Regression

(1IK172 Introduction to Data Analytics)

Correlation vs. Regression

- A scatter plot can be used to show the relationship between two variables
- Correlation analysis is used to measure the strength of the association (linear relationship) between two variables
    - Correlation is only concerned with strength of the relationship
    - No causal effect is implied with correlation

Prediction

Label

- A possible outcome of an event
- Binary
  - person can be "child" or "adult"
- Nominal
  - car can be "family", "sport", "terrain" or "truck"
- Ordinal
  - movies can be rated "worst", "bad", "neutral", "good" and "excellent"
- Quantitative
  - houses have prices

Predictive task

- Goal: build a predictive model, from the labeled (train) instances in the data, which maps a vector of predictive attribute values to labels
- In order to assign the correct labels for the unlabeled (test) instances in the data

Regression task

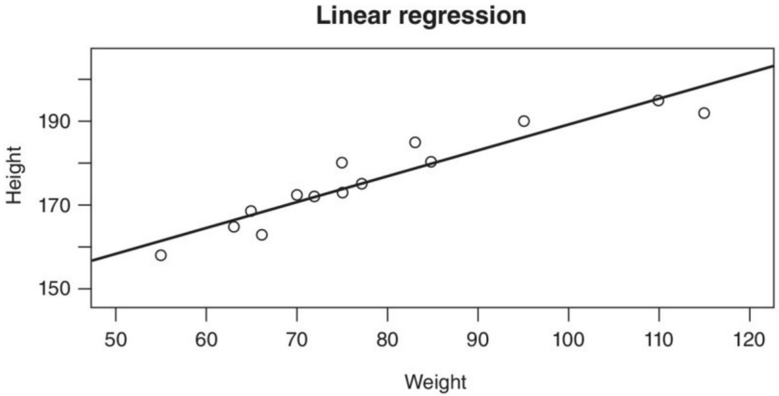
- Labels are quantitative

Classification task

- Labels are binary, nominal or ordinal

Example

Name	Weight	Height
Andrew	77	175
Bernhard	110	195
Carolina	70	172
Dennis	85	180
Eve	65	168
Fred	75	173
Gwyneth	75	180
Hayden	63	165
Irene	55	158
James	66	163
Levin	95	190
Lea	72	172
Mary	83	185
Nigel	115	192



$$\text{Height} = \underbrace{128.017}_{\hat{\beta}_0} + \underbrace{0.611}_{\hat{\beta}_1} \times \text{Weight}$$

prediction of the height for a person with 90 kg weight would be equal to 183.007 = 128.017 + 0.611 x 90

Regression

Regression methods are used in many different domains:

- **Stock market:** to predict the value of a share in one week’s time
- **Transport:** travel-time prediction for a given path
- **Higher education:** to predict how many students a given course will have next year
- **Survival analysis:** to predict how long a person will live after a given treatment
- **Macroeconomics:** to predict the expected unemployment level given a set of proposed policies.

Simple Linear Regression Model

- Only one independent variable, X
- Relationship between X and Y is described by a linear function
- Changes in Y are assumed to be related to changes in X

Dependent Variable

Population Y intercept

Population Slope Coefficient

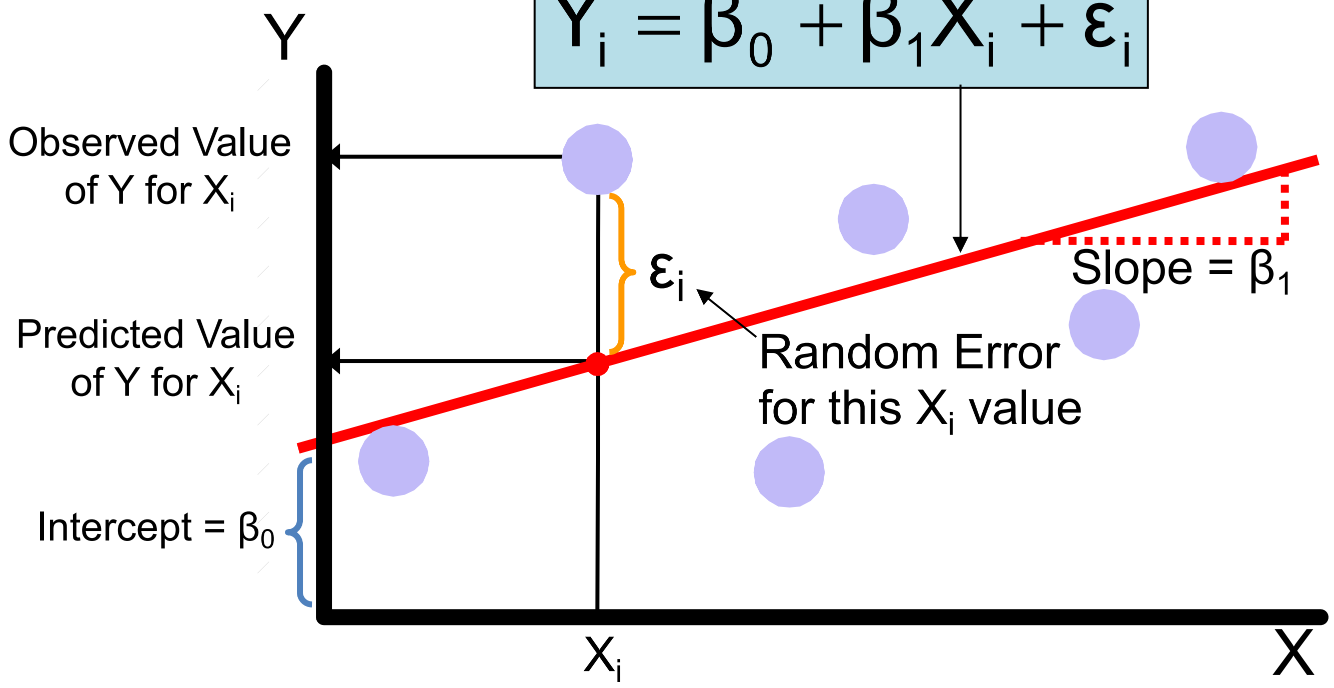
Independent Variable

Random Error term

$$\boxed{Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}}$$

Simple Linear Regression Model

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$



Simple Linear Regression Equation (Prediction Line)

The simple linear regression equation provides an estimate of the population regression line

Estimated (or predicted) Y value for observation i

Estimate of the regression intercept

Estimate of the regression slope

Value of X for observation i

$$\boxed{\hat{Y}_i = b_0 + b_1 X_i}$$

Generalization

- We want to induce a model able to correctly predict new objects of the same task
- We want to minimize the number or extent of future mispredictions

- However, we cannot predict the future
- What we can do is to estimate the predictive performance of the model for new data

We separate the training data set into two mutually exclusive parts

- **Train set**: Model parameter tuning
- **Test set**: Evaluating the induced model on new data for which the labels are known

Two important issues are:

- How to do the train-test splitting?
- What metric to use for evaluation on the test set?

Predictive performance measures

Model

- Height = 128.017 + 0.611 x Weight

Test instances			Predicted Height ($\hat{y}$)
Name	**Weight**	**Height (y)**	
Omar	91	176	183.618
Patricia	58	168	163.455

$MAE = \frac{1}{2} \times \left(|176 - 183.618| + |168 - 163.455|\right) = 6.082$;

$MSE = \frac{1}{2} \times \left((176 - 183.618)^2 + (168 - 163.455)^2\right) = 39.345$;

$RMSE = \sqrt{MSE} = \sqrt{39.345} = 6.273$;

$\bar{y} = \frac{175+195+172+180+168+173+180+165+158+163+190+172+185+192}{14} = 176.286$

$RelMSE = \frac{(176-183.618)^2+(168-163.455)^2}{(176-176.286)^2+(168-176.286)^2} = 1.145$;

$CV = \frac{6.273}{176.286} = 0.036$

Mean absolute error

$$MAE = \frac{1}{n} \times \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Mean squared error

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Root mean squared error

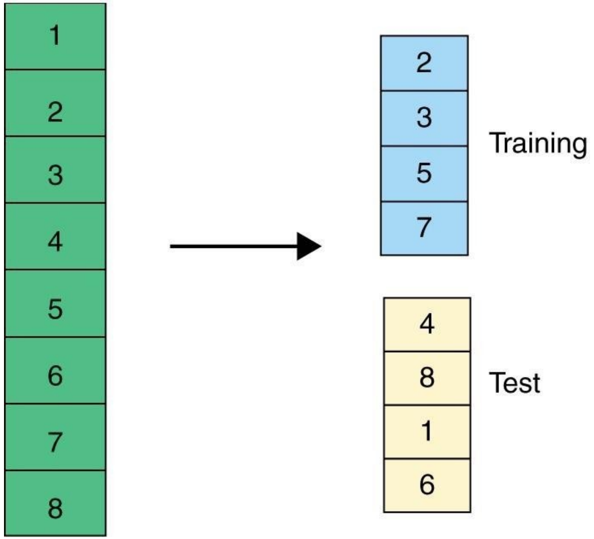
$$RMSE = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Relative mean squared error

$$RelMSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Model validation

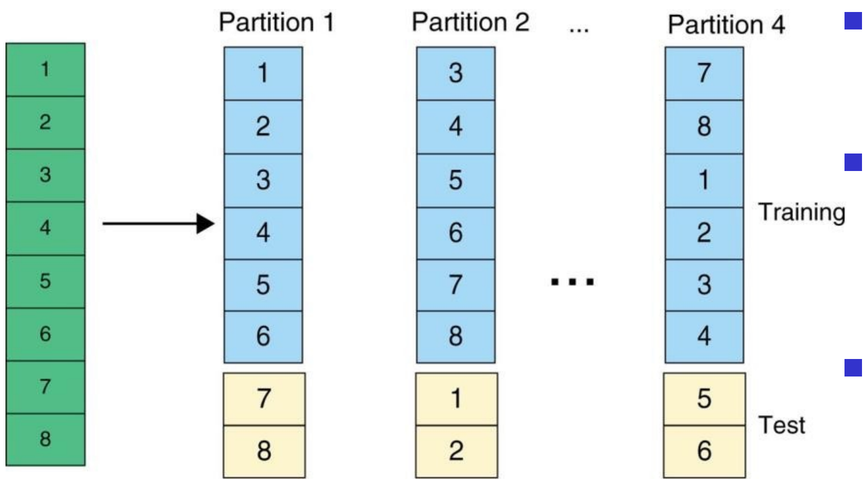
Holdout validation



- Split the data once and train once on the train set, evaluate on the test set
- Split ratio is arbitrary
- Very easy to understand and implement
- Can yield misleading results due to imbalance

Model validation

k-fold cross validation

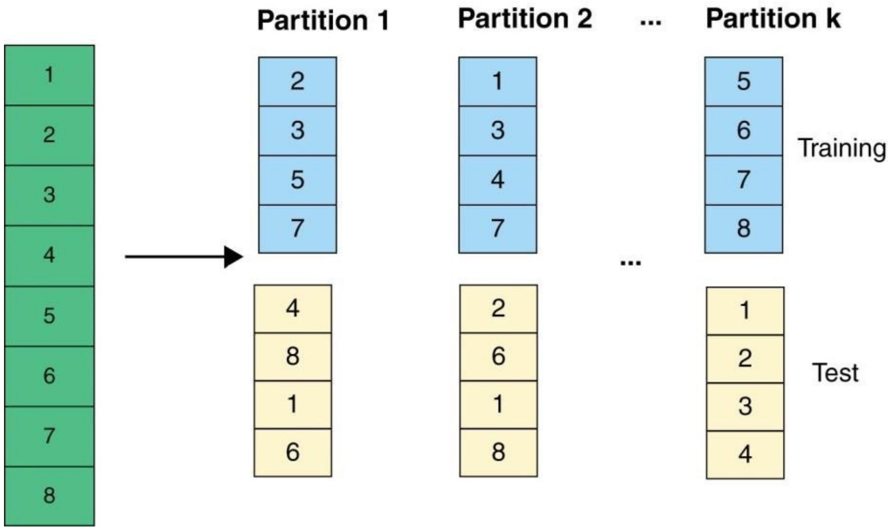


- Split the data *k* times into *k* equal sets and train the model k times
- At each training use 1 set for evaluation the rest for training
- After the *k* training is done the final evaluation metric is given by taking the average of the k training sessions
- Computationally expensive but gives a better idea about how well does the model work

Model validation

- Create multiple random split of the data and fit the model on each random split
- Final evaluation metric equals the average metric score
- Split ratio is not dependent on the iteration count
- Possible that some rows are never evaluated

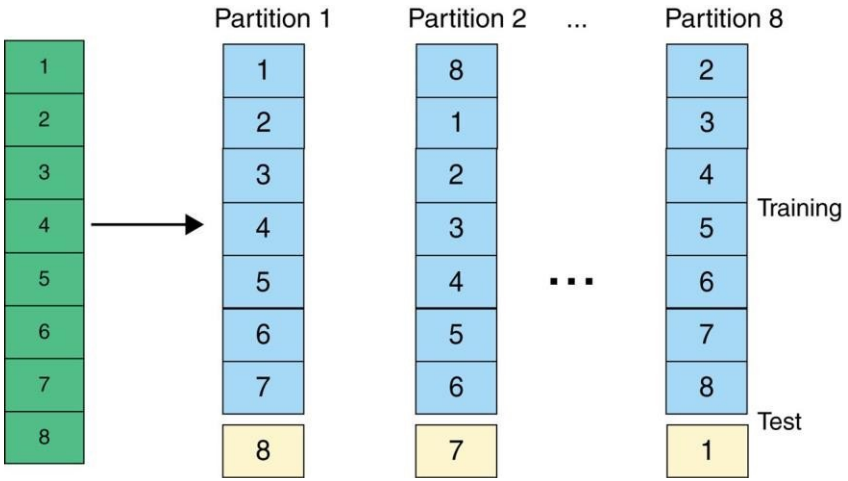
Random sub-sampling



Model validation

- Special case of k-fold cross validation where k equals the number of samples
- Computationally very expensive

Leave-one-out



Model training

- Optimization problem of finding the right parameters for our model so that the error is minimized
- The function calculating the error is called **cost function**
- The calculated error is called **loss**
- Model: $\hat{\mathbf{y}} = \mathbf{w} * \mathbf{x}$, where $\hat{y}$ is the predicted target variable, **w** is a vector of model parameters and **x** is a vector containing predictor attributes
- There are many optimization methods but most of them derive from Gradient Descent

Gradient Descent

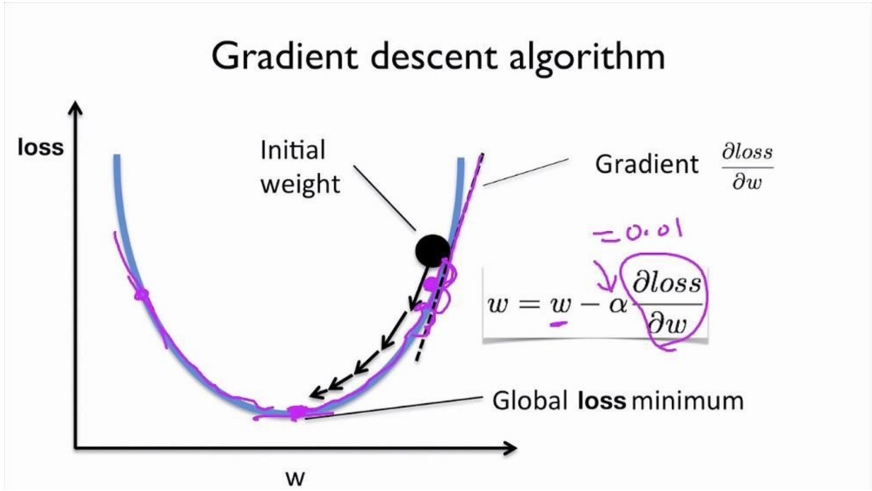
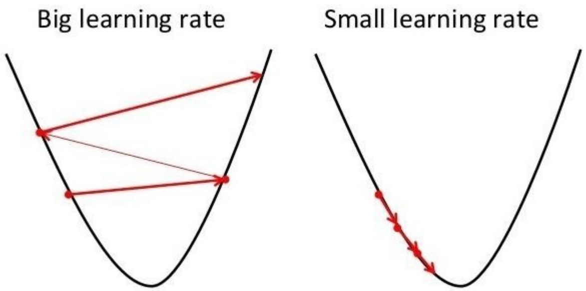


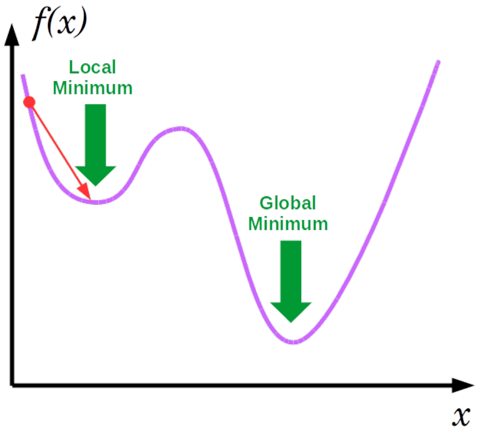
Image source: https://mc.ai/gradient-descent-and-its-types/

Gradient Descent

- Calculates the slope of the error using the derivative and adjusts parameter values based on the direction of the closest minima
- The size of the step we take towards the minima is defined by learning rate α
- Algorithm:
  1. Randomize **w** parameter vector
  2. **w** = **w** – αs
     where s is the slope and α is the learning rate
  3. Repeat 2. until the error rate decreases

Gradient Descent





- Learning rate is key
  - Too small and it converges slowly and higher chance of stuck at local minima
  - Too big and it can diverge
- The problem of constant learning rate is solved in more advanced derivatives of Gradient Descent
- Variations differ based on parameter update time
  - Stochastic Gradient Descent: update after each training sample
  - Batch Gradient Descent: update after an epoch (a run through all the samples)
  - Mini-batch Gradient Descent: update happens after batches of samples

Linear regression

Pros

- Good interpretability
- No hyperparameters
- Strong mathematical foundations

Cons

- Poor fit if relationship between predictive attributes and target is non-linear
- Number of instances must be greater than number of attributes
- Sensitive to correlated predictive attributes
- Sensitive to outliers

Noise in labels

Suppose that there is a hypothetical function f expressing the relationship between instances $\mathbf{x}_i$ and their labels y_i such that

- $y_i = f(\mathbf{x}_i) + noise_i$
- $noise_i$ is the difference from the hypothetical value $f(\mathbf{x}_i)$ and the measured value y_i
- assume noise is normally distributed with zero mean
  - there are cases when y_i are slightly above $f(\mathbf{x}_i)$ and cases when y_i are slightly above $f(\mathbf{x}_i)$ but the average noise should be zero
- **Noise** is also called **Irreducable Error**

Since f is unknown, we want to induce a model which is as close to f as possible

- And not only for the training instances but also for new unknown instances
- i.e. we want the model to be quite general
- But also precise
  - give good predictions for yet unseen instances
  - with a low bias
- And robust
  - parameters do not vary much if optimized on slightly different sample of training data
  - with low variance

Bias and variance

Bias

- The difference between the model's expected value and the true value of the attribute being estimated.

Variance

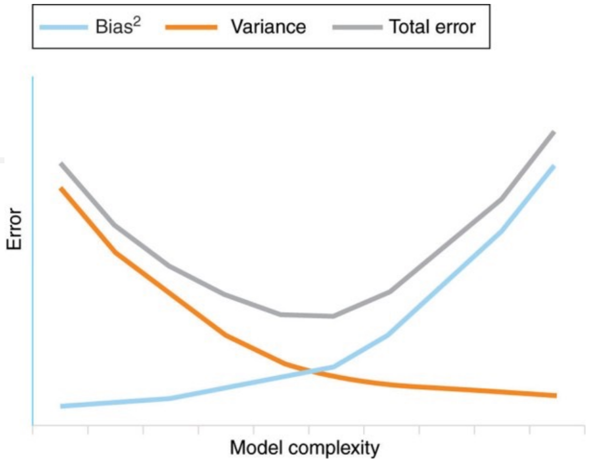
- Expresses the estimated variance of various instances of a given type of model, trained on various training data, respectively.

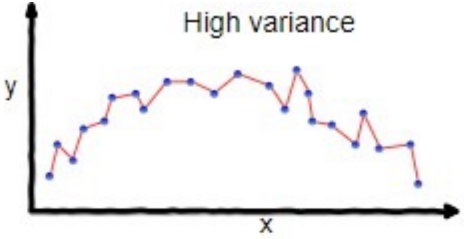
$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

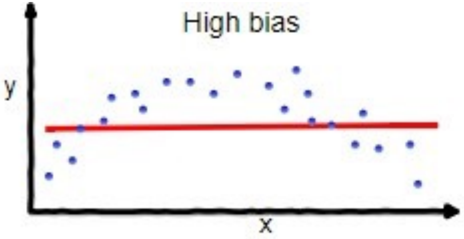
Bias-variance trade-off

- Low bias implies high variance and vice-versa
- We would like to find a model with a good trade-off
  - not too complex but with good predictive power

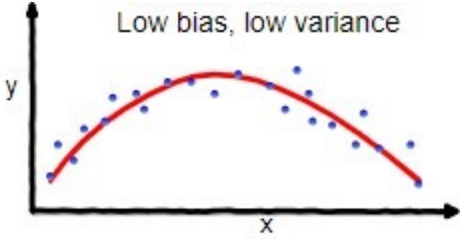




overfitting



underfitting



Good balance

Using linear combinations of attributes

Principal Components Regression (PCR)

- PCR defines the principal components without evaluating how correlated the principal components generated are with the target attribute.
- The principal components are used as predictive attributes in the formulation of the multivariate linear regression problem

Final remarks

- It is important to choose a model with good bias-variance trade-off.
- Linear regression models are popular due to their easy interpretation, computational cost as well as fair predictive performance.
- Shrinkage methods are used to increase the generalization power of the models.
- Using linear combination of attributes or their non-linear transformations can extend the usability of linear models to non-linear cases, too.

Thank You!