Feature Engineering, Model Evaluations

Giri Iyengar

Cornell University

gi43@cornell.edu

Feb 5, 2018

Overview

Overview

1. ETL
2. Feature Engineering
   - Handling Numerical Features
   - Handling Categorical Features
3. Dealing with correlated features
4. Feature Selection
   - Forward and Backward Selection
   - Via Regularization
5. Evaluating Model Performance
   - Handling rare classes

ETL: Preparing your data for modeling

- Frequently, the data that you collect is in **long** format
- For example, logging systems will put timestamps, some entity id and then a few columns
- In addition, you might need to **join** multiple inputs together
- Further, you may want to apply some transformations / aggregations to the individual columns
- All of these can be considered jointly as an **ETL** operation
- Frequently in Data Science, we end up with **ELT**. That is, the Transformation step is done closer to modeling stage

ETL: Preparing your data for modeling

	Month	Day	Feature	Value
1	5	1	Ozone	41
2	5	2	Ozone	36
3	5	5	Ozone	NA
...				
610	9	28	Temp	75
611	9	29	Temp	76
612	9	30	Temp	68

Pivot

Typical output found in a log file. You need to convert this into usable formats by pivoting the individual rows

Month	Day	Ozone	Solar.R	Wind	Temp
5	1	41	190	7.4	67

Overview

Feature Engineering

- In addition to pivoting, you might look at various aggregations / combinations
- How many times did this event occur
- In the past **time window** at this **location** how many times did the **event** occur
- How much away from the mean does this event deviate
- Usually task dependent – with deep subject matter expertise, you ask more meaningful questions

Numerical Features

- Count – How many times did something happen
- Amount – How much was it
- Continuous – Both positive and negative values have meaning

Normalization techniques

- Standardize the numerical variables
- Min-max normalization
- sigmoid / tanh / log transformations
- Handling zeros distinctly – potentially important for Count based features
- Decorrelate / transform variables

One Hot Encoding

- Transforms an attribute taking d distinct values into a $d-1$ dimensional binary vector
- Simple, Easy and gives a great deal of interpretability of the attribute
- Great technique for small number of distinct values
- Categorical attribute with large number of distinct values becomes an issue (e.g. zipcode/MSA/items)

Label			
Dog	1	0	0
Cat	0	1	0
Mouse	0	0	1

Target Rate Encoding

- Compute $P(c|\mathbf{d})$ empirically
- Replace attribute by a $k-1$-dimensional vector (one for each target class)
- Statistics need to be robust enough (take care of distinct values with small counts)
- Handling missing values – those not in training partition

Feature	Target
Up	1
Up	0
Down	1
Flat	0
Flat	1
Down	0
Up	1
Down	0

Feature	Encoding
Up	0.66
Down	0.33
Flat	0.5

TF-IDF and related techniques

- Use summary statistics to reduce the number of distinct values
- Use Informative measures such as TF-IDF (`https://en.wikipedia.org/wiki/TfĐidf`)
  - TF and IDF can both be computed in several ways
  - TF: raw, boolean, log-scaled, augmented (to account of document size variations)
  - IDF: raw, log-scaled, smoothed log-scaled, . . .
- Entropy ($-p_i \log p_i$) to select distinct values of importance

Low Rank Factorizations

- **low-rank** approximation is an optimization technique
- **Cost** is the closeness of fit between original matrix and **low-rank** approximation
- PCA/SVD
- Non-negative Matrix Factorization (NMF) $\Theta^T X \approx Y$. Use Gradient Descent, for instance

LSH/Random Hashing

- When dimensionality is really large, often times a random low-dim projection seems to work
- That is, take your really large dimensional vector and project it down to a really small space randomly
- Collisions will occur but they don't seem to matter. Interpretability is lost, of course
- LSH (Locality Sensitive Hashing) is a more principled approach that does the same
- Several types of projections (Walsh-Hadamard matrix, random Gaussian matrix) have proven distance preserving properties
- `https://en.wikipedia.org/wiki/Locality-sensitive_hashing`

Overview

PCA/SVD

- Useful technique to analyze the inter-relationships between variables
- Feature dimensionality reduction with minimum loss of information
- $y_i = \sum_{j=1}^{k} \beta_{ij} x_j$
- $y_i = \beta_i^T X$, $Y = \beta^T X$
- Σ covariance matrix of X and Σ_Y covariance matrix of Y

PCA

- $\Sigma_Y = \beta^T \Sigma \beta$
- $var(y_i) = \sum_{p=1}^{k} \sum_{m=1}^{k} \beta_{ip} \beta_{im} \sigma_{pm} = \beta_i^T \Sigma \beta_i$
- $covar(y_i, y_j) = \sum_{p=1}^{k} \sum_{m=1}^{k} \beta_{ip} \beta_{jm} \sigma_{pm} = \beta_i^T \Sigma \beta_j$
- Spectral Decomposition Theorem: $\Sigma = \beta D \beta^T$, where β contains the eigenvectors
- Easy to show that $var(y_i) = \lambda_i$ and $covar(y_i, y_j) = 0$
- $trace(\Sigma) = \sum_{i=1}^{k} \sigma_i^2 = \sum_{i=1}^{k} \lambda_i$
- Total variance of the data is accounted in the eigenvalues. Choosing the top $l << k$ eigenvalues and their corresponding eigenvectors gives you a distortion minimizing low-dimensional projection

Linear Discriminant Analysis

- PCA comes up with a projection that represents the data in a low-dimensional feature space
- It is optimized towards **representation**. We are interested in **discrimination**

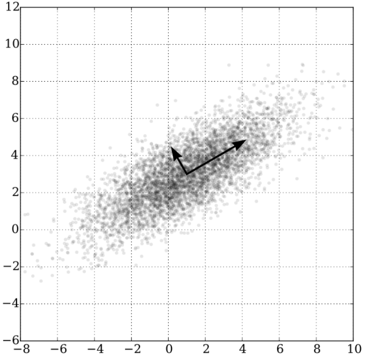


Figure: Gaussian Data

PCA vs LDA: An Example

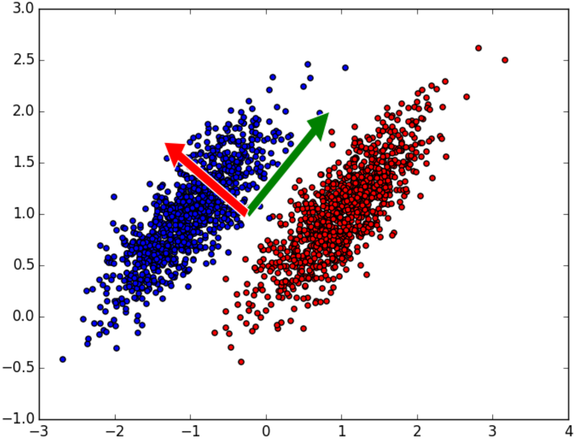


Figure: PCA vs LDA comparison

LDA

- Motivated from Bayes' Rule: $P(c|\mathbf{x}) = \frac{P(\mathbf{x}|c)P(c)}{\sum_d P(\mathbf{x}|d)P(d)}$
- Assume Gaussian distributed data
- Further assume that, classes have **identical** covariance
- End up with: $f_i = \mu_i \Sigma^{-1} x^T - \frac{1}{2}\mu_i \Sigma^{-1} \mu_i^T + ln(P(i))$ which is a line
- See `http://people.revoledu.com/kardi/tutorial/LDA/` for a nice tutorial
- See `http://www.ics.uci.edu/%7Ewelling/classnotes/papers_class/Fisher-LDA.pdf` for a detailed explanation
- The most discriminant projection comes out as eigenvectors of $\Sigma^{-1}\Sigma_b$

Overview

Feature Selection

- Not all features are important
- Often times, removing features from the model improves its performance
- If there are N features, there are 2^N subsets $\implies$ feature selection is a hard task!

Forward Feature Selection

- Build a model with one feature
  - From features, build models in turn and retain the model with the strongest performance
- From the remaining features, keep adding one feature at a time (using search method above)
- Stop if there is no **significant** improvement

Backward Feature Selection

- Similar to Forward Selection Procedure
- Build model with all features
- Remove the **weakest** feature
- Keep removing as long as there is no **significant** change

Regularization

- The more parameters a model has, the more flexible it is
- Too much flexibility leads to overfitting. Model learns the peculiarities of a particular data set and doesn't generalize well
- Solution: Simplify the model
- *Everything must be made as simple as possible, but not simpler!*
- We can add a penalty term to the objective function and do joint minimization
- $\underset{\Theta}{argmin} J(\Theta) + \lambda||\Theta||$

$\mathbf{L}_2$ Regularization / Ridge Regression

- $\underset{\Theta}{argmin} J(\Theta) + \lambda \sum_i \theta_i^2$

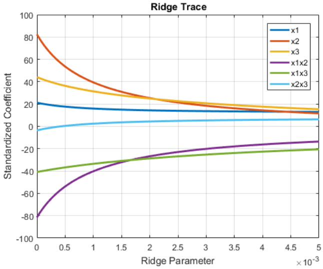


Figure: θ values as a function of λ

In the end, you get regularized weights of features. Some weights are 0 or close to 0. These features are not important to the model.

$\mathbf{L}_1$ Regularization / LASSO

- $\underset{\Theta}{argmin} J(\Theta) + \lambda \sum_i |\theta_i|$

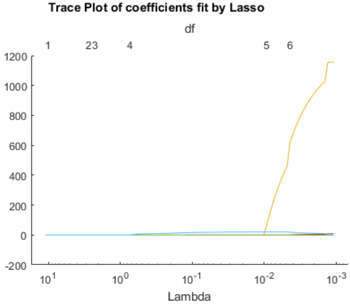


Figure: θ values as a function of λ

`http://statweb.stanford.edu/~owen/courses/305/Rudyregularization.pdf`

Geometric View of Lasso and Ridge

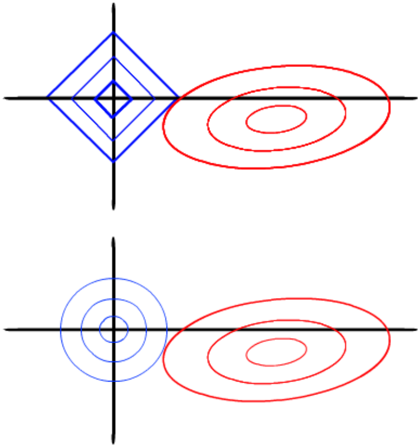


Figure: Lasso and Ridge Regularization

Elastic Net Regularization

- If there are a set of highly correlated features, Lasso ends up selecting only one of them
- When we have high dimensionality, low sample count problem, Lasso saturates quickly
- Ridge doesn't achieve the sparsity achieved by Lasso
- $\underset{\Theta}{argmin} J(\Theta) + \lambda_1 \sum_i |\theta_i| + \lambda_2 \sum_i \theta_i^2$
- Combines the advantages of both Lasso and Ridge, especially for large dimensionality problems

Overview

Model Evaluation

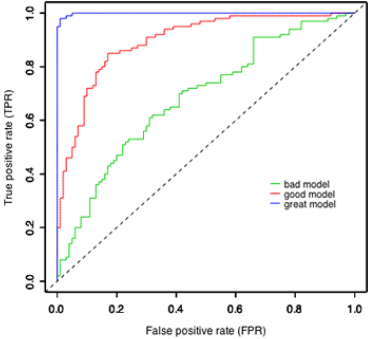
- Accuracy. $\frac{\#Correct}{\#Samples}$
- Confusion matrix

$$\begin{vmatrix} & Pred. + ve & Pred. - ve \\ Actual + ve & TP & FN \\ Actual - ve & FP & TN \end{vmatrix}$$

- Precision: $\frac{TP}{TP+FP}$, Recall: $\frac{TP}{TP+FN}$

Receiver Operating Characteristic

- Precision and Recall reflect a particular setting of decision thresholds
- E.g. Logistic Regression Score / Naive Bayes Neg. Log Likelihood - select any threshold you want
- ROC curve shows the performance for any possible choice of this threshold. AUC – Normalized under ROC curve is another number you can use to compare models



Handling rare class classification

- Often, the class of interest is very rare
- E.g. detecting a rare disease
- Identifying a purchaser from the average internet browser
- Default answer of **No** has a very high accuracy
- But, that is not a very useful classifier

Handling rare class classification

- Penalize different classes differently
- Subsample the background class / Up-sample the foreground class
- Construct hierarchical / back-off classifiers (Russian doll model)
    - Browser / Site Visitor
    - Shopping Cart / Site Visitor
    - Purchase / Shopping Cart

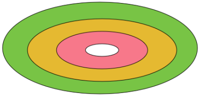


Figure: Russian Doll Model


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
спасибо
bedankt
dziękuję
obrigado
hvala
mauruuru
sukriya
kop khun krap
go raibh maith agat
mochchakkeram
tapadh leat
dank je
sagolun
taiku
terima kasih
grazie
arigatô
takk
dakujem
мерси
ありがとう
감사합니다
ευχαριστώ
merci