Machine, Data and Learning

Selected slides for lectures on ML Topic

Generalization & Goodness of Fit

- Based on Chapter 1 of Python Machine Learning by Example by Yuxi Liu

- **Generalization** refers to how well the concepts learned by a ML model generalizes to specific examples or data not yet seen by the model.
  - …
- **Goodness of fit** describes how well a model fits for a set of observations.
  - Overfitting and Underfitting

Overfitting

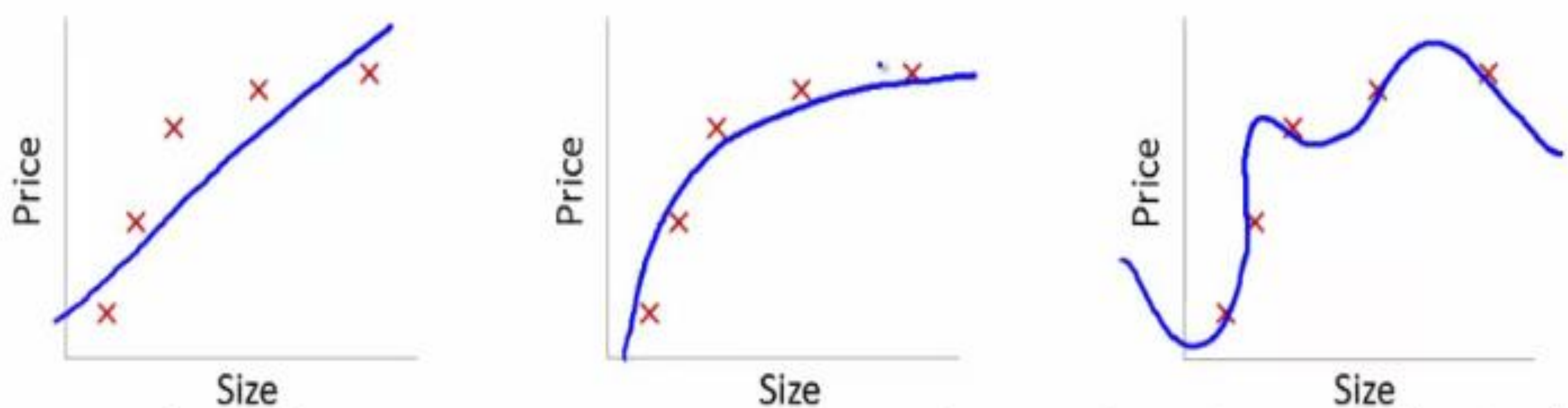
- Phenomenon of extracting too much information from training sets or memorization can cause overfitting
  - Makes ML model work well with training data called **low bias**
  - Bias refers to error due to incorrect assumptions in learning algorithm
  - However, does not generalize well or derive patterns, performs poorly on test datasets called **high variance**
  - Variance measures error due to small fluctuations in training set

Underfitting

- Model is underfit if it does not perform well on training sets and will not do so on test sets
- Occurs when we are not using enough data to train or if we try to fit wrong model to the data
  - E.g., if you do not read enough material for exam or if you prepare wrong syllabus
- Called **high bias** in ML although **variance is low** [i.e. consistent but in a bad way]
- May need to increase number of features since it expands the hypothesis space.

Goodness of fit

- For same data:



High bias (underfit)	"Just right"	High variance (overfit)
$\theta_0 + \theta_1 x$	$\theta_0 + \theta_1 x + \theta_2 x^2$	$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Bias-Variance Tradeoff

- If the model is too simple and has very few parameters then it may have high bias and low variance
- If the model has large number of parameters it may have high variance and low bias
- We need to find a right/good **balance** without overfitting or underfitting the data
- As more parameters are added to a model
  - Complexity of the model rises
  - Variance becomes primary concern while bias falls steadily.

Bias-Variance Tradeoff

- Suppose a training set consists of points x1, …, xn and real values yi associated with each point xi
- We assume there is a function y = f(x) + Ɛ, where the noise Ɛ has zero mean and variance σ^2
- Find $\hat{f}(x)$, tht approximates f(x) as well as possible
- To measure how well the approximation was performed, we minimize the mean square error $(y - \hat{f}(x))^2$
- A number of algorithms exist to find $\hat{f}(x)$, that generalizes to points outside of our training set

Bias-Variance Tradeoff

- Variance measures how far a set of (random) numbers are spread out from their average value.
- Measured as expectation of the squared deviation of a random variable from its mean.

$$Var(X) = E[(x-\mu)^2]$$

$$Var(X) = E[(x-E[x])^2]$$

$$= E[x^2 - 2xE[x] + E[x]^2)$$

$$= E[x^2] - 2E[x]E[x] + E[x]^2$$

$$= E[x^2] - E[x]^2$$

Bias-Variance Tradeoff

- Turns out expected (mean squared) error of $\hat{f}$ on an unseen sample in general can be decomposed as:

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = (Bias[\hat{f}(x)])^2 + Var[\hat{f}(x)] + \sigma^2$$

where,

$$Bias\left(\hat{f}(x)\right) = E[\hat{f}(x) - f(x)]$$

$$= E[\hat{f}(x)] - E[f(x)] = E[\hat{f}(x)] - f(x)$$

$$Since\ f\ is\ deterministic, E[f] = f$$

and

$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$$

Note that all three terms are positive

Bias-Variance Tradeoff

- Notations:

$$Var[x] = E[x^2] - (E[x])^2$$

$$E[X^2] = Var(X) + (E[x])^2$$

$$Given\ y = f + \varepsilon\ and\ E[\varepsilon] = 0, E[y] = E[f + \varepsilon] = E[f] = f$$

$$Since\ Var[\varepsilon] = \sigma^2, Var[y] = E[(y - E[y])^2] = E[(y - f)^2]$$

$$= E[(f + \varepsilon - f)^2] = E[\varepsilon^2] = Var[\varepsilon] + (E[\varepsilon])^2 = \sigma^2$$

Bias-Variance Tradeoff

- The expected error on an unseen sample x can be decomposed as:

$$E[(y-\hat{f})^2 = E[(f + \varepsilon - \hat{f})^2]$$

$$= E[(f + \varepsilon - \hat{f} + E[\hat{f}] - E[\hat{f}])^2]$$

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

$$+ 2E[(f - E[\hat{f}])\varepsilon] + 2E[\varepsilon(E(\hat{f}) - \hat{f})] + 2E[(E(\hat{f}) - \hat{f})(f - E[\hat{f}])]$$

$$= (f - E(\hat{f}))^2 + E(\varepsilon^2) + E[(E[\hat{f}] - \hat{f})^2]$$

$$+2(f - E[\hat{f}])E(\varepsilon) + 2E(\varepsilon)E(E[\hat{f}] - \hat{f}] + 2E[E[\hat{f}] - \hat{f}](f - E[\hat{f}])$$

Bias-Variance Tradeoff

$$= (f - E[\hat{f}])^2 + E[\varepsilon^2] + E[(E[\hat{f}] - \hat{f})^2]$$

$$= (f - E[\hat{f}])^2 + Var[y] + Var[\hat{f}]$$

$$= Bias[\hat{f}]^2 + Var[y] + Var[\hat{f}]$$

$$= Bias[\hat{f}]^2 + \sigma^2 + Var[\hat{f}]$$

- Hence the derivation.

Avoiding Overfitting

- A variety of techniques to avoid overfitting:
  - Cross-validation
  - Regularization
  - Feature selection
  - Dimensionality reduction

Non-exhaustive Cross-validation

Exhaustive Cross-validation

Nested Cross-validation

- Popular way to tune parameters of an algorithm
- One version: k-fold cross validation with validation and test set
- Lets say parameter X needs tuning
  - Possible values 10, 20, 30, 40, 50

Nested Cross-validation

- k = 7 in our example
  - One set each picked as Test and Validation, (k-2) picked for training
- For the picked Test set
  - Perform k-fold cross validation on Train & Validation set [Here k = 6]
  - Compute the average training error for each value of X
  - Pick the best X
- Repeat for each possible Test set [i.e. 7 times]
- Pick X that was returned maximum times to outer loop

Regularization

Regularization

- Let $\hat{f}(x) = \theta_0 + \ \theta_1 x^2 + \theta_3 x^2 + \theta_4 x^3$
- We want to minimize the MSE:

$$\frac{1}{m} * \min_{\theta_0,\theta_1,\theta_3,\theta_4} \sum_{i=1}^{m} (\hat{f}_\theta(x^{(i)}) - y^{(i)})^2$$

- where m is the number of training samples, theta's are the weight parameters
- Let MSE be represented by $J(\theta)$
- Lets say we want to penalize the higher order terms (2 and 3)

Regularization

- Can add penalty terms say +1000 θ_3 + 1000 θ_4
- The effect of this would be that θ_3 and θ_4 need to be quite small to minimize error
- A significantly high penalty can actually convert a overfit problem to an underfit problem
  - Since all the terms with high regularization parameter would become 0 or close to 0
  - E.g. if all terms except θ_0 have a high enough regularization parameter then $\hat{f}(x)$ can become a constant !!!

Feature Selection

- Filter methods: …
- Wrapper methods: …
  - Recursive Feature Elimination
- Embedded methods: …

Dimensionality Reduction

Data Preprocessing

- A popular methodology in data mining is CRoss Industry Standard Process for data mining (CRISP DM)
- …

Feature Engineering

- …
- **One-hot-encoding or one-of-K:** Refers to splitting the column which contains numerical *categorical data* to many columns depending on the number of categories present in that column.
  - Each column contains "0" or "1" corresponding to which column it has been placed.

Feature Engineering

Fruit	Categorical value of fruit	Price
apple	1	5
mango	2	10
apple	1	15
orange	3	20

- After one hot encoding

apple	mango	orange	price
1	0	0	5
0	1	0	10
1	0	0	15
0	0	1	20

Feature Engineering

- …