CPSC 340: Machine Learning and Data Mining

Feature Selection

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Admin

- Assignment 3:
  - Due Friday

- Midterm:
  - Feb 14 in class
  - 1 page of notes allowed
  - Past exams available on course homepage

Change of Basis Notation

- Linear regression with original features:
  - We use 'X' as our data matrix, and 'w' as our parameters.
  - We can find d-dimensional 'w' by minimizing the squared error:

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

- Linear regression with change of basis:
  - We use 'Z' as our data matrix, and 'v' as our parameters.
  - We can find k-dimensional 'v' by minimizing the squared error:

$$f(v) = \frac{1}{2}\|Zv - y\|^2$$

  - Notice that in both cases the target is still 'y'.

Finding the "True" Model

- What if our goal is find the "true" model?
  - We believe that y_i really is a polynomial function of x_i.
  - We want to find the degree of the polynomial 'p'.

- Should we choose the 'p' with the lowest training error?
  - No, this will pick a 'p' that is way too large.
    (training error always decreases as you increase 'p')

Finding the "True" Model

- What if our goal is find the "true" model?
  - We believe that y_i really is a polynomial function of x_i.
  - We want to find the degree of the polynomial 'p'.

- Should we choose the 'p' with the lowest validation error?
  - This will also often choose a 'p' that is too large.
  - If 'p' is too big then we overfit, but might still get a lower validation error.
    - Another example of 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.

Complexity Penalties

- There are a lot of "scores" people use to find the "true" model.
- Basic idea behind them: put a penalty on the model complexity.
  - Want to **fit the data and have a simple model**.
- For example, minimize training error plus the degree of polynomial.

$$\text{Let } Z_p = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ 1 & x_3 & (x_3)^2 & \cdots & (x_3)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$$

Find 'p' that minimizes:

$$\text{score}(p) = \underbrace{\frac{1}{2}\|Z_p v - y\|^2}_{\text{train error for best 'v' with this basis}} + \underbrace{p}_{\text{degree of polynomial}}$$

  - If we use p=4, use "training error plus 4" as error.
- If two 'p' values have similar error, this prefers the smaller 'p'.
- Can't optimize this using normal equations, since it's discontinuous in 'p'.

Choosing Degree of Polynomial Basis

- How can we optimize this score?

$$\text{score}(p) = \frac{1}{2}\|Z_p v - y\|^2 + p$$

  - Form Z_0, solve for 'v', compute score(1) = $\frac{1}{2}||Z_0 v - y||^2 + 1$.
  - Form Z_1, solve for 'v', compute score(2) = $\frac{1}{2}||Z_1 v - y||^2 + 2$.
  - Form Z_2, solve for 'v', compute score(3) = $\frac{1}{2}||Z_2 v - y||^2 + 3$.
  - Form Z_3, solve for 'v', compute score(4) = $\frac{1}{2}||Z_3 v - y||^2 + 4$.

  - Choose the degree with the lowest score.
    - "You need to decrease training error by at least 1 to increase degree by 1."

Information Criteria

- There are many scores, usually with the form:

$$\text{score}(p) = \frac{1}{2}\|Z_p v - y\|^2 + \lambda k$$

  - The value 'k' is the "number of estimated parameters" ("degrees of freedom").
    - For polynomial basis, we have k = (p+1).
  - The parameter λ > 0 controls how strong we penalize complexity.
    - "You need to decrease the training error by least λ to increase 'k' by 1".

- Using (λ = 1) is called Akaike information criterion (AIC).
- Other choices of λ give other criteria:
  - Mallow's C_p.
  - Adjusted R^2.

Choosing Degree of Polynomial Basis

- How can we optimize this score in terms of 'p'?

$$\text{score}(p) = \frac{1}{2}\|Z_p v - y\|^2 + \lambda k$$

  - Form Z_0, solve for 'v', compute score(0) = $\frac{1}{2}||Z_0 v - y||^2 + \lambda$.
  - Form Z_1, solve for 'v', compute score(1) = $\frac{1}{2}||Z_1 v - y||^2 + 2\lambda$.
  - Form Z_2, solve for 'v', compute score(2) = $\frac{1}{2}||Z_2 v - y||^2 + 3\lambda$.
  - Form Z_3, solve for 'v', compute score(3) = $\frac{1}{2}||Z_3 v - y||^2 + 4\lambda$.

  - So we need to improve by "at least λ" to justify increasing degree.
    - If λ is big, we'll choose a small degree. If λ is small, we'll choose a large degree.

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.
  - λ is just treated as another hyperparameter

(pause)

Motivation: Discovering Food Allergies

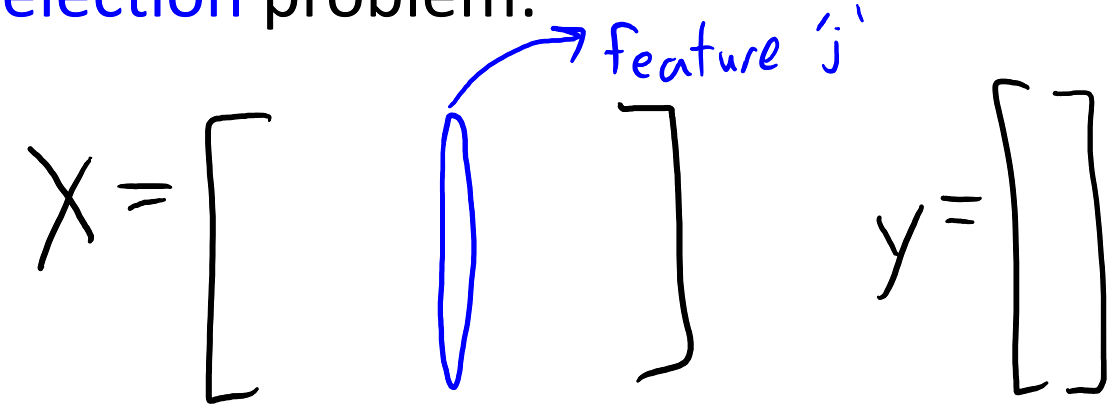
- Recall the food allergy example:

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

- Instead of predicting "sick", 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?"
    - "What 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'll 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 0.9 or below -0.9.
- 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".
    - Diet coke and Mentos don't make you sick on their own, but *together* they make you sick.

“Regression Weight” Approach

- A 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.
  - If you get enough data, and you sometimes eat tacos on other days.
    (And the relationship is actually linear.)

- This could recognize that “Diet Coke” and “Mentos” are relevant.
  - Assuming this combination occurs enough times in the data.

"Regression Weight" Approach

- A 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.

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

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

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

  - We will deal with this next class.

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 using feature 1.
  – Compute “score” of using feature 2.
  – Compute “score” of using feature 3.
  – Compute “score” of using features {1,2}.
  – Compute “score” of using features {1,3}.
  – Compute “score” of using features {2,3}.
  – Compute “score” of using features {1,2,3}.
  – Compute “score” of using features {}.
  – Return the set of features ‘S’ with the best “score”.

Which Score Function?

- The score can't 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).
  - 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:

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

  - E.g., we could use squared error and number of non-zeroes.
  - We're using 'x_{iS}' as the features 'S' of example x_i.

- If two 'S' have similar error, this prefers the smaller set.
  - It prefers having $w_3 = 0$ instead of $w_3 = 0.00001$.

- Instead of "size(S)", we usually write this using the "L0-norm"...

L0-Norm and "Number of Features We Use"

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

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

set $w_2 = 0$

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

- The L0 "norm" is the number of non-zero values.

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 doesn't use many features.

L0-penalty: optimization

- L0-norm penalty for feature selection:

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

- 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.
  - With λ=0, we get least squares with all features.
  - With λ=∞, we must set w=0 and not use any features.
  - With other λ, balances between training error and number of non-zeroes.
    - Larger λ puts more emphasis on having zeroes in 'w' (more feature selection).
    - 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. Continue until no single-variable addition improves the score.

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. If no ‘j’ improves the score, stop.
  4. Otherwise, add the ‘j’ that improves the score the most to ‘S’.
     - Then go back to Step 2.

- Not guaranteed to find the best set, but reduces many problems:
  – Considers $O(d^2)$ models: cheaper, overfits less, has fewer false positives.

Summary

- Information criteria are scores that penalize number of parameters.
  - When we want to find the “true” model.
- 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.

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.

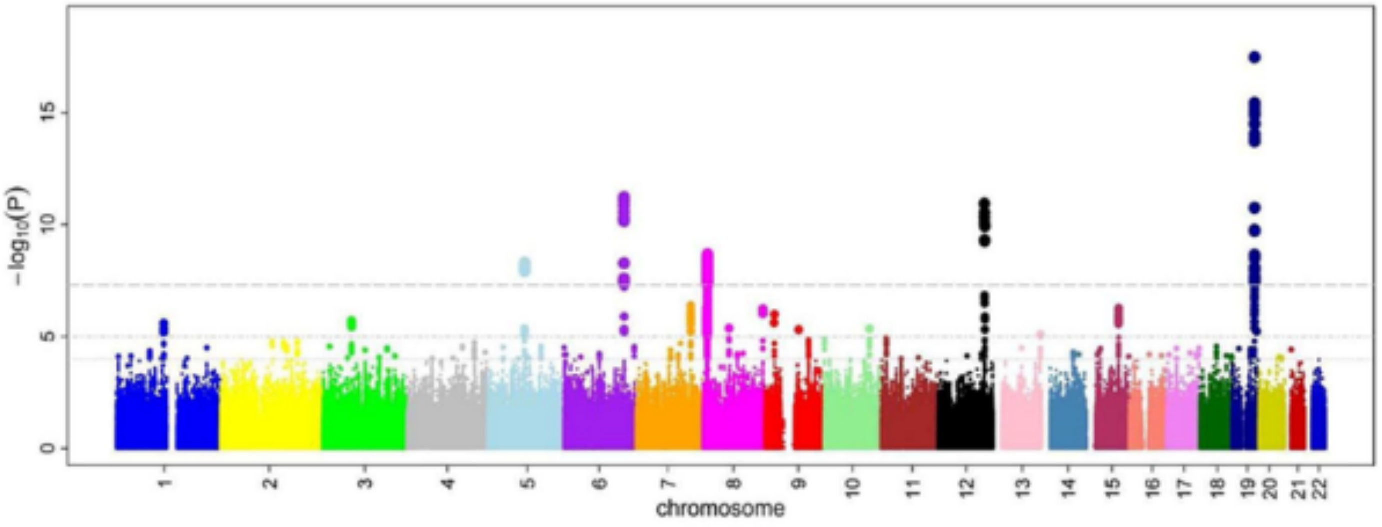
- In these cases we replace "L0-norm" with "degrees of freedom".
  - In linear models fit with least squares, degrees of freedom is number of non-zeroes.
  - Unfortunately, it is not always easy to measure "degrees of freedom".

Discussion of other Scores for Model Selection

- There are many other scores:
  - Elbow method (similar to choosing 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 540).
- These can adapted to use the L1-norm and other errors.

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

Backward Selection and RFE

- Forward selection often works better than naïve methods.
- A related method is backward selection:
  - Start with all features, remove the one that most improves the score.
- If you consider adding or removing features, it's called stagewise.
- Stochastic local search is a class of fancier methods.
  - Simulated annealing, genetic algorithms, ant colony optimization, etc.
- Recursive feature elimination (RFE) is another related method:
  - Fit parameters of a regression model.
  - Prune features with small regression weights.
  - Repeat.