1 Module 7: Linear & Nonlinear Regression

Contents

1.1 Regression

- Only linear things can have linear regressions:
  e.g. predicting on LABELS and not continuous data won't work (incl. to some degree, "Probability" of a particular label)

- Given some features θ and a prediction output $\hat{y}$

$$\hat{y} = \omega_1 \times \theta_1 + \omega_2 \times \theta_2 + ...$$

- LOWESS or LOESS method of fitting regressions will work for nonlinear cases.

- Simple linear:
  obj = sklearn.linear_model.LinearRegression(fit_intercept = bool) where fit_intercept = False will set the intercept to be 0
  obj.fit(features:DataFrame, target:Series)
  Use obj.predict([int]:list)
  f.coef_ for slope, f.intercept_

- Using plotly:
  fig = px.scatter(data:DataFrame, x:str, y:str, trendline='ols') for ols = ordinary least squares
  results = px.get_trendline_results(fig)
  results.px_fit_results.iloc[0].params

1.2 Loss Functions

- Numerically characterizes error in a prediction (more loss = worse)

- Most common fcn. is squared error L2:

$$L(y, \hat{y}) = (y - \hat{y})^2 \text{ for data y and prediction } \hat{y}$$

- Mean squared error (MSE) is just the average of the loss function of all datapoints

$$L(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i) \text{ for dataset } \mathcal{D}$$

  sklearn.metrics.mean_squared_error(y, $\hat{y}$)

- Minimize the loss fcn. to optimize model params
  scipy.optimize.minimize(lambda θ: mean_squared_error(y, $\hat{y}(\theta)$), x0:float)
  for x0 the starting point

- Absolute loss (L1):

$$L(y, \hat{y}) = [y - \hat{y}]$$

- L2 MSE is more affected by (gives higher penalty to) outliers than L1 MAE

- MAE is also piecewise linear whereas MSE is continuous; the former might require really fine sampling

- Huber loss (smooth mean absolute error): composite of MSE and MAE with an optimizeable term ϵ which allows handles outliers
  If error is $< \epsilon$, then MSE is used. Otherwise, MAE.

- Mean squared logarithmic error (MSLE): treat differences in the larger domain equally to the small domain (but penalizes an underpredicted estimate more than an overpredicted estimate)

- Mean bias error (MBE): calculates model's average bias by taking difference between actual difference and absolute difference of target/predicted values (unreliable because positive errors tend to cancel out negative ones)

1.3 Multiple Linear Regression

- Performed identically as before, whereas the features are a DataFrame instead of a Series
  obj.fit(features:DataFrame, target:Series)
  $\rightarrow$ multiple returned coefficients/intercepts

1.4 Non-numeric features (labelled data)

- Categorical data: split into Ordinal and Nominal
  - Nominal (labelled or named) data is used to name variables with no inherent numerical values or ordering (e.g. sex)
  - Ordinal data has clear ordering (e.g. customer satisfaction)
- One-hot coding with K features corresponding to the unique labelled values (i.e. Sex has 0,1)
- To create the labels, use
  pd.get_dummies(Series)
- pd.concat([DataFrame, DataFrame], axis=1)