Infinite Feature Selection

Chapman Siu

4 September 2017

Outline

- Gradient Descent Approaches
- Statistical Approaches
- Markov Chain

Unless stated otherwise. . . we will only be dealing with linear models

Motivation

How do we get infinite features?

1. Feature engineering (creating more feature maps/adding new layers to a neural network? Maybe?)
2. Adding new sources of data - e.g. IoT devices

Why should we reduce feature set?

- Curse of dimensionality
- Spurious correlations - makes it harder to determine causal factors

1. Gradient Descent Approaches

Almost all approaches hinge on regularization!

- Grafting
- Online Substitution Algorithm
- Minimal Description Length

1. Reviewing Regularization. . . Quickly

Regularization term is typically

$$\Omega_p(\theta) = \lambda \sum_{i=1}^{m} \alpha_i |\theta_i|^p$$

Common regularizers are

- Ω_1 which is the "lasso" penalty
- Ω_2 which is the regularizer commonly seen in ridge regression.
- $\alpha_1\Omega_1 + \alpha_2\Omega_2$ which is elasticnet, typically chosen where $\alpha_1, \alpha_2 > 0$ and $\alpha_1 + \alpha_2 = 1$

Selecting $p < 2$ forces subset of weights to be exactly zero - and thus is used for feature selection.

n.b. regularization generally requires scaled data first

1. Grafting

Makes use of l_1 regulariser.

For an incoming feature:

1. Determine if it will arrive and have non-zero weight
2. Determine if it will cause existing feature(s) to have zero weight

Given our regularizer penalty λ, a new feature w_j is accepted if:

$$|\frac{\partial \bar{\mathcal{L}}}{\partial w_j}| > \lambda$$

Where $\bar{\mathcal{L}}$ is the mean loss.

Intuitively, if the reduction in mean loss outweighs the regularizer incoming feature is chosen, otherwise it is discarded.

1. Grafting - an equivalent formulation

For linear models, grafting is the same as boosting a weak learner f

$$f(\mathbf{x}) = w_0 + w_1 x_i$$

Where x_i is a new incoming feature.

Implications

Feature can be grafted if it can be assumed to be independent from other predictors.

1. Grafting - Future considerations

Grafting paper also showed grafting beyond linear models and in MLP Neural Networks

We can perhaps extend the boosting ideas to "graft" groups of features, or non-linear models. Perhaps grafting trees? This could propose:

- Online Gradient Boosted Trees, using online tree algorithms (e.g. VFDT)

1. Other Variants

Online Substitution Algorithm:

1. Have a fix number of features in model
2. Stream new features and substitute them if it appears to do "better" when added (or when one can be removed)

Substitution is based on magnitude of weights and partial gradient descent update

. . . and several others. . .

2. Statistical Approaches

- Criterion Based (change in likelihood)
- Similarity Based (similar observations/features)

2.1. Streaming Criterion Approaches

$$-2\log(\text{likelihood}) + F$$

Common criterions which fall under this model include:

- Akaike information criterion, with $F = 2m$
- Bayesian information criterion, with $F = m\log(n)$
- risk inflation criterion, with $F = 2m\log(m)$

Where m is the number of features in the model.

A feature is then added/removed, if the *change* in log-likelihood from adding the feature is greater/less than the penalty.

2.1. α-investing

We can frame the change in log likelihood to be equivalent to t-statistic - i.e. we can set up a test to be greating that some α.

Goal: Have an *adaptive* α for adding features...

Give parameters w (wealth) and α_Δ, initialize w_0 to be some value (say 0.05), and set $\alpha_i = w_i/2i$, where i represents ith iteration.

1. For an incoming feature x_i, if pvalue$(x_i) < \alpha_i$:
   1.1 $w_{i+1} \leftarrow w_i + \alpha_\Delta$
2. $w_{i+1} \leftarrow w_{i+1} - \alpha_i$

Effect is if a feature is accepted, then we *invest* in wealth (accept more). If it gets rejected, we do not invest.

2.1. α-investing

- Does not evaluate redundancy - only the immediate relevancy
- Can not remove a feature if it turns out to be redundant in the future

2.2 Similarity

- Unsupervised (dis)Similarity
- Spectral Analysis (Online Group Feature Selection)

2.2.1 Unsupervised Feature Selection using Similarity

1. Calculate feature level similarity
2. Construct a dendogram and traverse to discover and rank sets of most dissimilar features

2.2.2 Spectral Feature selection

Step 1: Calculate observation level similarity

Step 2: Construct Graph

Using the similarity matrix, construct a graph where:

- Nodes: each observation
- Edges: The weight of similarity as calculated in step 1

Edge weights can be set to 0 if two observations are "different" (e.g. have different classification labels, or are different in regression sense)

Step 3: Rank features using some feature-ranking function

There are many choices that can be made here...

2.2.3 Online Group Feature Selection (OGFS)

Focus on *relevancy* and *redundancy* in groups of features. Two step process:

1. Select relevant features using similarity approaches above
2. Remove redundant features using regularization approaches

3. Miscellaneous

- Ensemble Methods
- Markov Chain

3.1 Ensemble Methods

Similar to grafting - combine models together which have different feature sets. This is used in Kaggle and works due to NFL theorem.

Stacking

If β_k are our weights for the kth model and g_k is our k model, then stacking is:

$$\sum_{k=1}^{K} \beta_k g_k(x)$$

where the base learner is a linear model.

One can construct ensembles using neural networks or any other learner. If this is done it is a **superlearner**.

Putting muliple "layers" of stacked or superlearners is called **blending**

3.2 Markov Chain

- RJMCMC
- Birth Death Allocation Model

3.2.1 RJMCMC

Model determination problem - parameter of model is birth, death state of new features.

Set up is a MH problem, where probability is:

$$\min\{1, (\text{Bayes Factor}) \times (\text{prior ratio}) \times (\text{proposal ratio}) \times (\text{Jacobian})\}$$

- Extremely flexible
- Probably not optimal without very careful choice of priors

3.2.2 Birth-Death Feature Allocation Process (BDFP)

Extremely similar to above - but without RJMCMC (it is merely a Markov Chain)

Suppose s is a feature allocation with K_s nonzero features and s' is another feature allocation that differs from s in having one additional feature of size $|a|$, so that $K'_s = K_s + 1$

The state transition for *birth* is defined to be the transition rate as

$$q_{ss'} = R\frac{(|a|-1)!(N-|a|)!}{N!}$$

In comparison the state transition for death is

$$q_{s's} = \frac{Rr}{\alpha}$$

Where R is the parameter governing the birth rate and $\frac{R}{\alpha}$ is the parameter governing the death rate.