Model Selection

Criteria to choose a model

- Error
- Training computational cost
- Test (or production) computational cost
- Ability to explain predictions

Model Complexity

- We are looking for a model with good generalization
  - Able to correctly predict values for new examples
- If the model is less complex than the function that actually generated the data it will not represent it well and underfit it
  - Using a line to fit a cuadartic function
- If the model is more complex then it will overfit and also “learn” the noise in the data
  - A 4th degree polinomial to approximate a linear process
- The complexity of a model depends on
  - Its degrees of freedom
  - The number of examples used to train it

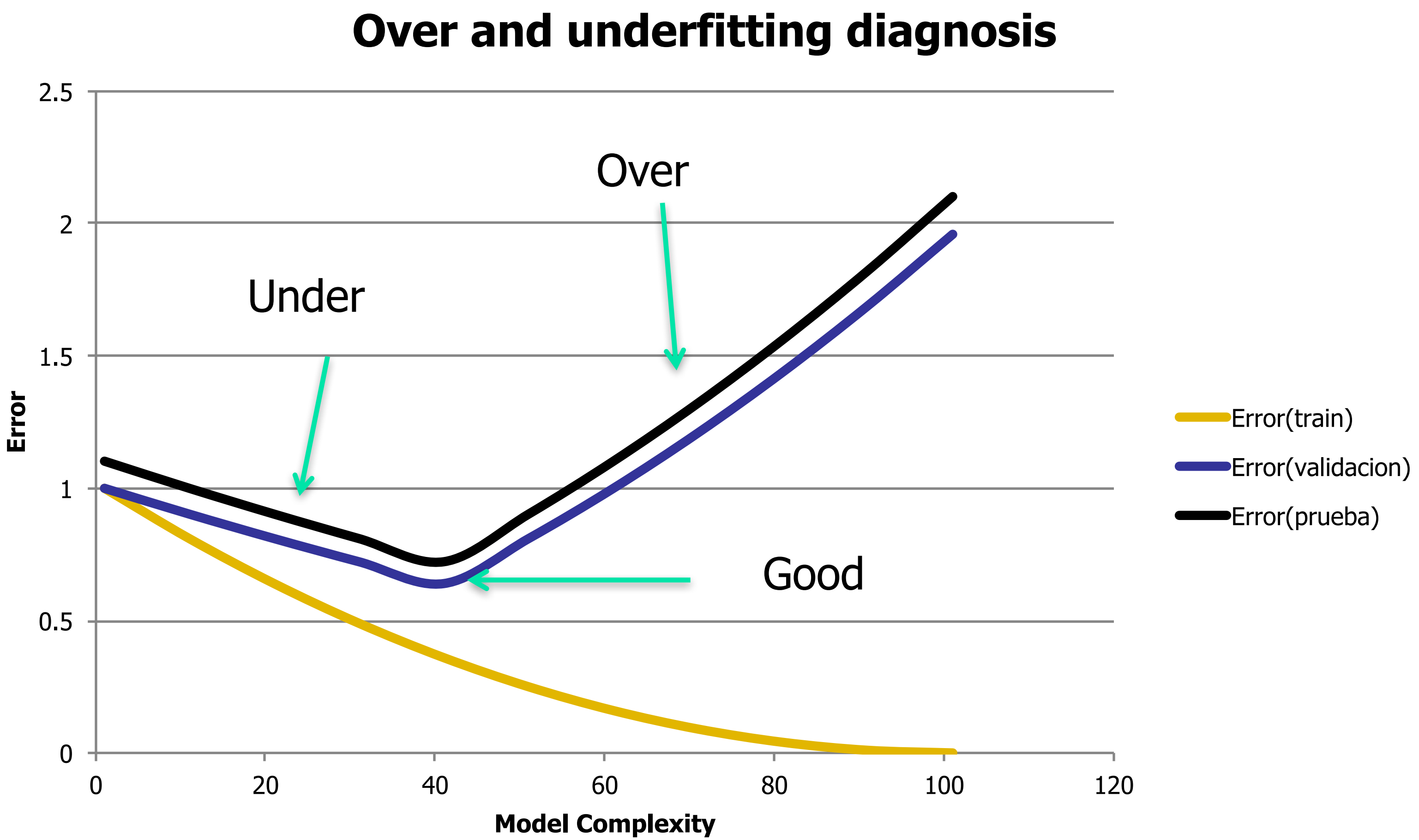
Symptoms of over and under fitting

- Underfitting
  - Large error in the training data.
  - This means that the model has a large bias.
- Overfitting
  - Low errror in the training set
  - Large error in the validation or test set
  - This means that the model has a large variance

¿How can you determine the adequate complexity?

- Evaluate the training and validation errors for different model complexities
  - For each value of the complexity parameter estimate the generalization error
    - Use cross-validation or bootstrapping

The Adequate Complexity



¿What to do?

- Ovefitting (High variance)
  - Use more training examples
  - Reduce the complexity of the model
    - Reduce the number of attributes
    - Simplify the model by tuning its parameters
      - Increase regularization constant
      - Reduce the number of neurons in a neural net
      - Decrease the height of a decision tree

¿What to do?

- Underfitting (High bias)
  - Add attributes
    - Device new variables (x*x, x*y, sin(x) etc). Use PCA, Factor analysis, etc.
    - Collect new and better variables
  - Increase the models complexity
    - Reduce the regularization constante
    - Use more neurons in a neural net
    - Change learning algorithm
  - Change data distribution

Error estimation and hyper-parameter selection

Model Creation
Basic cycle

- Partition data into two groups
  - Training Set
    - Execute learning-validation cycle to choose model parameters (lambda, number of neurons, etc)
      - Details in following slides
  - Test set
    - The test set should not be used in any part of the learning and validation cycle
- Data should be partition following the original distribution and respecting temporal ordering
  - On occasion stratified sampling is in order
  - In general 25% of the data is left for testing
- Train with all the training set
- Report the error statistics from the learning-validation cycle for the selected hyperparameters and the test set error

Model Selection
Learning-validation cycle

- Most learning algorithms have hyper-parameters that need to be tuned
  - Lambda, neural net topology, etc.
- Hyper parameters need to be learnt too
  - The learning algorithm is basically trial and error
    - For each value h to explore of the hyper parameter
      - Train the model using h and estimate its error
    - Select the h with that yields the lowest error
- We cant use the test set to select h (why?)
  - We need an extra cycle, the learning-validation cycle

Model Selection
Learning-validation cycle

- We should execute learning-validation multiple times and gather error statistics
  - Randomize initialization parameters
  - Use different data subsets
- The different data subsets are generated using cross-validation or bootstrapping

Learning-validation cycle

- Cross-validation
  - Partition data into k subsets (normally between 5 and 15)
  - Use one subset for validation and the remaining k-1 for learning
  - Repeat for each of the k subsets
  - Compute statistics of the k results (mean and standard deviation of the error)
- Bootstrapping
  - Suppose we have n data points in the training set
  - Take k samples with replacement of size n
  - Divide each sample into learning and validation
    - Remove from validation data that appears in learning
  - Compute statistics of the k results (mean and standard deviation of the error)
- Repeat process for each value of the hyper parameter and choose the one with the best statistics

Note

- Attribute selection, standarization etc., mut go inside the learning-validation cycle

Excercise

- Implement cross-validation to select the appropriate value of lambda for the iterative lineal regression model
  - Use the regLinPoli2.csv data file
  - You can use, if you wish, the Regularization4class.ipynb starter code

Error Decomposition

Error

- The learning error can be dividen into three components
  - Irreductuble error due to noisy data
  - Error due to high bias (unable to capture the phenomenon)
  - Error due to high variance (captures irrelevant aspects of the training)
- High bias is seen as underfitting and high variance as overfitting

Error Decomposition

- Suppose the real funcion is of the form:
  - $y=f(x) + \varepsilon$, where ε is noise normaly distributed with mean zero and σ^2 variance
- Our model produces a prediction
  - $V\hat{}(x)$, for every x
- And we measure error (in the case of regression) as
  - $\Sigma(y-V\hat{}\,(x))^{2},$

Error Decomposition

- We wish to estimate the error for a new data point x*

$$Err(x^*)=E[(y-\hat{V}(x^*))^2]$$

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

$$=\sigma^2+[E(\hat{V}(x^*))-f(x^*)]^2+E[\hat{V}(x^*)-E(\hat{V}(x^*))]^2$$

$$=\sigma^2+Bias^2(\hat{V}(x^*))+Var(\hat{V}(x^*))$$

$$=Irreductible\_error+Bias^2+Variance$$

- Normally there is a tradeoff between bias and variance

Error Decomposition

- Note that expectations are taken over everything that can be randomized
  - Initial weights (w's)
  - Training set (expectation over all possible training sets)
    - For example in the iterative linear regression: E(V^(x*)) is the expected output of the model over all possible training sets and all possible initial w's

Derivación Descomposición de Error

Derivación Versión 1

- Una propiedad importante (truco para derivar)
  - $Var(X)=E(X^2)-[E(X)]^2$
- Sustituimos la variable aleatoria X por la discrepancia de nuestro modelo
  - $Var(V\hat{}(x)-f(x)-\varepsilon)=Var(V\hat{}(x)) + \sigma^2$
    - Porque la varianza de f(x) es cero pues no es una variable aleatoria y la covarianza entre el ruido y V^(x) es cero

Derivación Versión 1

- De la fórmula de la varianza sustituyendo:
- $Var(V\hat{}(x)) + \sigma^2 = E[(V\hat{}(x)-f(x)-\varepsilon)^2]-(E[V\hat{}(x)-f(x)-\varepsilon])^2$
  - $E[(V\hat{}(x)-f(x)-\varepsilon)^2]=MSE$ (error cuadrático medio)
  - $(E[V\hat{}(x)-f(x)-\varepsilon])^2=(E(V\hat{}(x))-E(f(x))-E(\varepsilon))^2$

  $=[E(V\hat{}(x))-f(x)]^2 = Bias^2$
    - Porque $E(f(x))= f(x)$ y $E(\varepsilon) =0$
- Sustituyendo en la primer formula
- $Var(V\hat{}(x)) + \sigma^2=MSE - Bias^2$
- $MSE=Var(V\hat{}(x)) + \sigma^2 + Bias^2$

Version 2 Derivación

- Algunas propiedades importantes:
  1. $E(E(x))=E(x)$
  2. $E((x-E(x))^2)=E[x^2-2xE(x)-E(x)^2]$

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

     $=E(x^2)-2E(x)E(x)+E(x)^2$

     $=E(x^2)-E(x)^2$
  3. $E(x^2)=E((x-E(x))^2) +E(x)^2$ (fórmula varianza)
  4. $E((c+N(0,\sigma))x)$

     $=E(cx+xN(0,\sigma))=cE(x)$ (la covarianza es cero)

Derivación

- Regresando al error esperado:

$E[(y-V^\wedge(x^*))^2] = E[y^2 - 2yV^\wedge(x^*) + V^\wedge(x^*)^2]$

$= E[y^2] - 2E(yV^\wedge(x^*)) + E[V^\wedge(x^*)^2]$

$= E((y-E(y))^2) + E(y)^2$ (propiedad 3)

$-2E(V^\wedge(x^*))f(x^*)$ (propiedad 4)

$+ E[(V^\wedge(x^*) - E(V^\wedge(x^*)))^2] + E(V^\wedge(x^*))^2$ (propiedad 3)

$= E((y-E(y))^2)$ (ruido. El desarrollo da σ^2 usando prop.4 y 1)

$+E(y)^2 - 2E(V^\wedge(x^*))f(x^*) + E(V^\wedge(x^*))^2$ (sesgo^2 esto se reduce a $(y-E(V^\wedge(x^*)))^2$ note que $E(y)=y$)

$+E[(V^\wedge(x^*) - E(V^\wedge(x^*)))^2]$ (varianza)