Artificial Intelligence

Linear Regression

Linear Regression ?

- A form of supervised learning
  - It learns from labeled training data
    - The labels are numerical targets, not categorical features
- Statistical method for predictive modelling
- Goal: to model a linear relationship between the target values (dependent var) and independent var(s), representing the feature(s) AKA "the observed data"
- Simple Linear Regression (SLR)
  - There is only 1 independent var, 1 feature
  - Formulas for "slope" and "bias"/"intercept"
- Multiple Linear Regression (MLR)
  - There is >=2 independent vars, >=2 features
  - Matrix ops for "slopes"/"coefficients"

Linear Regression ?

- Concepts
  - Dependent var (target): what we're trying to predict or explain
  - Independent var(s): the inputs, the features
  - Linear expression (of a line, plane, ... , hyperplane)
    - y = mx + b [1 feature]
    - y = b0 + m1*x1 + m2*x2 [2 features]
    - ...
    - y = b0 + b1*x1 + ... bn * xn [n features]
      - b0 is the y-intercept
      - b1 .. bn are the "slopes"/"coefficients"
  - Coefficients represent the change in the dependent variable (y) for a one-unit change in the respective independent variable, holding all other variables constant

SLR formulas

- SLR slope m:
  - x = observed values of the independent var
  - y = observed values of the dependent var
  - n = number of samples or observations
  - numerator = n*sum(x*y) - sum(x)*sum(y)
  - denominator = n*sum(x^2) - (sum(x)^2)
  - m = numerator / denominator
- SLR intercept b:
  - numerator = sum(y) - m*sum(x)
  - denominator = n

MLR formulas

- Algebra for all the coefficients b
  - b = (XT * X)^-1 * XT*y
    - XT : the transpose of T
    - (XT*X)^-1 : the inverse of the matrix product of XT with X
    - y : the vector of target values
  - X: m*(n+1) "design matrix"
    - m the number of observations
    - n the number of features
    - +1 for a leftmost column of 1s for the intercept term
    - Each row of X is a sample, 1st column (leftmost) with 1 (intercept)
  - b is the vector of coefficients
- With sklearn.linear_model, just do:

```
model = LinearRegression()
model.fit(X,y)
model.intercept_ # this is y-intercept
model.coef_[0] ... model.coef_[n-1] # the n coefficients
```

How good is the LR model ? R-squared

- SST = sum of squares total =
  - sum((y - ymean)^2)
- SSR = sum of squares regression =
  - sum((ypred - ymean)^2)
- SSE = sum of squares error (or "residual sum of squares")
  - Sum((y-ypred)^2)
- R-squared = SSR/SST
  - Focus on how much of the total variance is captured by the model
- R-squared = 1 - SSE/SST
  - Focus how much variance is not captured by the model (i.e., the errors)
  - 1 - SSE/SST is, again, the total variance captured by the model
- Formulas are equivalent because SST = SSR+SSE

R-squared - meaning

- R-squared AKA "coefficient of determination"
  - Value in [0,1]
  - 1 = "perfect fit"
  - 0 = "no fit"
  - "high value": large proportion of the variance in the dependent variable has been captured but the model
    - Very high suggests "overfitting"
  - "low value": the model does not capture much of the variance in the dependent variable; maybe the assumption that there is linear relationship between the vars is wrong!
  - Represents the proportion of the variance for a dependent variable that is captured/explained by the independent var(s) in a regression model
  - How well do predictions approximate real data points?

R-squared - context

- In fields where data is more unpredictable, such as social sciences, lower values of R-squared might be acceptable
- Not a measure of accuracy: high values do not guarantee "unbiased" predictions
- If there are plural features, "Adjusted R-squared" is better
  - Because adding more features/predictors to a model can inflate the R-squared value, even if they don't actually improve the model
    - R-squared does NOT change with scale-ops

R-squared adjusted (RSA)

- Modified version of R-squared, adjusted for the number of features in the model
- Introduces a penalty for adding more predictors
- +features +penalty
- RSA will only increase when a new feature improves the model
- RSA = 1 - (1-R-squared)*(n-1/n-p-1)
  - n : the number of samples
  - p : the number of features

Feature Scaling

- Normalization
- Standardization
- Techniques to standardize the range of features
- Techniques that adjust the scale of the features, but DO NOT change the relationship between them
- So, in the context of LR, feature scaling does not affect R-squared measurement
- But can be important:
  - Interpretation and comparison
    - easier to interpret which variables have a more significant effect on the outcome
  - Facilitated convergence in some algorithms that use LR

References

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- https://github.com/amsm/am_lr_mlr