier coefficients and spectrograms (sound data).
  - Wavelets (image data).
  - Convolutions (we'll talk about these later).







https://en.wikipedia.org/wiki/Fourier_transform
https://en.wikipedia.org/wiki/Spectrogram
https://en.wikipedia.org/wiki/Discrete_wavelet_transform

Discussion of Feature Engineering

- The best feature transformations are application-dependent.
  - It’s hard to give general advice.

- Mark’s advice: ask the domain experts.
  - Often have idea of right discretization/standardization/transformation.

- If no domain expert, cross-validation will help.
  - Recall: choice of feature transforms is a hyper-parameter
  - Or if you have lots of data, use deep learning methods from Part 5.

- Next: some features used for text applications.

Coming Up Next

REPRESENTING TEXT DATA

Text Example 1: Language Identification

- Consider data that doesn’t look like this:

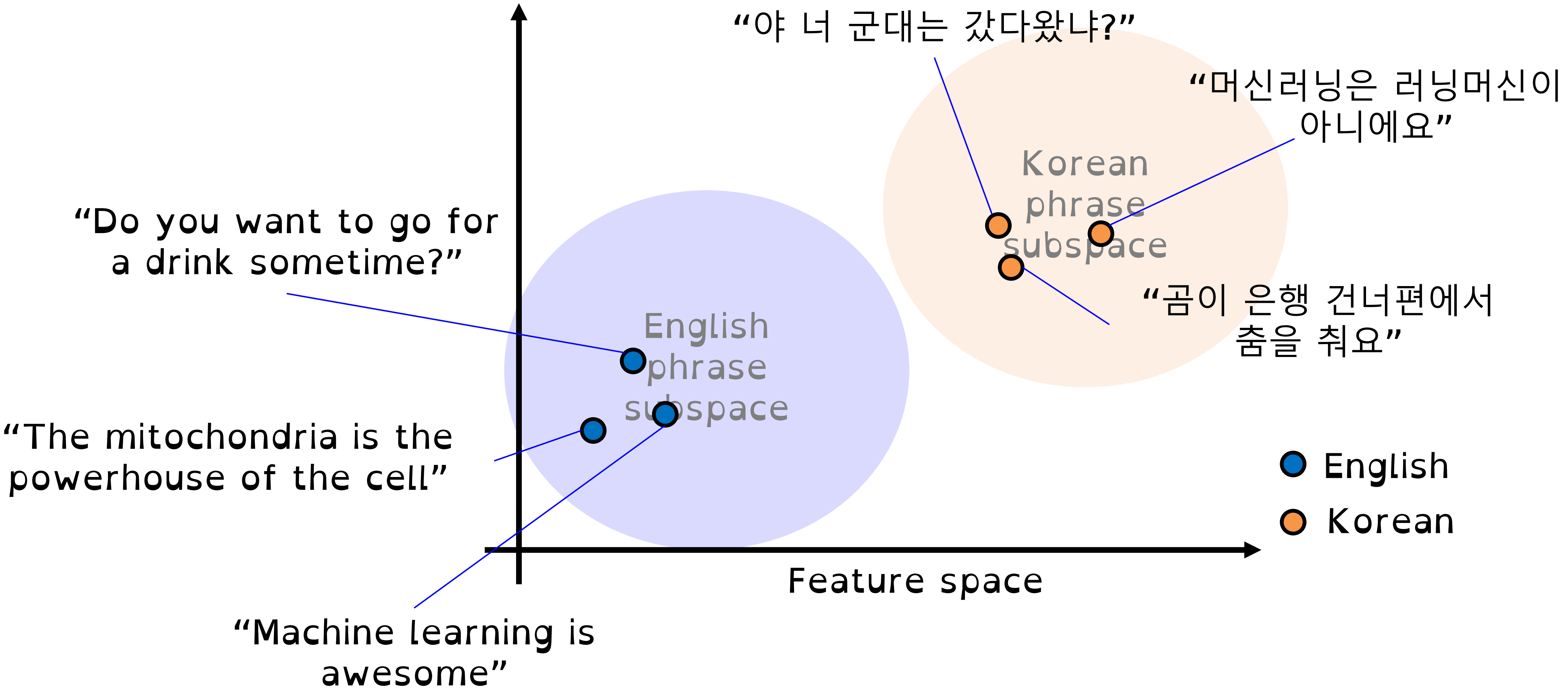
$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

- But instead looks like this:

$$X = \begin{bmatrix} \text{Do you want to go for a drink sometime?} \\ \text{J'achète du pain tous les jours.} \\ \text{Fais ce que tu veux.} \\ \text{There are inner products between sentences?} \end{bmatrix}, y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

Q: How should we represent sentences using features?

Can We Build a Feature Space Like This?



“야 너 군대는 갔다왔냐?”

“머신러닝은 러닝머신이 아니에요”

“Do you want to go for a drink sometime?”

“곰이 은행 건너편에서 춤을 춰요”

“The mitochondria is the powerhouse of the cell”

“Machine learning is awesome”

A (Bad) Universal Representation

- Treat character in position ‘j’ of the sentence as a categorical feature.
  - “fais ce que tu veux” => x_i = [f a i s ‘’ c e ‘’ q u e ‘’ t u ‘’ v e u x .]

    102 97 105

- “Pad” end of the sentence up to maximum #characters:

    0

  - “fais ce que tu veux” => x_i = [f a i s ‘’ c e ‘’ q u e ‘’ t u ‘’ v e u x . γ γ γ γ γ γ γ γ …]

- Advantage:
  - – No information is lost, KNN can eventually solve the problem.
- Disadvantage: throws out everything we know about language.
  - – We don’t have positional invariance:
    - Needs to learn that “veux” starting from any position indicates “French”.
    - Here, sentences are made of characters not words.
  - – High overfitting risk, you will need a lot of examples for this easy task.

Recall: Bag-of-Words

- Bag-of-words: represent sentences/documents by word counts:

> The **International Conference on Machine Learning** (ICML) is the leading international academic conference in machine learning

ICML	International	Conference	Machine	Learning	Leading	Academic
1	2	2	2	2	1	1

- Bag of words loses a ton of information/meaning:
  - E.g. order of words, relative positions
  - But it easily solves language identification problem

Universal Representation vs. Bag of Words

- Why is bag of words better than “array of characters” here?
  - It needs less data because it captures invariances for the task:
    - Most features give strong indication of one language or the other.
    - It doesn’t matter *where* the French words appear.
  - It overfits less because it throws away irrelevant information.
    - Exact sequence of words isn’t particularly relevant here.

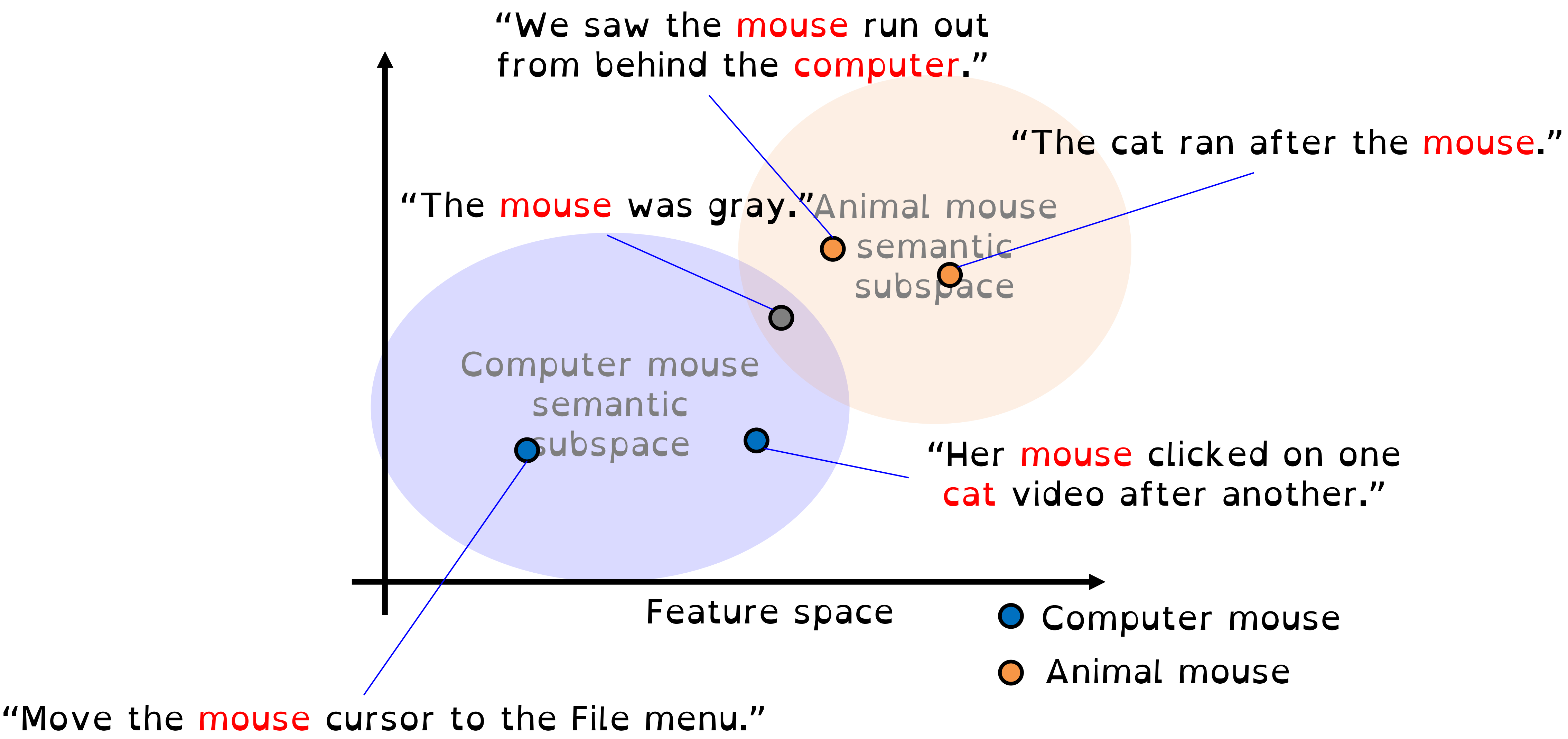
Text Example 2: Word Sense Disambiguation

- Consider the following two sentences:
  - "The cat ran after the mouse."
  - "Move the mouse cursor to the File menu."

- Word sense disambiguation (WSD): classify "meaning" of a word:
  - A surprisingly difficult task.

- You can do ok with bag of words, but it will have problems:
  - "Her mouse clicked on one cat video after another."
  - "We saw the mouse run out from behind the computer."
  - "The mouse was gray." (ambiguous without more context)

Can We Build a Feature Space Like This?



“We saw the mouse run out from behind the computer.”

“The cat ran after the mouse.”

“The mouse was gray.”

“Her mouse clicked on one cat video after another.”

“Move the mouse cursor to the File menu.”

Animal mouse semantic subspace
Computer mouse semantic subspace
Feature space
Computer mouse
Animal mouse

Bigrams and Trigrams

- A bigram is an ordered set of two words:
  - Like “computer mouse” or “mouse ran”.
- A trigram is an ordered set of three words:
  - Like “cat and mouse” or “clicked mouse on”.
- These give more context/meaning than bag of words:
  - Includes neighbouring words as well as order of words.
  - Trigrams are widely-used for various language tasks.
- General case is called n-gram.
  - Unfortunately, coupon collecting becomes a problem with larger ‘n’.

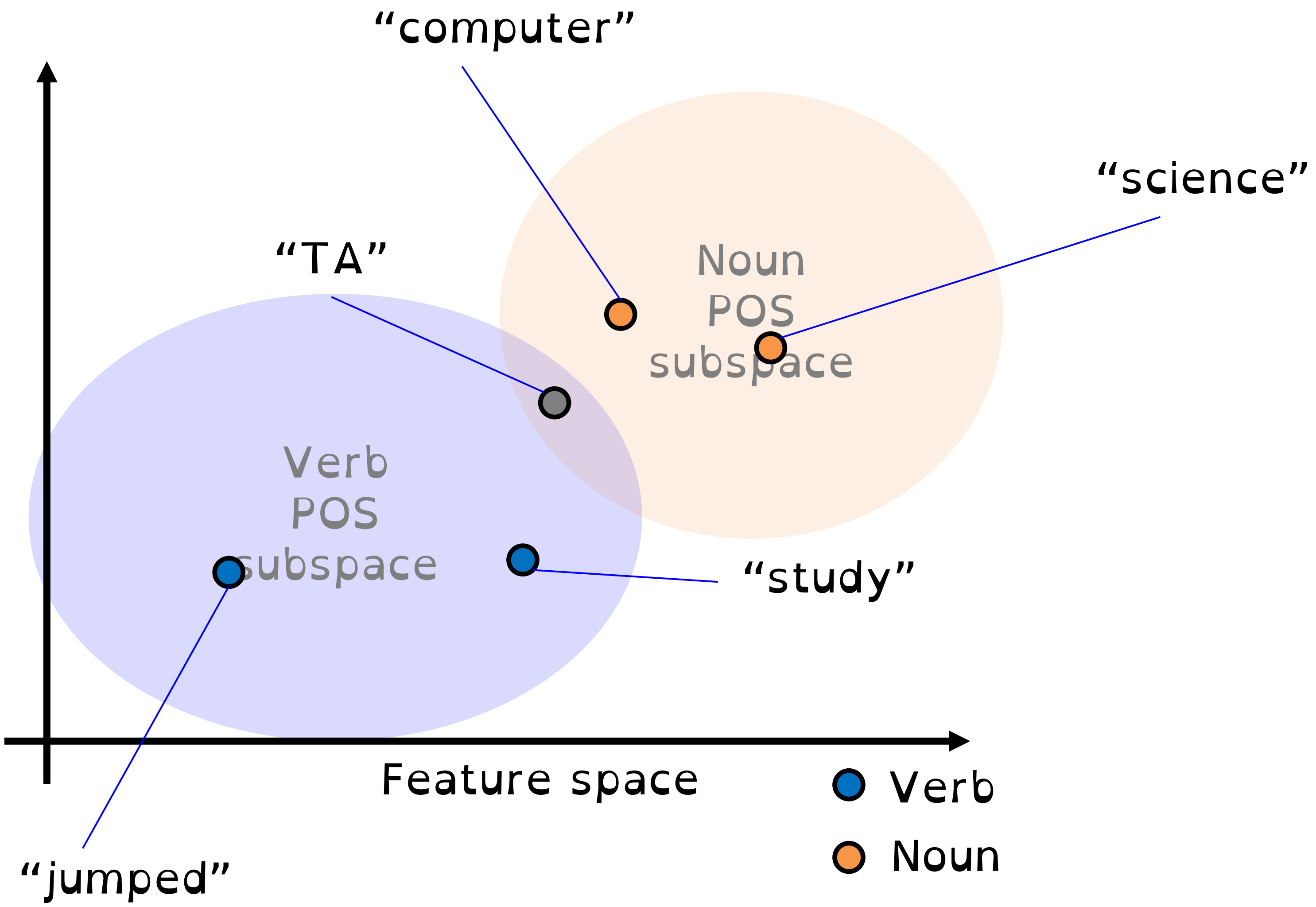
Text Example 3: Part of Speech (POS) Tagging

- Consider problem of finding the verb in a sentence:
  - “The 340 students jumped at the chance to hear about POS features.”

- Part of speech (POS) tagging is the problem of labeling all words.
  - >40 common syntactic POS tags.
  - Current systems have ~97% accuracy on standard (“clean”) test sets.
  - You can achieve this by applying a “word-level” classifier to each word.
    - That independently classifies each word with one of the 40 tags.

- What features of a word should we use for POS tagging?

Can We Build a Feature Space Like This?



POS Features

- Regularized multi-class logistic regression with these features gives ~97% accuracy:
  - Categorical features whose domain is all words (“lexical” features):
    - The word (e.g., “jumped” is usually a verb).
    - The previous word (e.g., “he” hit vs. “a” hit).
    - The previous previous word.
    - The next word.
    - The next next word.
  - Categorical features whose domain is combinations of letters (“stem” features):
    - Prefix of length 1 (“what letter does the word start with?”)
    - Prefix of length 2.
    - Prefix of length 3.
    - Prefix of length 4 (“does it start with JUMP?”)
    - Suffix of length 1.
    - Suffix of length 2.
    - Suffix of length 3 (“does it end in ING?”)
    - Suffix of length 4.
  - Binary features (“shape” features):
    - Does word contain a number?
    - Does word contain a capital?
    - Does word contain a hyphen?

Ordinal Features

- Categorical features with an ordering are called ordinal features.

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

→

Rating
2
5
4
4
1
4
3

- If using decision trees, makes sense to replace with numbers.
  - Captures ordering between the ratings.
  - A rule like (rating ≥ 3) means (rating ≥ Good), which make sense.

Ordinal Features

- With linear models, “convert to number” assumes ratings are equally spaced.
  - “Bad” and “Medium” distance is similar to “Good” and “Very Good” distance.
- One alternative that preserves ordering with binary features:

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

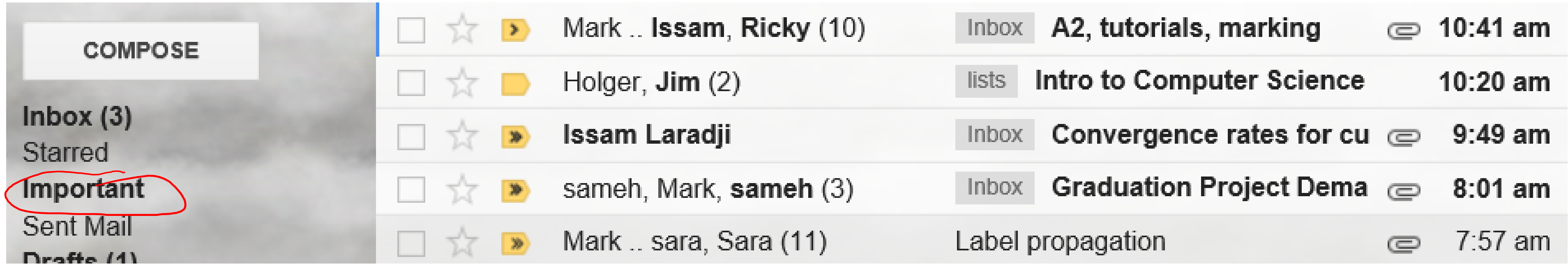
≥ Bad	≥ Medium	≥ Good	Very Good
1	0	0	0
1	1	1	1
1	1	1	0
1	1	1	0
0	0	0	0
1	1	1	0
1	1	0	0

- Regression weight w_{medium} represents:
  - “How much medium changes prediction over bad”.

Coming Up Next

GLOBAL AND LOCAL FEATURES

Motivation: “Personalized” Important Emails



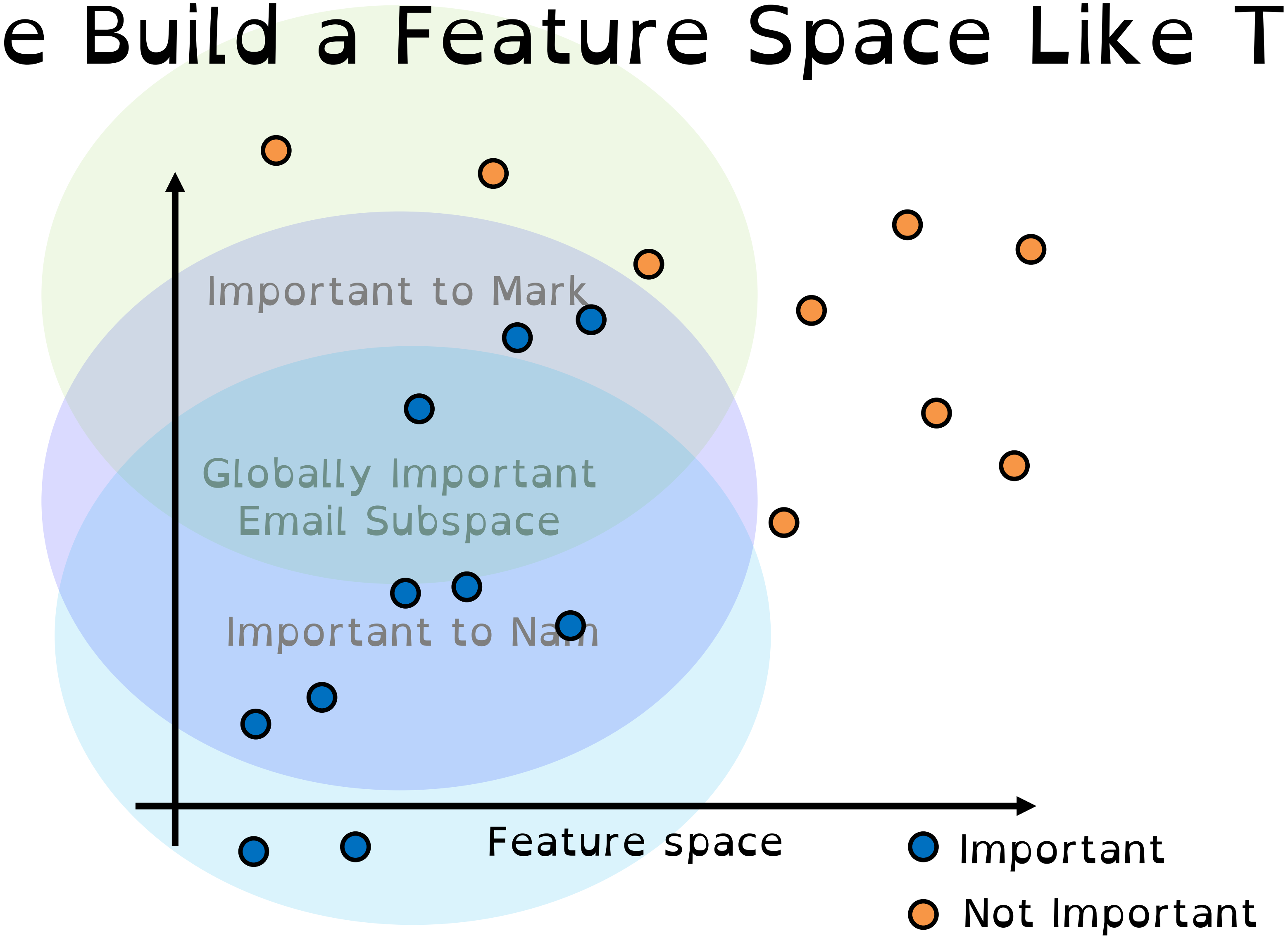
Mark .. Issam, Ricky (10)	Inbox	A2, tutorials, marking	📎	10:41 am
Holger, Jim (2)	lists	Intro to Computer Science		10:20 am
Issam Laradji	Inbox	Convergence rates for cu	📎	9:49 am
sameh, Mark, sameh (3)	Inbox	Graduation Project Dema	📎	8:01 am
Mark .. sara, Sara (11)		Label propagation	📎	7:57 am

- Features: bad of words, trigrams, regular expressions, and so on.

- There might be some “globally” important messages:
  - “This is your mother, something terrible happened, give me a call ASAP.”

- But your “important” message may be unimportant to others.
  - Similar for spam: “spam” for one user could be “not spam” for another.

Can We Build a Feature Space Like This?



“Global” and “Local” Features

- Consider the following weird feature transformation:

“340”
1
1
1
0
1

⇒

“340” (any user)	“340” (user?)
1	User 1
1	User 1
1	User 2
0	<no “340”>
1	User 3

did “340” appear in this e-mail?

if “340” appeared in this e-mail, who was it addressed to?

- First feature will increase/decrease importance of “340” for every user (including new users).
- Second (categorical feature) increases/decreases important of “340” for specific users.
  - Lets us learn more about specific users where we have a lot of data

“Global” and “Local” Features

- Recall we usually represent categorical features using “1 of k” binaries:

“340”
1
1
1
0
1

⇒

“340” (any user)	“340” (user = Nam)	“340” (user = Mark)
1	1	0
1	1	0
1	0	1
0	0	0
1	0	0

Contribute to “globally important” score

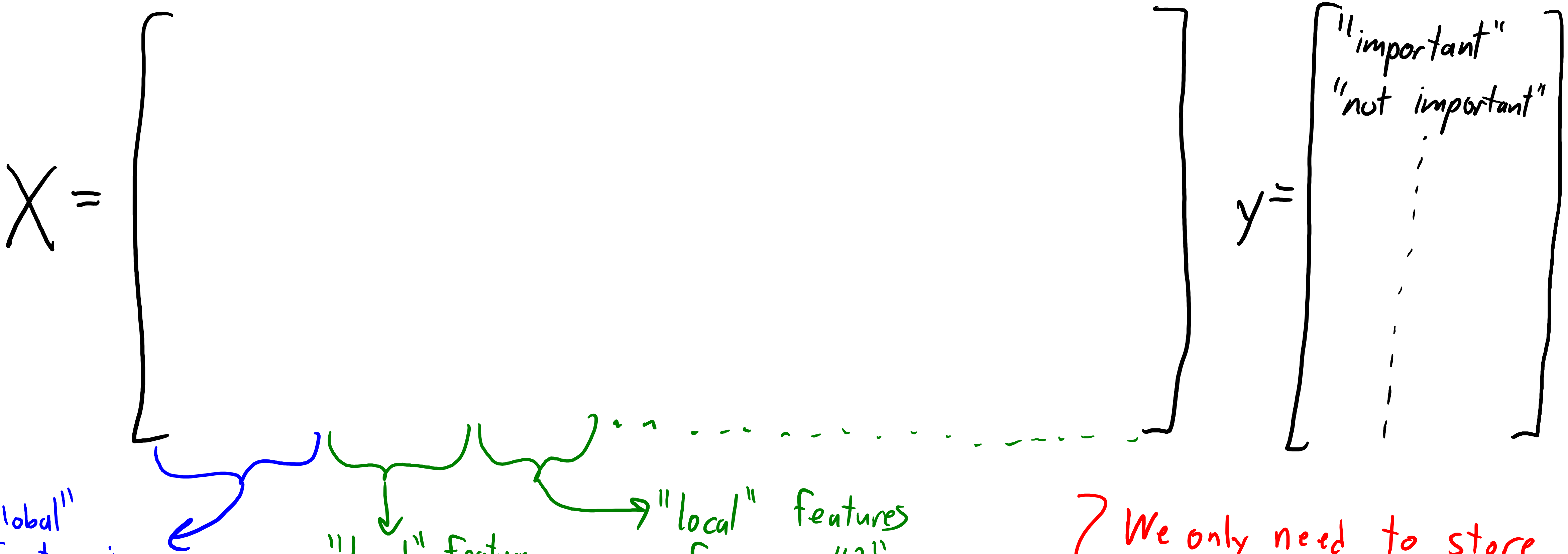
Contribute to “important to Nam” score

Contribute to “important to Mark” score

- First feature “moves the line up” for all users.
- Second feature “moves the line up” when the email is to user 1.
- Third feature “moves the line up” when the email is to user 2.

The Big Global/Local Feature Table for E-mails

- Each row is one email (there are lots of rows):



"global" features: shared by all users

"local" features for user "1": set to 0 for all other users.

"local" features for user "2"

We only need to store the user ID and list of non-zero features.

Predicting Importance of Email For New User

- Consider a new user:
  - We start out with no information about them.
  - So we use global features to predict what is important to a generic user.

$$\hat{y}_i = \text{sign}(\underbrace{w_g^T x_{ig}}_{\text{features/weights shared across users.}})$$

  - Local features are initialized to zero.
- With more data, update global features and user's local features:
  - Local features make prediction *personalized*.

$$\hat{y}_i = \text{sign}(w_g^T x_{ig} + \underbrace{w_u^T x_{iu}}_{\text{features/weights specific to user.}})$$

  - What is important to *this* user?
- Gmail system: classification with logistic regression.
  - Trained with a variant of stochastic gradient (later).

Summary

- Feature engineering can be a key factor affecting performance.
  - Good features depend on the task and the model.
- ~~Bag of words: not a good representation in general.~~
  - ~~But good features if word order isn't needed to solve problem.~~
- ~~Text features (beyond bag of words): trigrams, lexical, stem, shape.~~
  - ~~Try to capture important invariances in text data.~~
- ~~Global vs. local features allow "personalized" predictions.~~

- Next time: feature engineering for text/image/sound data.

Review Questions

- Q1: Why can’t we do anything about observation being a lossy and noisy operation?

- Q2: What is an example where feature complexity does not increase data’s usefulness?

- Q3: What does it mean that bag-of-words can encourage invariance?

- Q4: If we use linear models, how does the number of local features affect the two parts of the fundamental trade-off?

Feature Selection Hierarchy

- Consider a linear models with higher-order terms,

$$\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + w_{12} x_{i1} x_{i2} + w_{13} x_{i1} x_{i3} + w_{23} x_{i2} x_{i3} + w_{123} x_{i1} x_{i2} x_{i3}$$

- The number of higher-order terms may be too large.
  - Can't even compute them all.
  - We need to somehow decide which terms we'll even consider.

- Consider the following hierarchical constraint:
  - You only allow $w_{12} \neq 0$ if $w_1 \neq 0$ and $w_2 \neq 0$.
  - "Only consider feature interaction if you are using both features already."

Hierarchical Forward Selection

- Hierarchical Forward Selection:
  - Usual forward selection, but consider interaction terms obeying hierarchy.
  - Only consider $w_{12} \neq 0$ once $w_1 \neq 0$ and $w_2 \neq 0$.
  - Only allow $w_{123} \neq 0$ once $w_{12} \neq 0$ and $w_{13} \neq 0$ and $w_{23} \neq 0$.
  - Only allow $w_{1234} \neq 0$ once all threeway interactions are present.

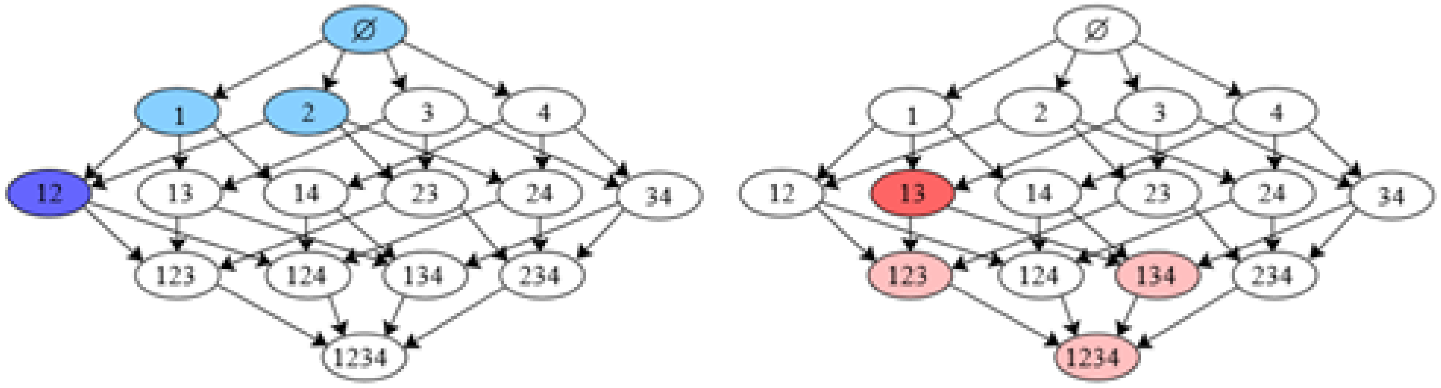


Fig 9: Power set of the set $\{1, \ldots, 4\}$: in blue, an authorized set of selected subsets. In red, an example of a group used within the norm (a subset and all of its descendants in the DAG).

http://arxiv.org/pdf/1109.2397v2.pdf