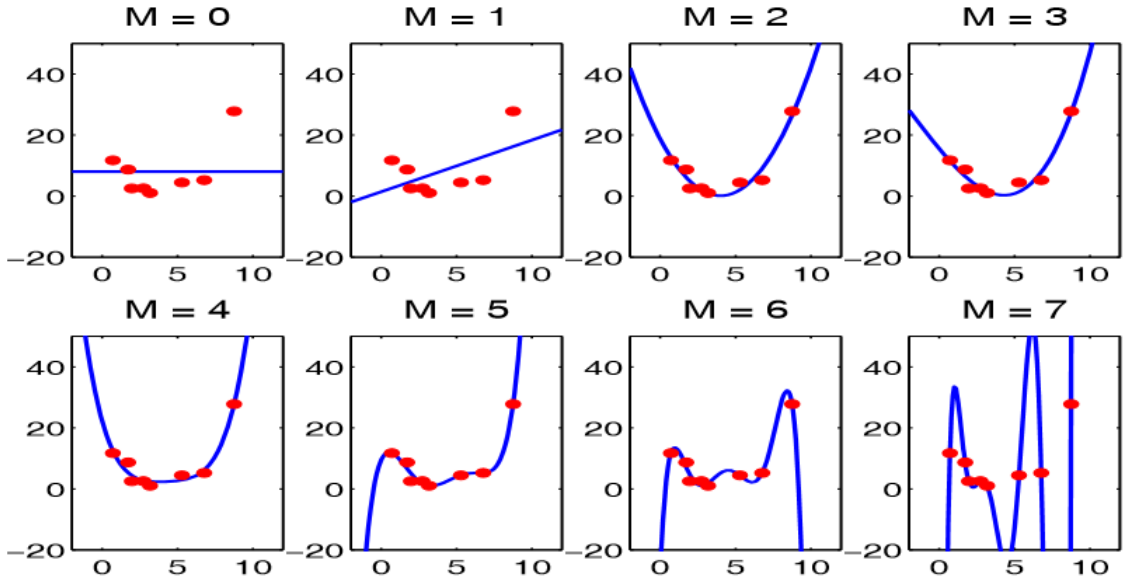
CPSC 340:
Machine Learning and Data Mining

Feature Selection

Fall 2022

Last Time: Finding the "True" Model

- What if y_i really is a polynomial function of x_i?
  - How can we find the "true" degree 'p' of the polynomial?



- Training error does not work:
  - It goes down as 'p' goes up.
- Cross-validation may also not work:
  - Tends to overestimate 'p'.
  - Due to optimization bias.

For example, imagine that the true model is $y_i = 2x_i^2 - 5 + (\text{noise})$

We might choose $d=3$ and a model like $\hat{y}_i = 0.001x_i^3 + 2x_i^2 - 5$,

since it might get a slightly smaller validation error.

Last Time: Complexity Penalties

- We discussed putting a penalty on the model complexity.
  - Want to **fit the data and have a simple model**.

Find 'v' and 'p' minimizing:

$$\text{score}(p) = \underbrace{\frac{1}{2}\|Z_p v - y\|^2}_{\text{usual error}} + \underbrace{\lambda}_{\text{"strength" of penalty}} \underbrace{K}_{\text{number of "degrees of freedom" (K=p+1 for degree-p polynomial)}}$$

  - "To increase the degrees of freedom by one, need to decrease error by λ".
- Prefers smaller degrees of freedom, if errors are similar.
  - Can't optimize this using gradient descent, since it's discontinuous in 'p'.
    - Need to search over values of 'p'.

Bayesian Information Criterion (BIC)

- A disadvantage of these methods:
  - Still prefers a larger 'p' as 'n' grows.

- Solution: make λ depend on 'n'.
- For example, the Bayesian information criterion (BIC) uses:

$$\lambda = \frac{1}{2}\log(n)$$

- BIC penalizes a bit more than AIC for large 'n'.
  - As 'n' goes to ∞, recovers "true" model ("consistent" for model selection).
- In practice, we usually just try a bunch of different λ values.
  - Picking λ is like picking 'k' in k-means.

Discussion of other Scores for Model Selection

- There are many other scores:
  - Elbow method (corresponds to specific choice of λ).
    - You could also use BIC for choosing 'k' in k-means.
  - Methods based on validation error.
    - "Take smallest 'p' within one standard error of minimum cross-validation error".
  - Minimum description length.
  - Risk inflation criterion.
  - False discovery rate.
  - Marginal likelihood (CPSC 440).

- These can adapted to use the L1-norm and other errors.

Next Topic: Feature Selection

Motivation: Discovering Food Allergies

- Recall the food allergy example:

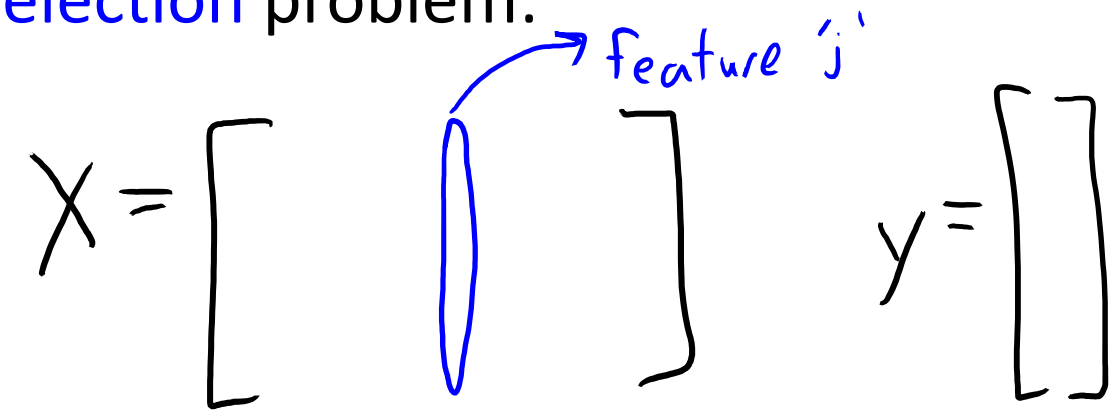
Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…		Sick?
0	0.7	0	0.3	0	0		→	1
0.3	0.7	0	0.6	0	0.01		→	1
0	0	0	0.8	0	0		→	0
0.3	0.7	1.2	0	0.10	0.01		→	1

- What I want to know which foods are making me sick?
  - Rather than building a black box that tells me if I will be sick.

- Instead of prediction, we want to do feature selection:
  - Which foods are "relevant" for predicting "sick".

Feature Selection

- General feature selection problem:



  - Find the features (columns) of 'X' that are important for predicting 'y'.
    - "What are the relevant factors?"
    - "Which basis functions should I use among these choices?"
    - "What types of new data should I collect?"
    - "How can I speed up computation?"

- One of most important problems in ML/statistics, but very messy.
  - For now, we will say a feature is "relevant" if it helps predict y_i from x_i.

"Association" Approach

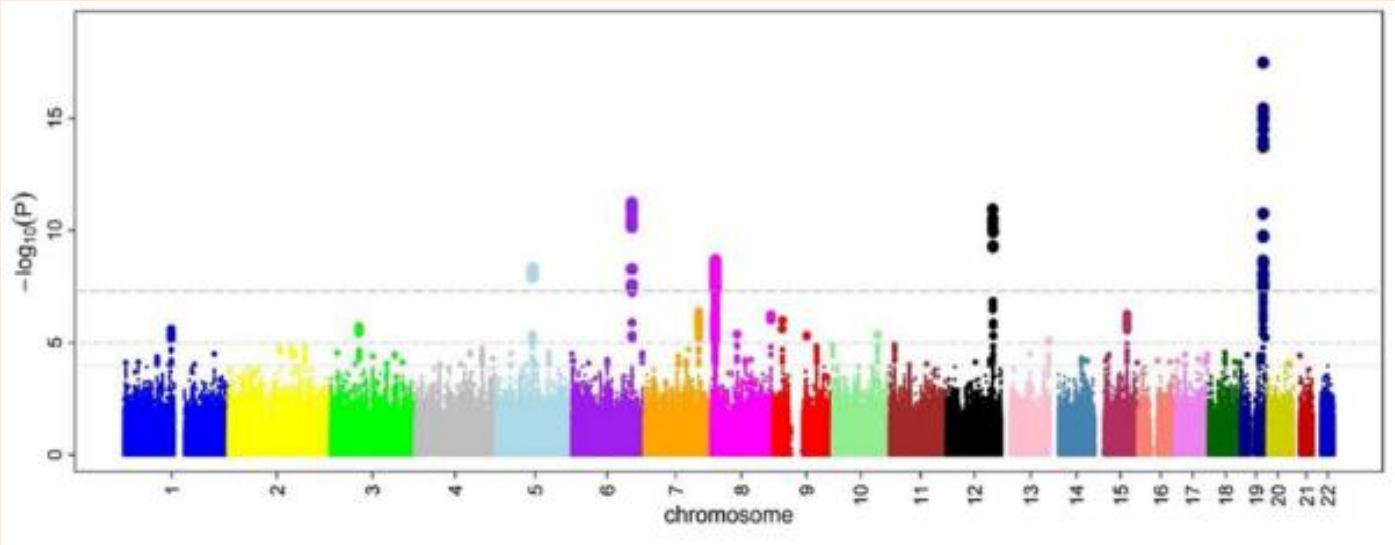
- A simple/common way to do feature selection:
  - For each feature 'j', compute correlation between feature values x^j and 'y'.
    - Say that 'j' is relevant if correlation is above 0.5 or below -0.5.

- Turns feature selection into hypothesis testing for each feature.
    - There are many other measures of "dependence" (Wikipedia).

- Usually gives unsatisfactory results as it ignores variable interactions:
  - Includes irrelevant variables: "Taco Tuesdays".
    - If tacos make you sick, and you often eat tacos on Tuesdays, it will say "Tuesday" is relevant.
  - Excludes relevant variables: "Diet Coke + Mentos Eruption".
    - Diet coke and Mentos don't make you sick on their own, but *together* they make you sick.

Genome-Wide Association Studies

- Genome-wide association studies:
  - Measure if there exists a dependency between each individual “single-nucleotide polymorphism” in the genome and a particular disease.



  - Has identified thousands of genes “associated” with diseases.
    - But *by design* this has a huge numbers of false positives (and many false negatives).

https://en.wikipedia.org/wiki/Genome-wide_association_study

"Regression Weight" Approach

- Another simple/common approach to feature selection:
  - Fit regression weights 'w' based on **all** features (maybe with least squares).
  - Take all features 'j' where weight $|w_j|$ is greater than a threshold.

- For example: you fit a least squares model with 5 features and get:

$$w = \begin{bmatrix} 0.01 \\ -0.2 \\ 10 \\ -3 \\ 0.0001 \end{bmatrix}$$

  - Feature 3 looks the most relevant.
  - Feature 4 also looks relevant.
  - Feature 5 seems irrelevant.

"Regression Weight" Approach

- Another simple/common approach to feature selection:
  - Fit regression weights 'w' based on **all** features (maybe with least squares).
  - Take all features 'j' where weight $|w_j|$ is greater than a threshold.

- This could recognize that "Tuesday" is irrelevant.
  - It could assign a large weight to "tacos", and a small weight to "Tuesday".
    - Since the tacos would "explain" the correlation between "Tuesday" and "sick".
    - Assuming you get enough data, and you sometimes eat tacos on other days. (And the relationship is actually linear.)

"Regression Weight" Approach

- Another simple/common approach to feature selection:
  - Fit regression weights 'w' based on **all** features (maybe with least squares).
  - Take all features 'j' where weight $|w_j|$ is greater than a threshold.

- Has major problems with collinearity:
  - If the "Tuesday" variable always equals the "taco" variable, it could say that Tuesdays are relevant but tacos are not.

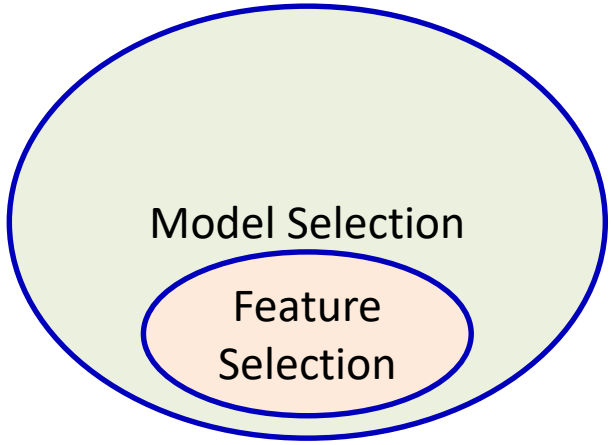
$$\hat{y}_i = w_1 * \text{taco} + w_2 * \text{Tuesday} = 0 * \text{taco} + (w_1 + w_2) * \text{Tuesday}$$

  - If you have two copies of an irrelevant feature, it could take both irrelevant copies.

$$\hat{y}_i = 0 * \text{irrelevant} + 0 * \text{irrelevant} = 10000 * \text{irrelevant} + (-10000) * \text{irrelevant}$$

Digression: "Feature" vs. "Model" Selection?

- Model selection: "which model should I use?"
  - KNN vs. decision tree, depth of decision tree, degree of polynomial basis.
- Feature selection: "which features should I use?"
  - Using feature 10 or not, using x_i^2 as part of basis.



- These two tasks are highly-related:
  - It is a different "model" if we add x_i^2 to linear regression.
  - But the x_i^2 term is just a "feature" that could be "selected" or not.
  - Usually, "feature selection" means choosing from some "original" features.
    - You could say that "feature" selection is a special case of "model" selection.

Next Topic: Search and Score Methods

Can it help prediction to throw features away?

- Yes, because linear regression can overfit with large 'd'.
  - Even though it's "just" a hyper-plane.

- Consider using d=n, with random features: X=randn(n,d).
  - With high probability, you will be able to get a training error of 0.
  - But the features were random, this is completely overfitting.

- You could view "number of features" as a hyper-parameter.
  - Model gets more complex as you add more features.

Search and Score Methods

- Most common feature selection framework is search and score:
  1. Define score function f(S) that measures quality of a set of features 'S'.
  2. Now search for the variables 'S' with the best score.

- Example with 3 features:
  - Compute "score" of selecting only feature 1.
  - Compute "score" of selecting only feature 2.
  - Compute "score" of selecting only feature 3.
  - Compute "score" of selecting only features {1,2}.
  - Compute "score" of selecting only features {1,3}.
  - Compute "score" of selecting only features {2,3}.
  - Compute "score" of selecting all features {1,2,3}.
  - Compute "score" of selecting no features {}.
  - Return the set of features 'S' with the best "score".

Which Score Function?

- The score cannot be the training error.
  - Training error goes down as you add features, so will select all features.

- A more logical score is the validation error.
  - "Find the set of features that gives the lowest validation error."
  - To minimize test error, this is what we want.

- But there are problems due to the large number of sets of variables:
  - If we have 'd' variables, there are 2^d sets of variables.
  - Optimization bias is high: we're optimizing over 2^d models (not 10).
    - So prone to false positives: irrelevant variables will sometimes help by chance.

"Number of Features" Penalties

- To reduce false positives, we can again use complexity penalties:

$$\text{score}(S) = \frac{1}{2}\sum_{i=1}^{n}(w_S^T x_{iS} - y_i)^2 + \lambda \text{size}(S)$$

  - We're using 'x_{iS}' as the features 'S' of example x_i.
  - Above we minimize squared error plus a penalty on number of features.
    - "You can include an extra feature if it reduces training error by at least λ."

- If two 'S' have similar error, this prefers the smaller set.
  - It prefers removing feature 3 instead of having $w_3 = 0.00001$.

- We often this the "L0-norm" instead of writing "size(S)"…

"L0-Norm" and "Number of Features We Use"

- In linear models, setting $w_j = 0$ is the same as removing feature 'j':

$$\hat{y}_i = w_1 x_{i1} + \underbrace{w_2 x_{i2}}_{\text{set } w_2 = 0} + w_3 x_{i3} + \cdots + w_d x_{id}$$

$$\hat{y}_i = w_1 x_{i1} + \underbrace{0}_{\text{ignore } x_{i2}} + w_3 x_{i3} + \cdots + w_d x_{id}$$

- The L0 "norm" is the number of non-zero values ($||w||_0 = \text{size}(S)$).

If $w = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 3 \end{bmatrix}$ then $||w||_0 = 3$

If $w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ then $||w||_0 = 0$.

  - Not actually a true norm.
  - If 'w' has a small L0-norm, then it does not use many features.

L0-penalty: optimization

- L0-norm penalty for feature selection:

$$f(w) = \frac{1}{2}\underbrace{\|Xw - y\|^2}_{\text{training error}} + \lambda \underbrace{\|w\|_0}_{\text{degrees of freedom 'k'}}$$

- Suppose we want to use this to evaluate the features S = {1,2}:
  - First fit the 'w' just using features 1 and 2.
  - Now compute the training error with this 'w' and features 1 and 2.
  - Add λ*2 to the training error to get the score.

- We repeat this with other choices of 'S' to find the "best" features.

L0-penalty: interpretation

- L0-norm penalty for feature selection:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_0$$

- Balances between training error and number of features we use.
  - For λ=0, we get least squares with all features (no penalty on non-zeroes).
  - For λ=∞, we must set w=0 and not use any features (infinite penalty).

  - For other λ, balances between training error and number of non-zeroes.
    - Larger λ puts more emphasis on having zeroes in 'w' (more features removed).
    - Different values give AIC, BIC, and so on.

Forward Selection (Greedy Search Heuristic)

- In search and score, it's also just hard to search for the best 'S'.
  - There are 2^d possible sets.

- A common greedy search procedure is forward selection:

1. Compute score if we use no features.
2. Try adding "taco", "milk", "egg", and so on (computing score of each)
3. Add "milk" because it got the best score.
4. Try {milk, taco}, {milk, egg}, and so on (computing score of each variable with milk)
5. Add "egg" because it got the best score combined with "milk"
6. Try {milk, egg, taco}, {milk, egg, pizza}, and so on...

Forward Selection (Greedy Search Heuristic)

- Forward selection algorithm for variable selection:
  1. Start with an empty set of features, S = [ ].
  2. For each possible feature 'j':
     - Compute scores of features in 'S' combined with feature 'j'.
  3. Find the 'j' that has the best score when added to 'S'.
  4. Check if {S ∪ j} improves on the best score found so far.
  5. Add 'j' to 'S' and go back to Step 2.
     - A variation is to stop if no 'j' improves the score over just using 'S'.

- Runtime of forward selection:
  - We fit $O(d^2)$ models, out of the 2^d possible models with different features
    - Each step requires fitting up to 'd' models, and there are up to 'd' steps.
  - Total cost will be $O(d^2)$ times the cost of fitting an individual model.
    - See bonus for the case of least squares, and how you fit "updated" model faster than re-fitting.
- Not guaranteed to find the best set, but fitting fewer models reduces many problems:
  - Cheaper, overfits less, has fewer false positives.

Backward Selection and RFE

- Forward selection often works better than naïve methods.
- A related method is backward selection:
  - Start with all features, compute score after removing each feature, remove the one that improves the score the most.
- If you consider adding or removing features, it's called stagewise selection.
- Stochastic local search is a class of fancier methods.
  - Simulated annealing, genetic algorithms, ant colony optimization, etc.
- Recursive feature elimination is another related method:
  - Fit parameters of a regression model, prune features with small regression weights, repeat.
- See bonus slide for discussion of feature selection in random forests.

Next Topic: Ambiguity of Feature Selection

Is "Relevance" Clearly Defined?

- Consider a supervised classification task:

gender	mom	dad
F	1	0
M	0	1
F	0	0
F	1	1

SNP
1
0
0
1

- Predict whether someone has particular genetic variation (SNP).
  - Location of mutation is in "mitochondrial" DNA.
    - "You almost always have the same value as your mom".
  - For simplicity we'll assume 1950s-style gender and parentage.

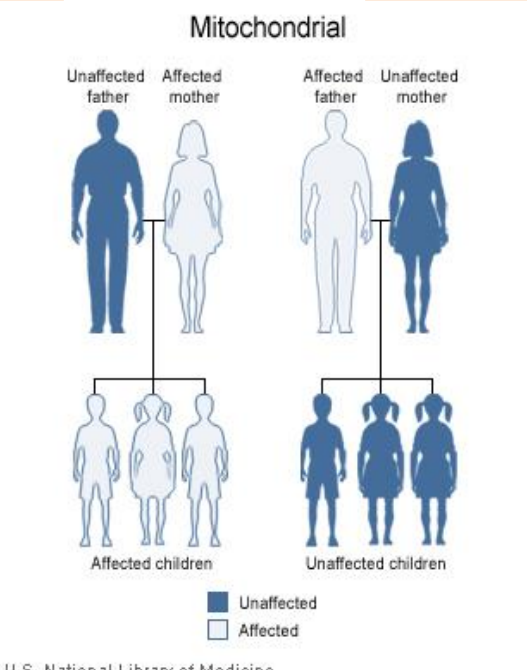
Is “Relevance” Clearly Defined?

- Consider a supervised classification task:

gender	mom	dad
F	1	0
M	0	1
F	0	0
F	1	1

SNP
1
0
0
1

- True model:
  - (SNP = mom) with very high probability.
  - (SNP != mom) with some very low probability.
- What are the “relevant” features for this problem?
  - Mom is relevant and {gender, dad} are not relevant.



https://en.wikipedia.org/wiki/Human_mitochondrial_genetics

Is "Relevance" Clearly Defined?

- What if "mom" feature is repeated?

gender	mom	dad	mom2
F	1	0	1
M	0	1	0
F	0	0	0
F	1	1	1

SNP
1
0
0
1

- Are "mom" and "mom2" relevant?
  - Should we pick them both?
  - Should we pick one because it predicts the other?

Neither of these is "correct", but not picking either is incorrect.

- If features can be predicted from features, can't know which to pick.
  - Collinearity is a special case of "dependence" (which may be non-linear).

Is “Relevance” Clearly Defined?

- What if we add (maternal) “grandma”?

gender	mom	dad	grandma
F	1	0	1
M	0	1	0
F	0	0	0
F	1	1	1

SNP
1
0
0
1

- Is “grandma” relevant?
  - You can predict SNP very accurately from “grandma” alone.
  - But “grandma” is irrelevant if I know “mom”.
    - There is no information gained from “grandma” if you already have “mom”.

Is “Relevance” Clearly Defined?

- What if we don’t know “mom”?

gender	grandma	dad
F	1	0
M	0	1
F	0	0
F	1	1

SNP
1
0
0
1

- Now is “grandma” is relevant?
  - Without “mom” variable, using “grandma” is the best you can do.

- A feature is only “relevant” in the context of available features.
  - Adding/removing features can make features relevant/irrelevant.

Summary

- Feature selection is task of choosing the "relevant" features.
  - Obvious simple approaches have obvious simple problems.
- Search and score: find features that optimize some score.
  - L0-norm penalties are the most common scores.
  - Forward selection is a heuristic to search over a smaller set of features.
- "Relevance" depends on context.
  - Adding/removing features can make things relevant/irrelevant.

- Next time: getting a good test error even with irrelevant features.

Feature Selection in Random Forests

- Decision trees naturally do feature selection while learning:
  - The features used for the splits are the ones that are "selected".

- There are a variety of ways to evaluate features in random forests:
  - Compute proportion of trees that use feature 'j'.
  - Compute average infogain increase when using feature 'j'.
  - Permute all values of feature 'j', and see how "out of bag" error increases.
- You could use any of above to select features from random forest.

Mallow's Cp

- Older than AIC and BIC is Mallow's Cp:

$$f(w) = \frac{\|Xw - y\|^2}{\frac{1}{n}\|X\hat{w} - y\|^2} - n + 2\|w\|_0$$

least squares weights if we used <u>all</u> features.

- Minimizing this score is equivalent to L0-regularization:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_0$$

with $\lambda = \frac{\|X\hat{w} - y\|^2}{n}$

- So again, viewing λ as hyper-parameter, this score is special case.

Adjusted R^2

- Older than AIC and BIC and Mallow's Cp is adjusted R^2:

$$f(w) = 1 - (1 - R^2)\frac{n-1}{n - \|w\|_0 - 1} \quad \text{where} \quad R^2 = 1 - \frac{\|Xw - y\|^2}{\|X\hat{w} - y\|^2}$$

- Maximizing this score is equivalent to L0-regularization:

$$= \frac{1}{2}\|Xw - y\|^2 + \lambda \|w\|_0$$

$$\text{with } \lambda = \frac{\|X\hat{w} - y\|^2}{2(n-1)}$$

- So again, viewing λ as hyper-parameter, this score is special case.

ANOVA

- Some people also like to compute this "ANOVA" quantity:

$$f(w) = \frac{\|Xw - \bar{y}\|^2}{\|y - \bar{y}\|^2}$$

mean of y_i values repeated 'n' times

- This is based on the decomposition of "total squared error" as:

$$\underbrace{\|y - \bar{y}\|^2}_{\text{"total" error}} = \underbrace{\|Xw - \bar{y}\|^2}_{\text{"explained" error}} + \underbrace{\|Xw - y\|^2}_{\text{"residual" (usual) error.}}$$

- Notice that "explained error" goes up as our usual ("residual") error goes down.
- Trying to find the 'k' features that maximize 'f' ("explain the most variance") is equivalent to L0-regularization with a particular λ (so another special case).

Information Criteria with Noise Variance

- We defined AIC/BIC for feature selection in least squares as:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_0$$

- The first term comes from assuming $y_i = w^T x_i + \varepsilon$, where ε comes from a normal distribution with a variance of 1.
  - We'll discuss why when we discuss MLE and MAP estimation.
  - If you aren't doing least squares, replace first term by "log-likelihood".
- If you treat variance as a parameter, then after some manipulation:

$$f(w) = \frac{n}{2}\log(\|Xw - y\|^2) + \lambda\|w\|_0$$

- However, this is again equivalent to just changing λ.

Complexity Penalties for Other Models

- Scores like AIC and BIC can also be used in other contexts:
  - When fitting a decision tree, only split a node if it improves BIC.
  - This makes sense if we're looking for the "true tree", or maybe just a simple/interpretable tree that performs well.

- In these cases we replace "L0-norm" with "degrees of freedom".
  - In linear models fit with least squares, degrees of freedom is number of non-zeroes.
  - Unfortunately, it is not always easy to measure "degrees of freedom".

Cost of Forward Selection

- Each step of forward selection fits up to 'd' model.
- And we do 'd' steps of forward selection.
- So cost of forward selection is $O(d^2)$ times cost of fitting one model.

- For linear regression with squared error, cost is $O(nd^2 + d^3)$.
  - So total cost of forward selection would be $O(nd^4 + d^5)$.

Faster Forward Selection for Least Squares

- Instead of fitting models from scratch, we can often speed up forward selection by re-using computation and/or updating models.

- For linear regression with the squared error:
  - Can reduce $O(nd^4)$ term from repeatedly compute $O(d^2)$ sub-matrices of X^TX:
    - Compute X^TX once for all, then grab relevant sub-matrix for each model.
    - Costs $O(nd^2)$ to compute X^TX, then $O(d^2)$ to grab each sub-matrix.
    - Reduces cost of this step to $O(nd^2 + d^4)$.
  - Can reduce $O(d^5)$ term from solving $O(d^2)$ linear systems involving sub-matrices of X^TX:
    - Each time you add or remove a feature, it is a rank-1 updated to the sub-matrix of X^TX.
    - By storing factorized X^TX sub-matrix, you could do a rank-1 update for $O(d^2)$.
      - And cost of solving a linear system for a factorized matrix is also $O(d^2)$.
    - Total cost is $O(d^4)$ to do this $O(d^2)$ times.
  - So by updating models, can reduce cost from $O(nd^4 + d^5)$ down to $O(nd^2 + d^4)$.
    - Which is similar to cost of solving one least squares problem, particularly if $n \gg d$.

Structure Learning: Unsupervised Feature Selection

- "News" data: presence of 100 words in 16k newsgroup posts:

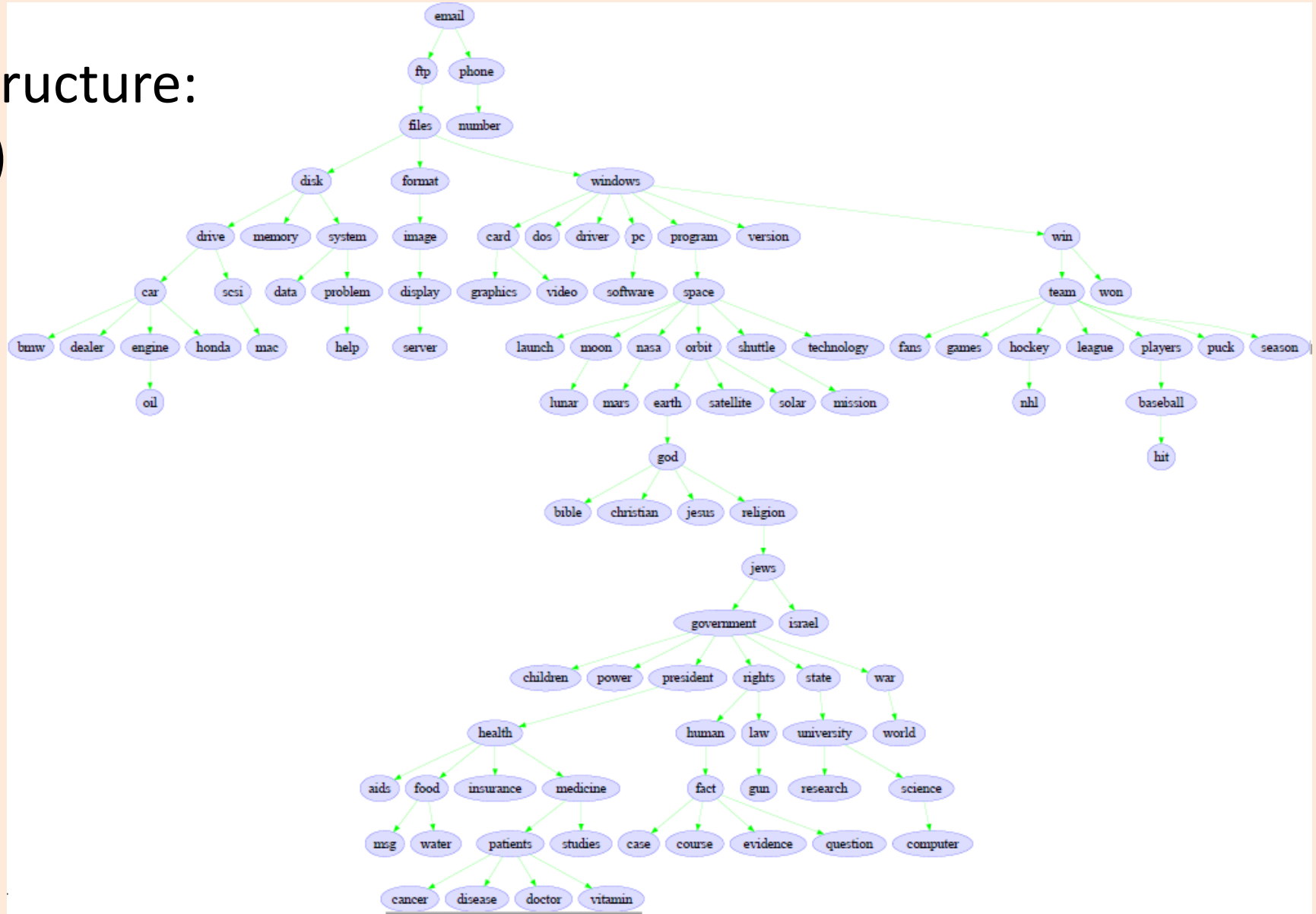
car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- Which words are related to each other?

- Problem of structure learning: unsupervised feature selection.

Structure Learning: Unsupervised Feature Selection

- Optimal tree structure:
  (ignore arrows)



Naïve Approach: Association Networks

- A naïve approach to structure learning ("association networks"):
  - For each pair of variables, compute a measure of similarity or dependence.

- Using these n^2 similarity values either:
  - Select all pairs whose similarity is above a threshold.
  - Select the "top k" most similar features to each feature 'j'.

- Main problems:
  - Usually, most variables are dependent (too many edges).
    - "Sick" is getting connected to "Tuesdays" even if "tacos" are a variable.
  - "True" neighbours may not have the highest dependence.
    - "Sick" might get connected to "Tuesdays" before it gets connected to "milk".

- (Variation: best tree can be found as minimum spanning tree problem.)

Example: Vancouver Rain Data

- Consider modeling the "Vancouver rain" dataset.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	…
Month 1	0	0	0	1	1	0	0	1	1	
Month 2	1	0	0	0	0	0	1	0	0	
Month 3	1	1	1	1	1	1	1	1	1	
Month 4	1	1	1	1	0	0	1	1	1	
Month 5	0	0	0	0	1	1	0	0	0	
Month 6	0	1	1	0	0	0	0	1	1	

- The strongest signal in the data is the simple relationship:
  - If it rained yesterday, it's likely to rain today (> 50% chance that $x^{t-1} = x^t$).
  - But an "association network" might connect all days (all dependent).

Dependency Networks

- A better approach is dependency networks:
  - For each variable 'j', make it the target in a supervised learning problem.

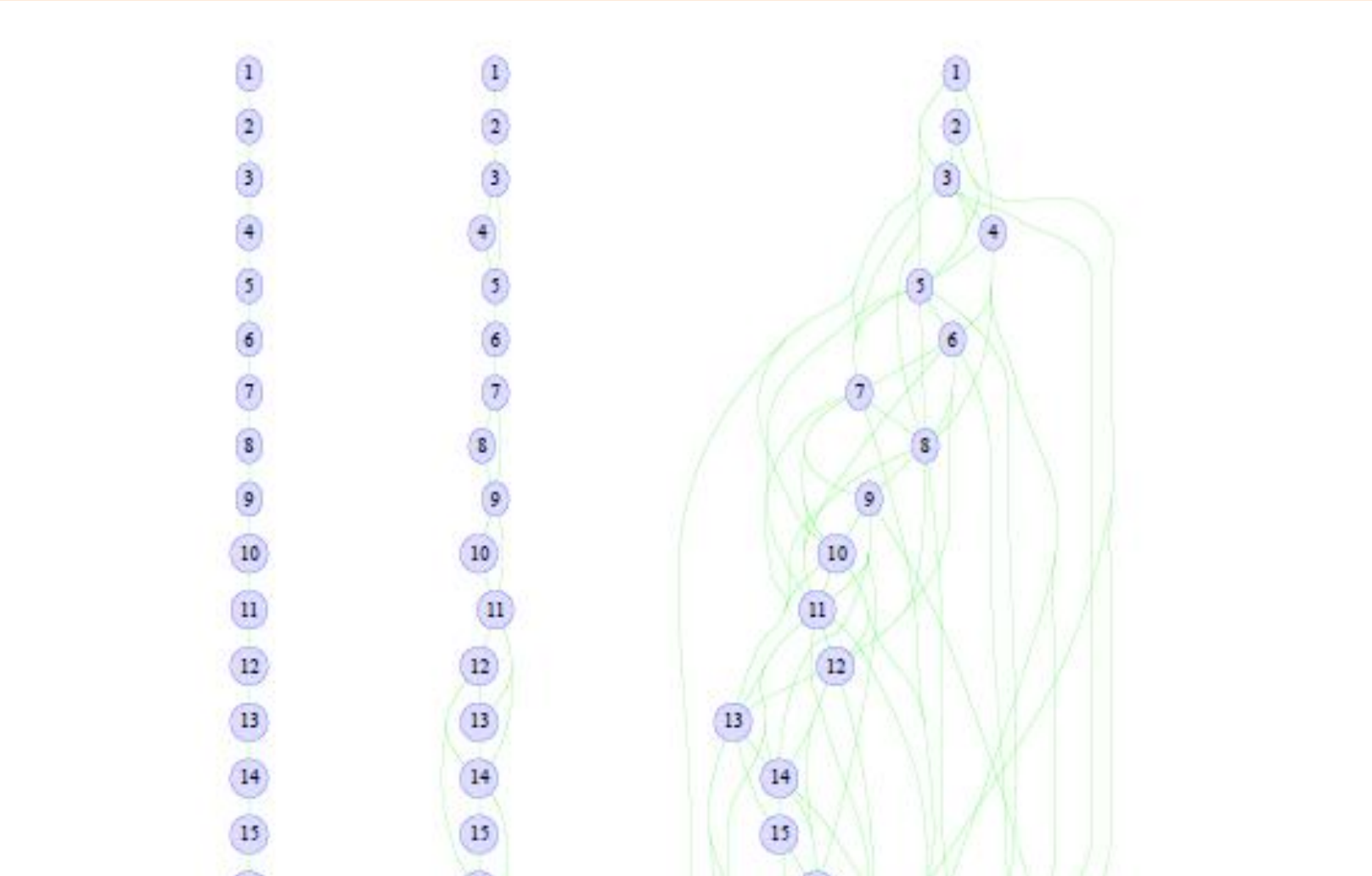
$$X = \begin{bmatrix} | & | & | & | & | \\ x^1 & x^2 & x^3 & x^4 & x^5 \\ | & | & | & | & | \end{bmatrix} \Longrightarrow \bar{X} = \begin{bmatrix} | & | & | & | \\ x^1 & x^2 & x^3 & x^5 \\ | & | & | & | \end{bmatrix} \quad y = \begin{bmatrix} | \\ x^4 \\ | \end{bmatrix}$$

  - Now we can use any feature selection method to choose j's "neighbours".
    - Forward selection, L1-regularization, ensemble methods, etc.

- Can capture conditional independence:
  - Might connect "sick" to "tacos", and "tacos" to "Tuesdays".
    - Without connecting "sick" directly to "Tuesdays".
  - Might connect "grandma" to "mom", and "mom" to "SNP".
    - Without connection "grandma" directly to "SNP".

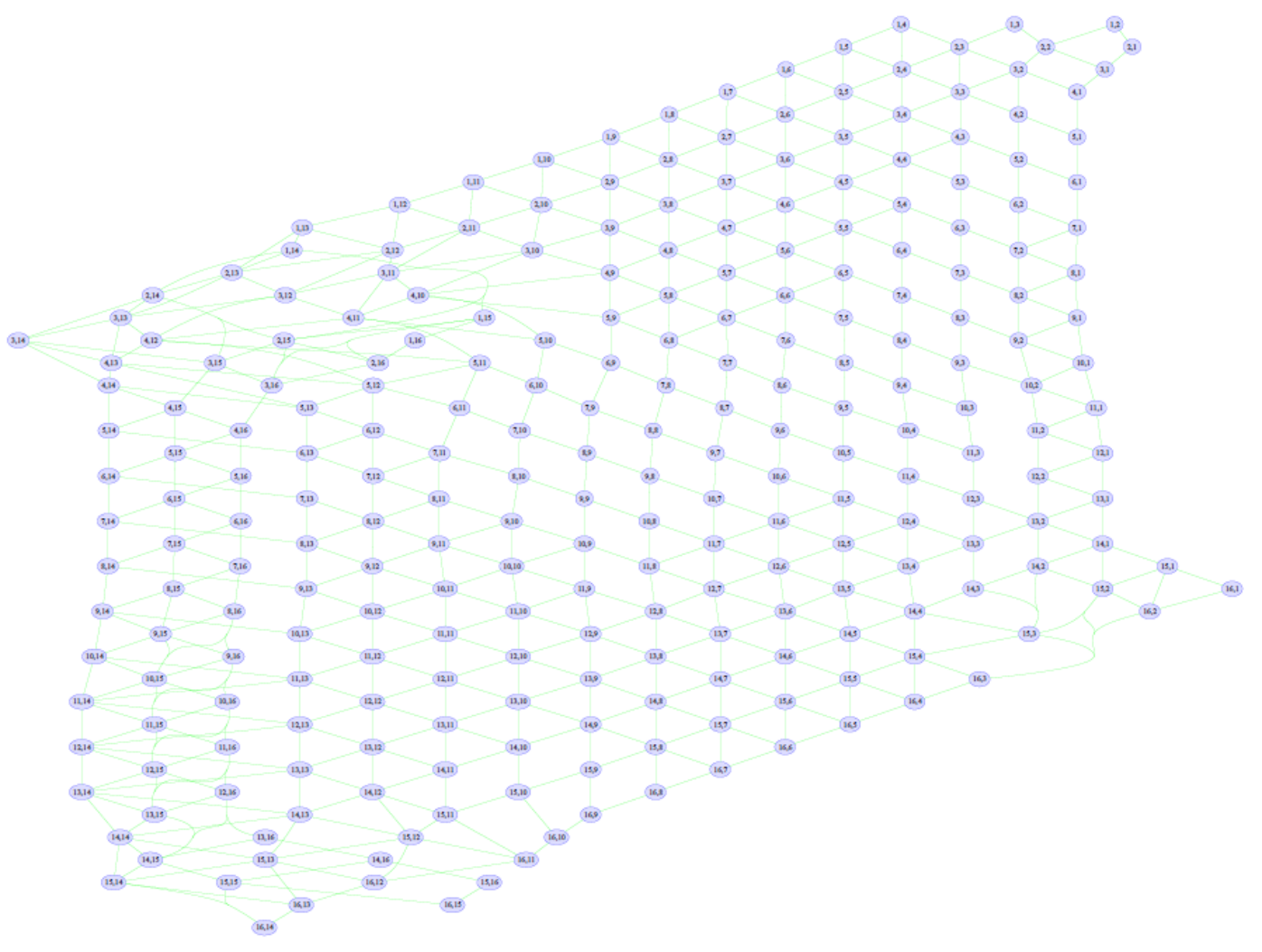
Dependency Networks

- Dependency network fit to Vancouver rain data (different λ values):



Dependency Networks

- Variation on dependency networks on digit image pixels:



Another popular structure learning method is the "PC" algorithm.