GENERAL ASSEMBLY

FEATURE ENGINEERING & UNBALANCED CLASSES

Matt Brems, Data Science Immersive

AGENDA

- Feature Engineering

- Feature Selection – Why?
- Feature Selection – Methods

- Feature Scaling – Why?
- Feature Scaling – Methods

- Unbalanced Classes – Examples
- Unbalanced Classes – Methods

WHAT IS FEATURE ENGINEERING?

- The Process of Feature Engineering:

1. Brainstorming or testing features.
2. Deciding what features to create.
3. Creating features.
4. Checking how the features work with your model.
5. Improving features (if needed).
6. Return to step 1 until complete.
7. Do data science!

Source: https://www.youtube.com/watch?v=drUToKxEAUA

WHAT IS FEATURE ENGINEERING?

- Feature engineering is the term broadly applied to the creation and manipulation of features that are relevant for machine learning algorithms.

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

- Two areas of feature engineering are feature selection and feature scaling.

WHY FEATURE SELECTION?

- We have many potential features that may be used, but only some may have predictive power.
- Think about text data or image data – we generate/produce many features but only some have predictive power.
- Multiple types of feature selection. Most common: best subset, forward, backward.
- Selection vs. extraction: Selection chooses from current features, extraction produces interactions of current features (linear or non-linear combinations)

BEST SUBSET FEATURE SELECTION

- Suppose we have p features. We can then build all possible combinations of models and compare them.

- In action (adapted from ISLR):
- 1. For $k = 0,1, \ldots, p$, fit all $\frac{p!}{k!(p-k)!}$ models that contain exactly k predictors. Pick the best model among each k and "save" that model as M_k.
- 2. Compare $M_0, M_1, \ldots, M_p$ based on some predefined metric.

- What are downfalls to this?

FORWARD FEATURE SELECTION

- Suppose we have p features.

- In action (adapted from ISLR):
- 1. Build M_0, the *null model*, where $Y \sim 1$ (a.k.a. intercept-only model).
- 2. Build every model with the intercept and one feature. Identify the best model, and save as M_1.
- 3. Using M_1 as the starting point, now build every model with the intercept and two features. Save the best model as M_2.
- 4. Repeat for all M_k, $k = 0, \ldots, p$.
- 5. Identify the best model M_k.

- What are downfalls to this?

BACKWARD FEATURE SELECTION

- Same as forward – but start with the full model! Suppose we have p features.

- In action (adapted from ISLR):
- 1. Build M_p, the *full model*, where $Y \sim X + 1$ (contains all features).
- 2. Build every model with the intercept and $p - 1$ features. Identify the best model, and save as M_{p-1}.
- 3. Using $p - 1$ as the starting point, now build every model with the intercept and $p - 2$ features. Save the best model as M_{p-2}.
- 4. Repeat for all M_k, $k = 0, \ldots, p$.
- 5. Identify the best model M_k.

- What are downfalls to this?

COMMON METRICS FOR FEATURE SELECTION

- When identifying what model is "best," R^2 likely won't get us the best model – why?

- 1. Cross-validated MSE or other prediction error.
- 2. R^2_{adj} (want highest value)
- 3. Mallow's C_p (want lowest value)
- 4. AIC (Akaike information criterion) (want lowest value)
- 5. BIC (Bayesian information criterion) (want lowest value)

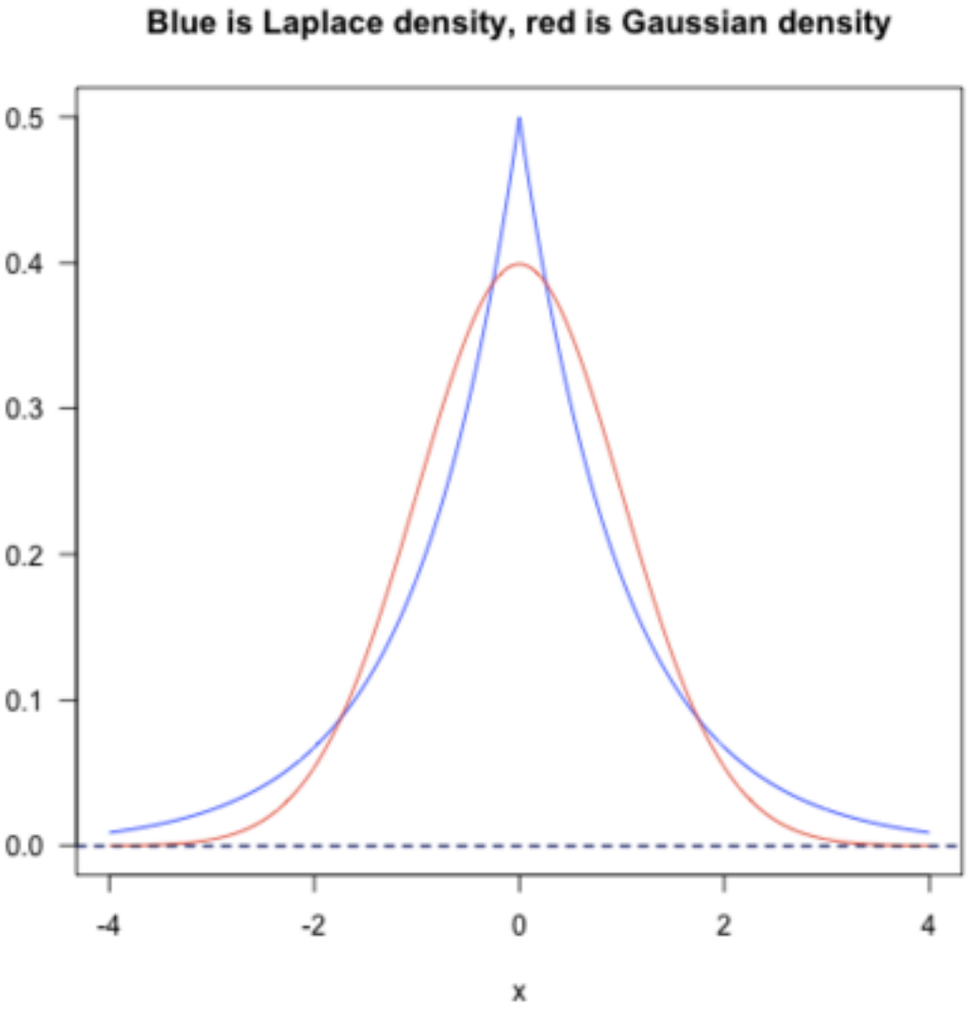
- These are all discussed in ISLR.

REGULARIZATION

- Regularization is an example of a top-down technique that works with parametric models (such as Logistic Regressions and Support Vector Machines). It imposes a global constraint on the values of the parameters that define the model. The regularized model is found solving a new minimization problem where two terms are present: the term defining the model and the term defining the regularization.

L1 AND L2 REGULARIZATION

- Regularization works by adding the penalty associated with the coefficient values to the error of the hypothesis. This way, an accurate hypothesis with unlikely coefficients would be penalized while a somewhat less accurate but more conservative hypothesis with low coefficients would not be penalized as much.



L1 AND L2 REGULARIZATION

- L1: Stricter constraints: more angular distribution for inclusion
- L2: Looser constraints, increases inclusion of features

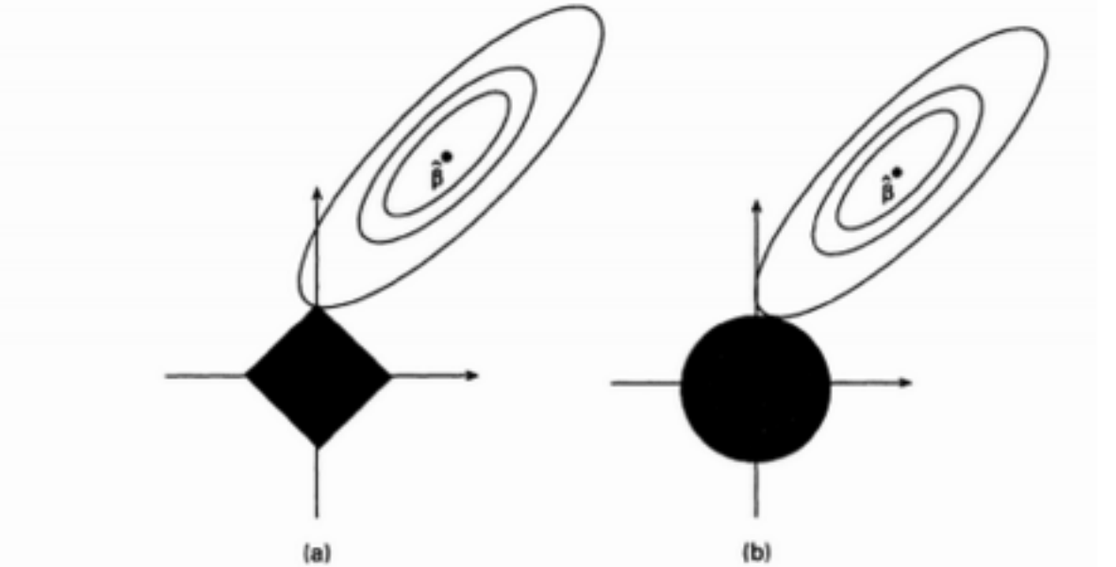


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

feature_selection.GenericUnivariateSelect([...])
feature_selection.SelectPercentile([...])
feature_selection.SelectKBest([score_func, k])
feature_selection.SelectFpr([score_func, alpha])
feature_selection.SelectFdr([score_func, alpha])
feature_selection.SelectFromModel(estimator)
feature_selection.SelectFwe([score_func, alpha])
feature_selection.RFE(estimator[, ...])
feature_selection.RFECV(estimator[, step, ...])
feature_selection.VarianceThreshold([threshold])

FEATURE SCALING

- Feature scaling describes a set of techniques used to counteract the effects of variables having different ranges or scales.

- Certain models rely on “distance” between observations to make predictions. This problem is particularly exacerbated in clustering.

- Scaling protects us from particular features having a disproportionate effect on our data.

WHEN TO FEATURE SCALE?

- We generally use feature scaling when models rely on distance to make predictions. (SVMs, clustering, regularization.)
- We **don't** use feature scaling when models examine features separately. (Tree-based methods.)
- If you're in doubt, build models under both cases and see if your errors/predictions change substantially.

METHODS OF FEATURE SCALING

- sklearn.preprocessing.**StandardScaler()**

$$x_{new} = \frac{x_{old} - \bar{x}}{s_X}$$

- sklearn.preprocessing.**MinMaxScaler()**

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

Statistics (min/max/mean/stdev) are all determined within an individual column.

BALANCED CLASSES

- In classification problems, methods generally work well when we have roughly equally-sized classes. (i.e. In binary problems, 50% in the positive class and 50% in the negative class.)

- However, there are many cases where this isn’t true.

- One example of poor performance with unbalanced classes: logistic regression.

METHODS FOR HANDLING BALANCED CLASSES

- Bias correction. (Gary King of Harvard wrote a paper on a bias correction when working with rare event data, but to my knowledge this isn't available in sklearn or statsmodels and this is pretty theoretical.)
- Over/undersampling.
- Weighting observations. (i.e. weighted least squares)
- Stratified cross-validation.
- Changing threshold for classification.
- Purposefully choosing evaluation metrics.