Hyperparameters and Validation Sets

Sargur N. Srihari
srihari@cedar.buffalo.edu

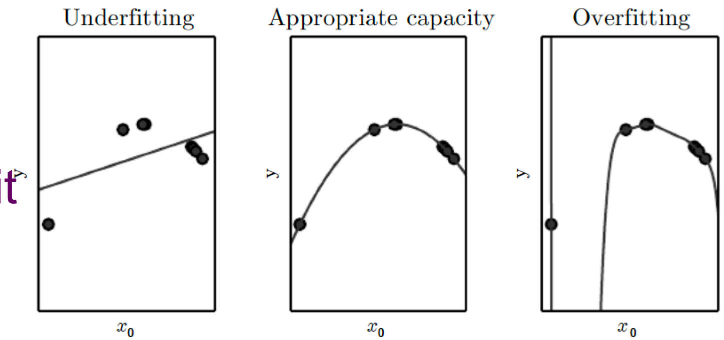
Topics in Machine Learning Basics

1. Learning Algorithms
2. Capacity, Overfitting and Underfitting
3. Hyperparameters and Validation Sets
4. Estimators, Bias and Variance
5. Maximum Likelihood Estimation
6. Bayesian Statistics
7. Supervised Learning Algorithms
8. Unsupervised Learning Algorithms
9. Stochastic Gradient Descent
10. Building a Machine Learning Algorithm
11. Challenges Motivating Deep Learning

Hyperparams control ML Behavior

- Most ML algorithms have hyperparameters
  - To control algorithm behavior
  - Values not adapted by learning algorithm itself
    - Although, can design nested learning where a learning algo learns best hyperparams for another learning algo
- In polynomial regression, a single hyperparam acts a capacity hyperparameter

Low capacity:
Struggles to fit



High capacity:
Overfit

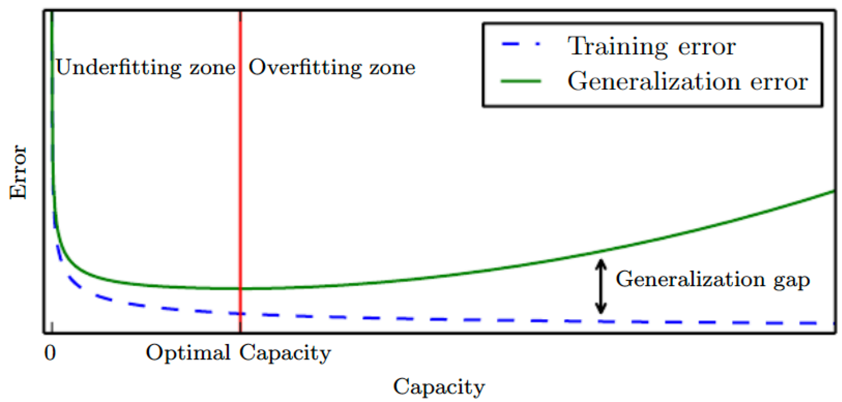
$$J(\boldsymbol{w}) = \text{MSE}_{\text{train}} + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

– Weight decay λ is also a hyperparameter

Reasons for hyperparameters

- Sometimes setting is chosen as a hyperparam because it is *too difficult* to optimize
- More frequently, the setting is a hyperparam because it is *not appropriate to learn* that hyperparam on the training set
  - Applies to all hyperparameters for model capacity
    - If learned on training set, they would always choose maximum model capacity resulting in overfitting

Can always fit the training set better with a higher degree polynomial and weight decay λ=0



Validation Set

- To solve the problem we use a validation set
  – Examples that training algorithm does not observe
- Test examples should not be used to make choices about the model hyperparameters
- Training data is split into two disjoint parts
  – First to learn the parameters
  – Other is the validation set to estimate generalization error during or after training
    - allowing for the hyperparameters to be updated
  – Typically 80% of training data for training and 20% for validation

Test sets also need to change

- Over many years, the same test set used repeatedly to evaluate performance of different algorithms
- With repeated attempts to beat state-of-the-art performance, we have optimistic evaluations with the test set as well
- Community tends to move to new, usually more ambitious and larger benchmark data sets

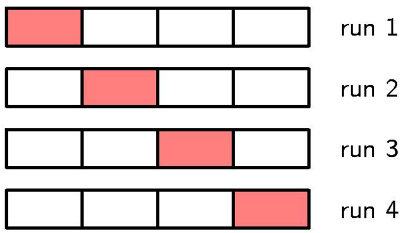
Cross-Validation

- When data set is too small, a fixed test set is problematic
- k - fold cross-validation
  - A partition of the data is formed by splitting it into k nonoverlapping subsets
  - On trial i, the i^{th} subset of the data is used as the test set
  - Rest of the data is used as the training set

k-fold Cross Validation

- Supply of data is limited
- All available data is partitioned into k groups
- k-1 groups are used to train and evaluated on remaining group
- Repeat for all k choices of held-out group
- Performance scores from k runs are averaged

k=4



If S=N this is the leave-one-out method

Variables for algorithm

- Used to estimate generalization error of a learning algorithm A from data set D
- L is the loss function, seen as a function of the learned function f and an input $\boldsymbol{z}^{(i)} \varepsilon\, \mathrm{D}$ to a scalar
- When the given data set D is too small to yield an accurate estimate of generalization error
  - i.e.., mean loss L on small test set has too high variance

k -fold cross validation algorithm

Define $\texttt{KFoldXV}(\mathbb{D}, A, L, k)$:

Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$

Require: A, the learning algorithm, seen as a function that takes a dataset as input and outputs a learned function

Require: L, the loss function, seen as a function from a learned function f and an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$

Require: k, the number of folds

Split $\mathbb{D}$ into k mutually exclusive subsets $\mathbb{D}_i$, whose union is $\mathbb{D}$.

for i from 1 to k **do**

$f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$ — Train A on dataset without D_i

for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ **do**

$e_j = L(f_i, \boldsymbol{z}^{(j)})$ — Determine errors for samples in D_i

end for

end for

Return $\boldsymbol{e}$ — Return vector of errors $\boldsymbol{e}$ for samples in D

Cross validation confidence

- Cross-validation algorithm returns vector of errors $\boldsymbol{e}$ for examples in $\mathcal{D}$
  - Whose mean is the estimated generalization error
  - The errors can be used to compute a confidence interval around the mean
    - 95% confidence interval centered around mean $\hat{\mu}_m$ is

$$\boxed{(\hat{\mu}_m - 1.96 SE(\hat{\mu}_m),\ \hat{\mu}_m + 1.96 SE(\hat{\mu}_m))}$$

where the standard error of the mean is: $\boxed{SE(\hat{\mu}_m) = \sqrt{Var\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}}}$

Which is square root of variance of the estimator

Caveats for Cross-validation

- No unbiased estimators of the average error exist; approximations are used
- Confidence intervals are not well-justified after use of cross-validation
- It is still common practice to declare that Algorithm A is better than Algorithm B only of the confidence interval of Algorithm A lies below and does not intersect the confidence interval of Algorithm B