Regularization
Feature scaling

Regularization

- Before we dive straight into coding regularization with Scikit-Learn, we need to discuss a few more relevant topics:
  - Feature Scaling
  - Cross Validation

Regularization

- Before we dive straight into coding regularization with Scikit-Learn, we need to discuss a few more relevant topics:
  - Feature Scaling
  - Cross Validation

Regularization – Feature Scaling

- Feature scaling provides many benefits to our machine learning process!
- Some machine learning models that rely on distance metrics (e.g. KNN) **require** scaling to perform well.
- Let's discuss the main ideas behind feature scaling...

Regularization – Feature Scaling

- Feature scaling improves the convergence of steepest descent algorithms, which do not possess the property of scale invariance.
- If features are on different scales, certain weights may update faster than others since the feature values x_j play a role in the weight updates.

Regularization – Feature Scaling

- Critical benefit of feature scaling related to gradient descent.
- There are some ML Algos where scaling won’t have an effect (e.g. CART based methods).

Regularization – Feature Scaling

- Scaling the features so that their respective ranges are uniform is important in comparing measurements that have different units.
- Allows us directly compare model coefficients to each other.

Regularization – Feature Scaling

- Feature scaling caveats:
  - Must always scale new unseen data before feeding to model.
  - Effects direct interpretability of feature coefficients
    - Easier to compare coefficients to one another, harder to relate back to original unscaled feature.

Regularization – Feature Scaling

- Feature scaling benefits:
  - Can lead to great increases in performance.
  - Absolutely necessary for some models.
  - Virtually no “real” downside to scaling features.

Regularization – Feature Scaling

- Two main ways to scale features:
  - Standardization
    - Rescales data to have a mean (μ) of 0 and standard deviation (σ) of 1.
  - Normalization
    - Rescales all data values to be between 0-1.

Regularization – Feature Scaling- standardization

- Standardization:
  - Rescales data to have a mean (μ) of 0 and standard deviation (σ) of 1 (unit variance).

$$X_{changed} = \frac{X - \mu}{\sigma}$$

Regularization – Feature Scaling- standardization

- Standardization:
  - Namesake can be confusing since this is also referred to as “Z-score normalization”.

$$X_{changed} = \frac{X - \mu}{\sigma}$$

Regularization – Feature Scaling-Normalization

- Normalization:
  - Scales all data values to be between 0 and 1.

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Regularization – Feature Scaling

- There are many more methods of scaling features and Scikit-Learn provides easy to use classes that “fit” and “transform” feature data for scaling.
- Let’s quickly discuss the fit and transform calls in more detail when it comes to scaling.

Regularization – Feature Scaling

- A .fit() method call simply calculates the necessary statistics (Xmin,Xmax,mean, standard deviation).
- A .transform() call actually scales data and returns the new scaled version of data.
- Previously saw a similar process for polynomial feature conversion.

Regularization – Feature Scaling

- Very important consideration for fit and transform:
    - We only **fit** to training data.
    - Calculating statistical information should only come from training data.
    - Don’t want to assume prior knowledge of the test set!

Regularization – Feature Scaling

- Using the full data set would cause **data leakage**:
  - Calculating statistics from full data leads to some information of the test set leaking into the training process upon transform() conversion.

Regularization – Feature Scaling

- Feature scaling process:
    - Perform train test split
    - Fit to training feature data
    - Transform training feature data
    - Transform test feature data

Regularization – Feature Scaling

- Do we need to scale the label?
  - In general it is not necessary nor advised.
  - Normalising the output distribution is altering the definition of the target.
  - Predicting a distribution that doesn't mirror your real-world target.
  - Can negatively impact stochastic gradient descent.