Python Machine Learning In Biology:

Bias-Variance Tradeoff

Nichole Bennett

Errors

We really care about two types of prediction errors:

- Errors due to “bias”
- Errors due to “variance”

There’s a tradeoff between a model’s ability to minimize both of these.

Diagnosing bias and variance can help us figure out if our models are overfitting or underfitting.

Bias and Variance

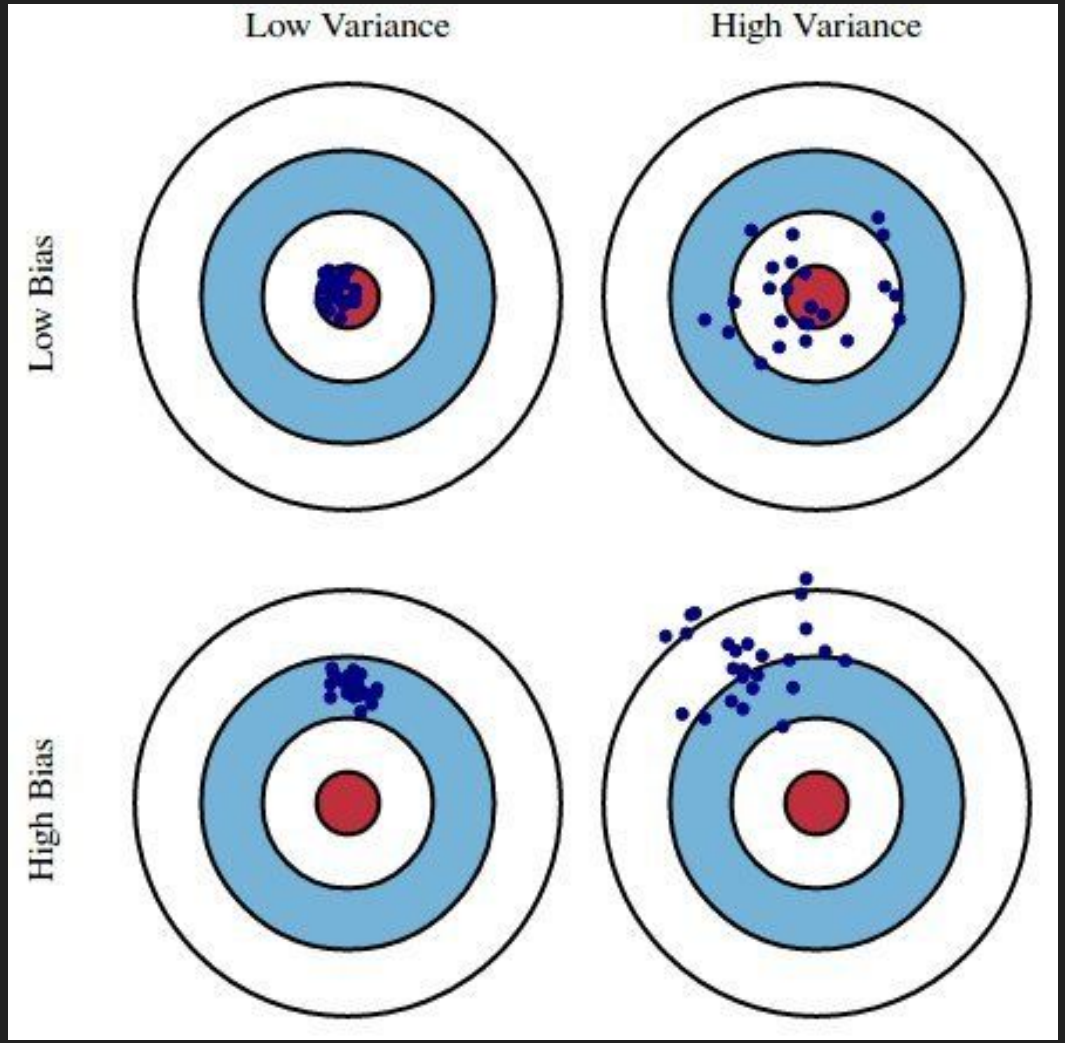
Imagine we are repeating the model building process more than once. Each time we gather new data and run a new analysis, creating a new model.

Error due to Bias: difference between the expected (or average) prediction of our model and the correct value which we are trying to predict.

$$Bias(\hat{\beta}_{OLS}) = E(\hat{\beta}_{OLS}) - \beta.$$

Error due to Variance: the variability of a model prediction for a given data point; the spread/uncertainty of these estimates

$$Var(\hat{\beta}_{OLS}) = \sigma^2 (X'X)^{-1},$$


Low Variance
High Variance
Low Bias
High Bias

Bias and Variance

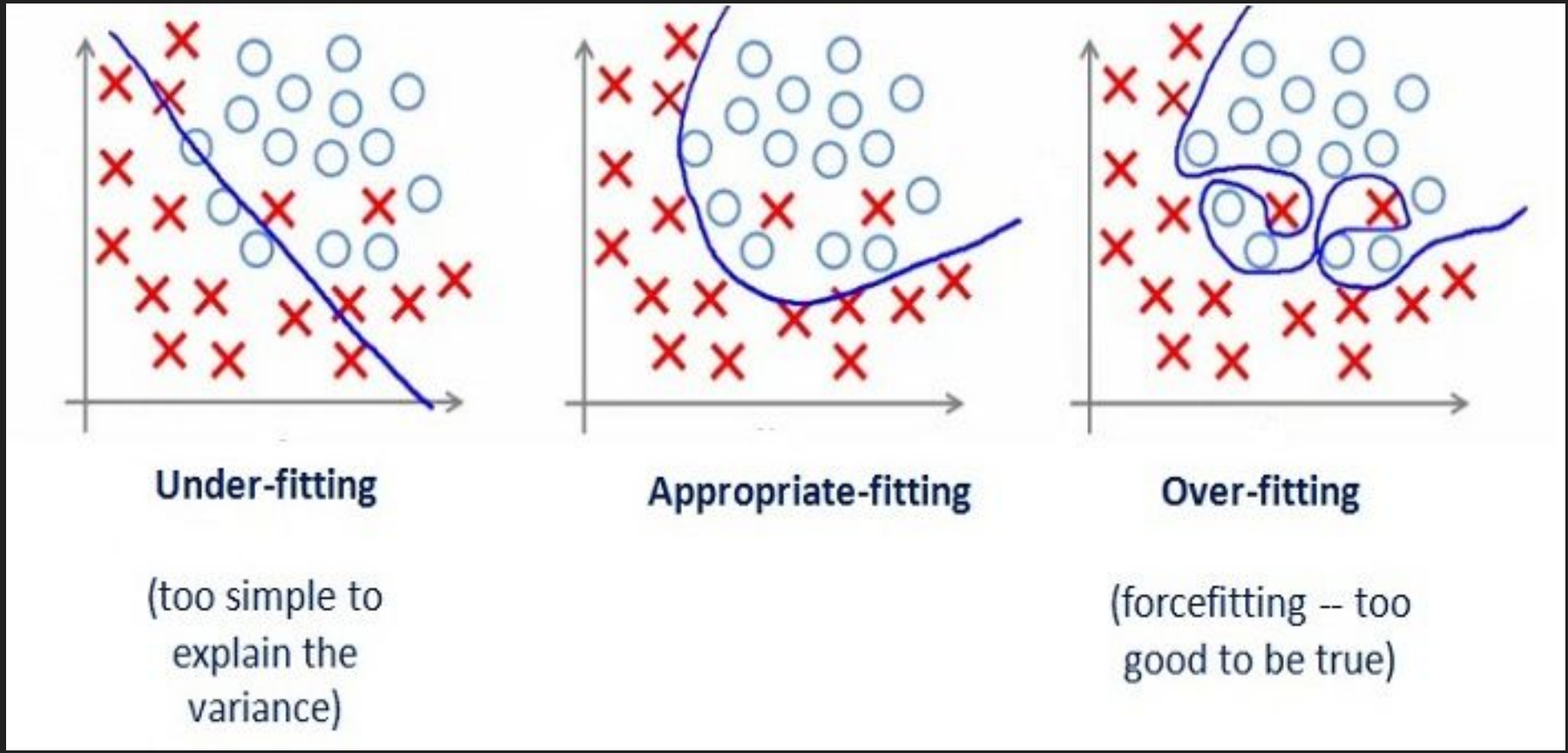
Model performs well on training data but does not generalize well to unseen data (test data)

If a models suffers from overfitting, we say it has high variance

- May be too many hyperparameters (too complex for underyling data)

If a model suffers from underfitting, we say it has high bias

- Model not complex enough to capture pattern in training data
- Low performance on unseen data (and seen data)


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

Voting Example

Let’s say we do a phone poll and try to figure out who is going to win the next election. We pick 50 numbers randomly out of the phone book and record who they say they will vote for.

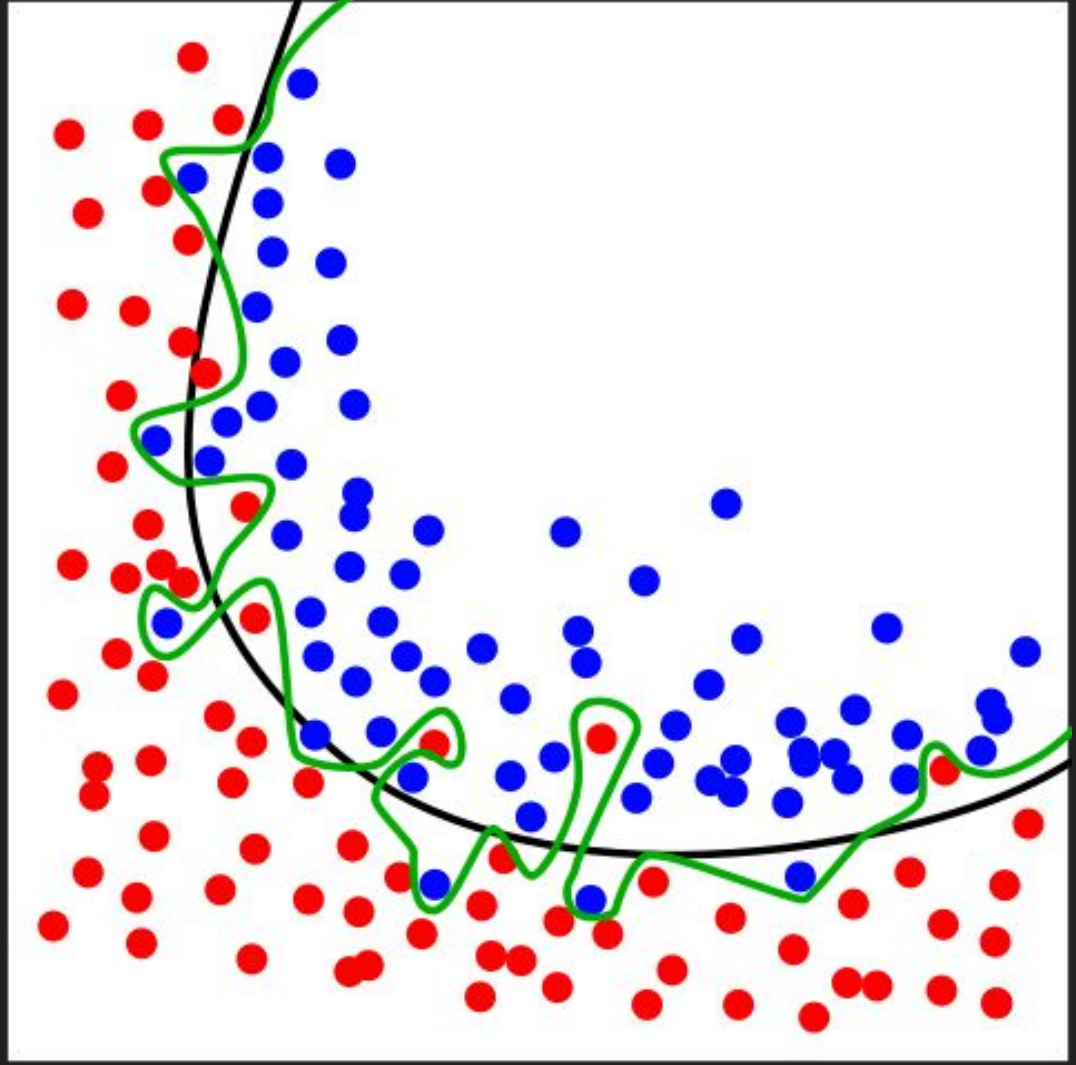
Our answers are off. We obviously have lots of sources of error, but let’s separate the sources into errors of bias and errors of variance.

Voting Example

What are some of the sources of error in our voting example?

- only sampled people from the phone book
- did not follow up with non-respondents
- have a very small sample size

Which of these are bias errors and which are variance errors?



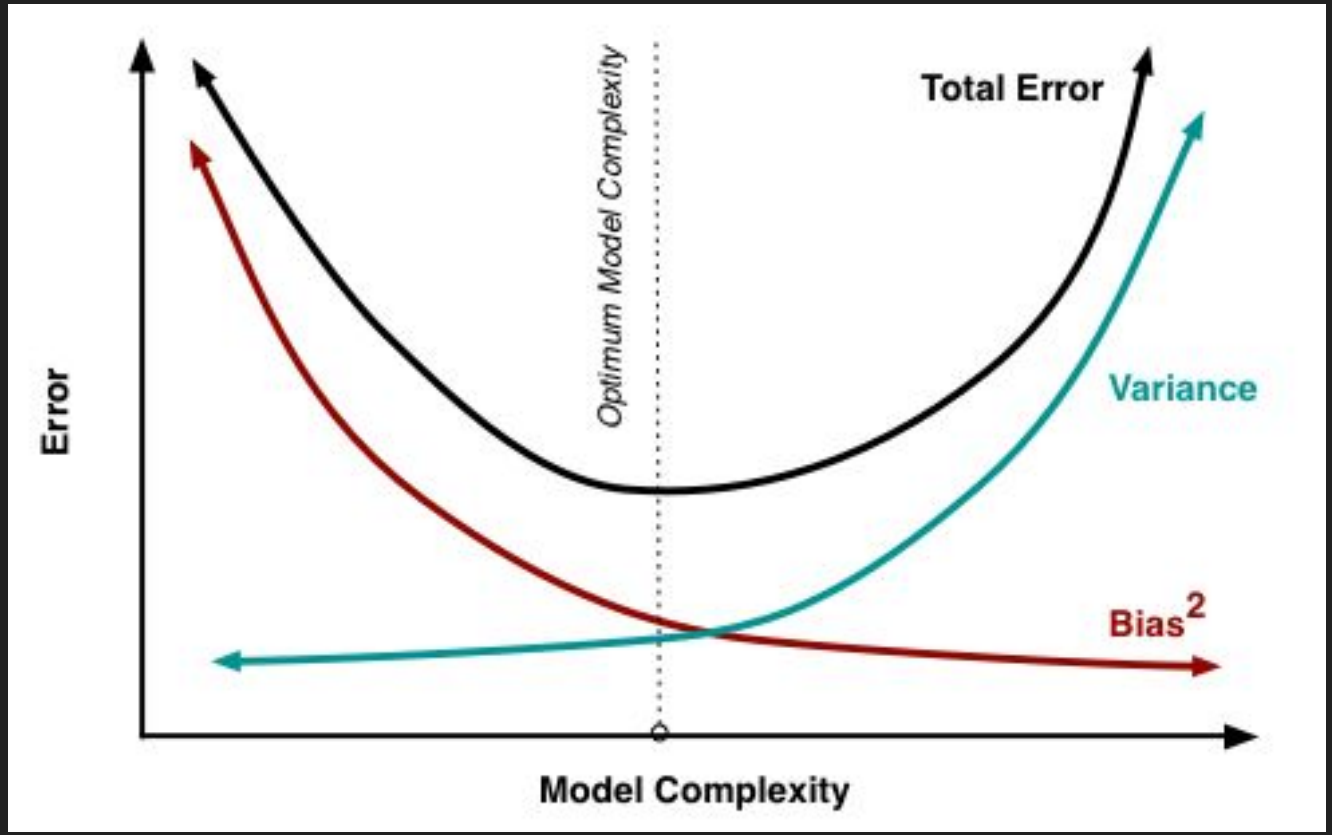
Managing Bias and Variance

Fight Your Instincts: that gut feeling that says to minimize bias at the expense of variance is wrong

> Doing well on the training set is easy (just memorize the data)--> it's generalization that counts

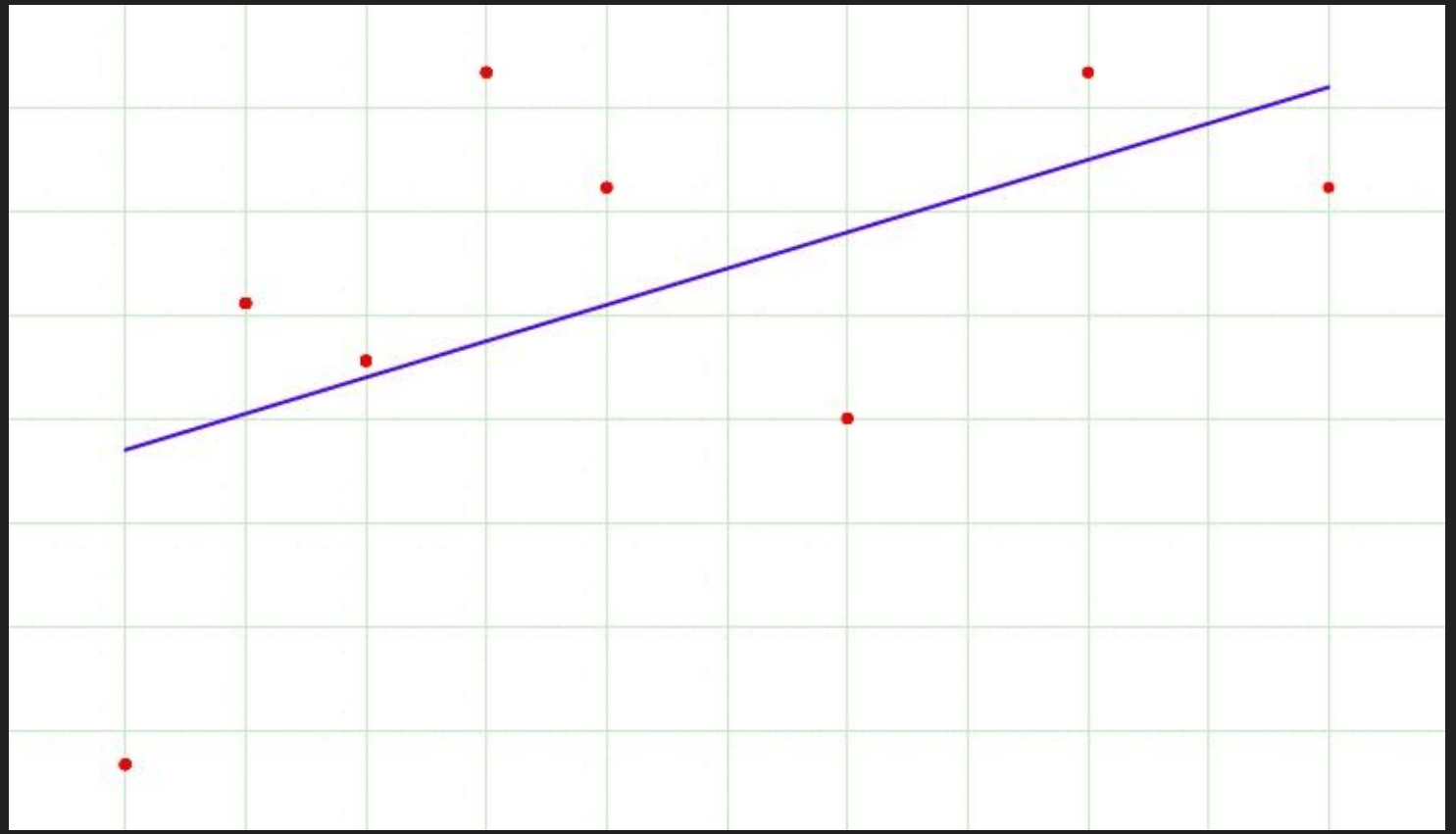
Bagging (Bootstrap Aggregating) and Resampling: use ensemble methods (e.g. Random Forests)

Bias-Variance Tradeoff



Regularization

Underfitting vs. Overfitting



Dealing with Overfitting

Cross-validation sampling

Choose a simpler model with fewer parameters

Reduce the dimensionality of the data

Pruning

Regularization

Collect more training data

Regularization

Handles collinearity (high correlation between features), filters out noise from data, and helps prevents overfitting.

Regularization adds a penalty as model complexity increases.

Regularization will add the penalty for higher terms. This will decrease the importance given to higher terms and will bring the model towards less complex equation.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

Choosing a Regulariation Parameter

more traditional approach:

choose λ such that some information criterion, e.g., AIC or BIC, is the smallest. (Emphasizes model's fit to data)

more machine learning-like approach:

perform cross-validation and select the value of λ that minimizes the cross-validated error. (Emphasizes predictive performance)

L1 vs L2 Regularization

Key difference is the penalty term

L1 = Lasso

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

If λ=0, back to original model.

If λ big, will make coefficients zero and will underfit.

Shrinks least important coefficient.

Selects most important features.

L2 = Ridge

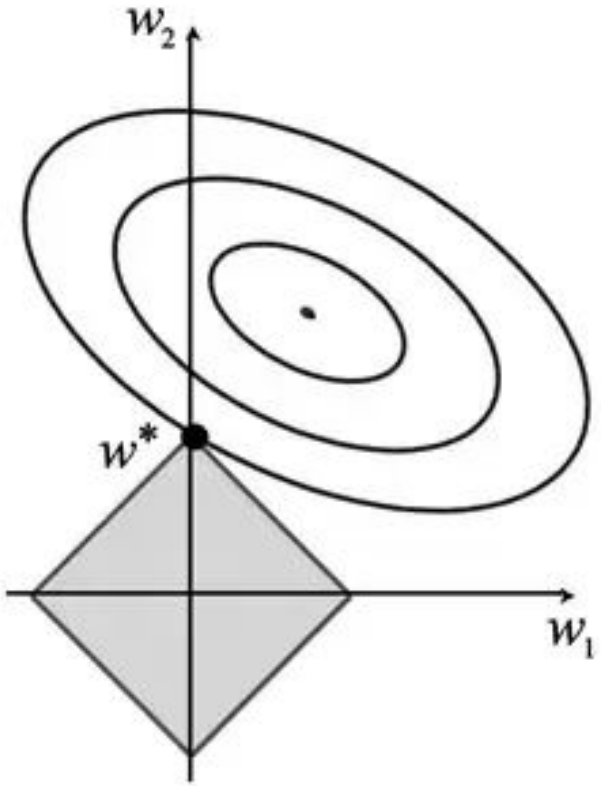
$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Adds a “squared magnitude” of coefficient as penalty term to the loss function.

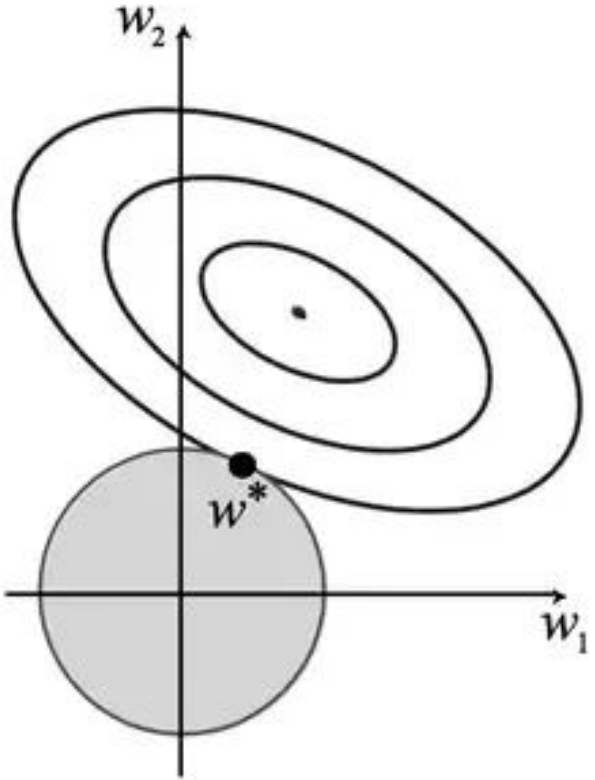
If λ=0, back to original model.

If λ big, adds too much weight and underfits.

Penalizes complexity.



L1



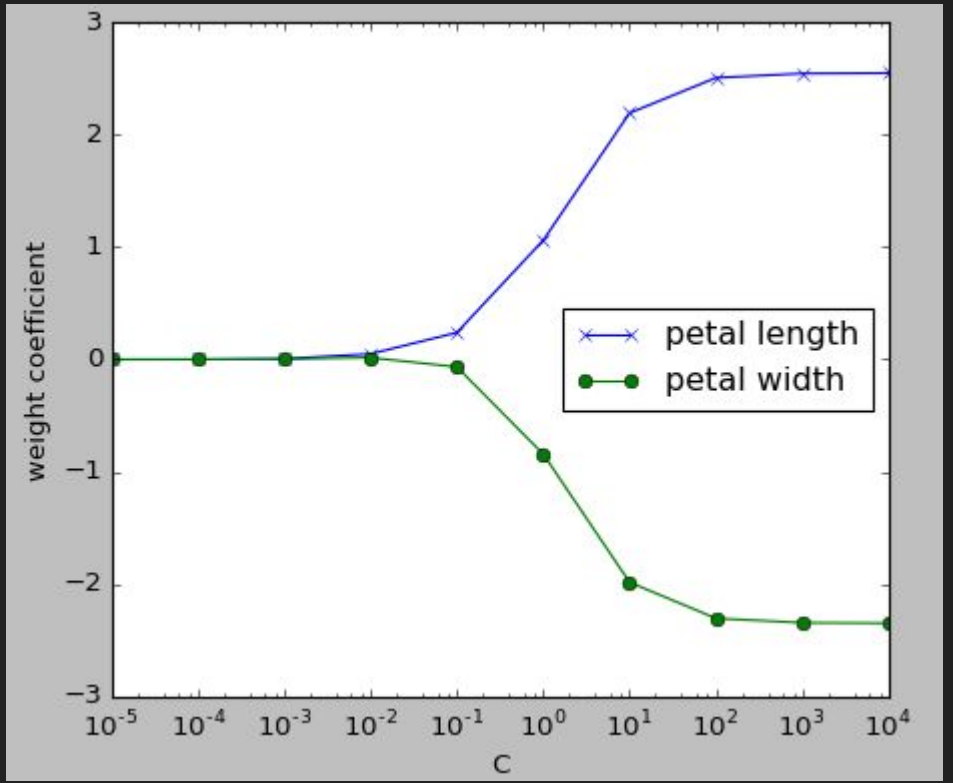
L2

L2 Regularization in Python

Increasing λ, increases regularization strength

The parameter `c` that is implemented in the `LogisticRegression` class in scikit-learn is the inverse of the regularization parameter λ.

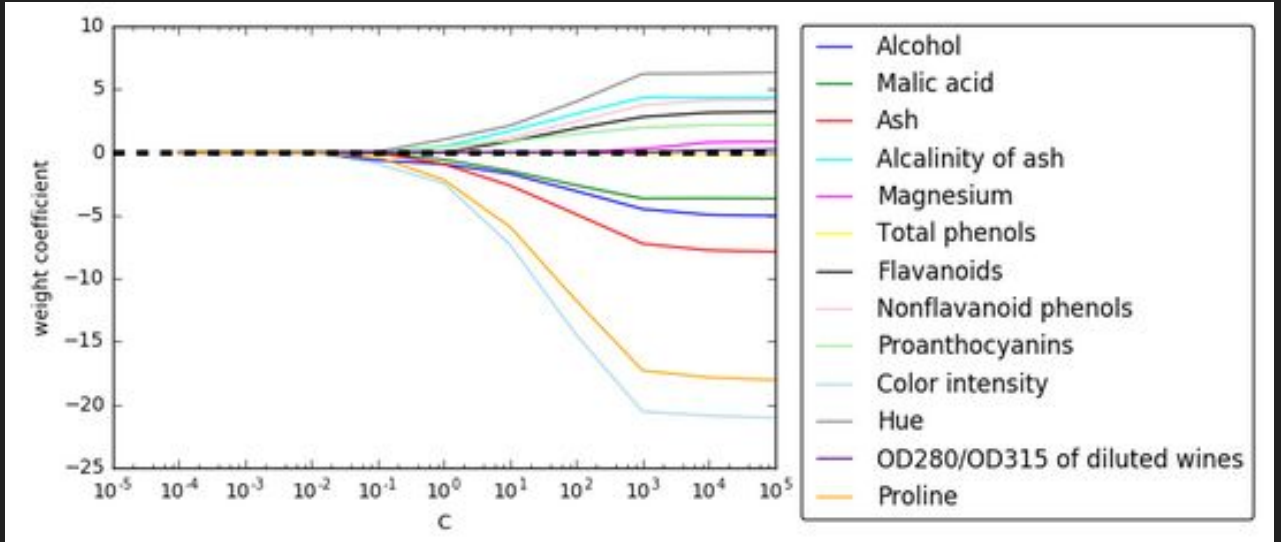
Decreasing c increases regularization strength.



L1 Regularization in Python

Set the penalty parameter to ‘l1’ to yield the sparse solution.

Increasing λ, increases regularization strength. The parameter c that is implemented in the LogisticRegression class in scikit-learn is the inverse of the regularization parameter λ. Decreasing c increases regularization strength.



When to use regularization

Other methods, like cross-validation or stepwise models handle overfitting and perform feature selection work well with a small set of features but these techniques are a great alternative when we are dealing with a large set of features.

Can’t use regularization on KNN and Decision Trees. (Have to rely on feature engineering and dimensionality reduction)

Best for when you have small amounts of training data compared to your amount of features. Becomes less important as amount of training data you have increases.

When to use Lasso (L1) vs. Ridge (L2)

- Often neither one is overall better.
- Lasso can set some coefficients to zero, thus performing variable selection, while ridge regression cannot.
- Both methods allow to use correlated predictors, but they solve multicollinearity issue differently:
  - In ridge regression, the coefficients of correlated predictors are similar;
  - In lasso, one of the correlated predictors has a larger coefficient, while the rest are (nearly) zeroed.

When to use Lasso (L1) vs. Ridge (L2)

- Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (when only a few predictors actually influence the response).
- Ridge works well if there are many large parameters of about the same value (when most predictors impact the response).

- But, practically speaking, we don’t know the true parameter values, so this is pretty theoretical

- Run cross-validation to select the more suited model for a specific case.
- Or... combine the two! Elastic Net!

Elastic Net

First emerged as a result of critique on Lasso, whose variable selection can be too dependent on data and thus unstable.

The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds. Elastic Net aims at minimizing the following loss function:

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i-1}^{n}(y_i - x_i'\hat{\beta})^2}{2n} + \lambda(\frac{1-\alpha}{2}\sum_{j-1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j-1}^{m}|\hat{\beta}_j|),$$

where α is the mixing parameter between Ridge (α = 0) and Lasso (α = 1).

Now, there are two parameters to tune: λ and α.