07-280: Model Selection

Nihar B. Shah

Consider this data with $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$.

Should we use linear regression? Well, $y = wx + b$ won't be a good fit.
However, observe that the data is quadratic. So $h(x) = \theta_2 x^2 + \theta_1 x + \theta_0$ may be a good fit.

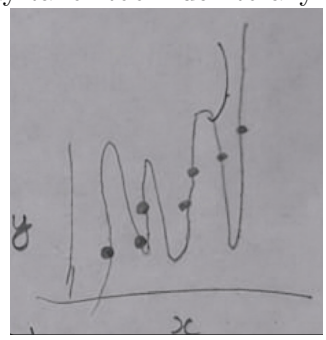
Idea! Instead of $h(x) = wx + b$, consider $h(x) = \theta_k x^k + \theta_{k-1} x^{k-1} + \cdots + \theta_1 x + \theta_0$.

Even more generally, given some $\mathcal{X}$ and $\mathcal{Y} = \mathbb{R}$, consider function $\phi : \mathcal{X} \to \mathbb{R}^{d'}$ for some d'.

Recipe:

- Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})$, compute $(\phi(x^{(1)}), y^{(1)}), (\phi(x^{(2)}), y^{(2)}), \ldots, (\phi(x^{(N)}), y^{(N)})$.
- Train a linear regressor $h : \mathbb{R}^{d'} \to \mathbb{R}$.
- Given a new point $x^{(\text{new})} \in \mathcal{X}$, output $h(\phi(x^{(\text{new})}))$ as your prediction.

How to choose k?
We could try using ERM. But ERM will generally choose higher k. Higher degree polynomials may tailor too much to any noise in the data.



This is called **"Overfitting."**

One way to address this issue: get more training data! But this is often be expensive. We will discuss two other methods: **Regularization** & **Cross Validation**.

Regularization Consider the following ERM expression augmented with an additional term:

$$\arg\min_{\theta\in\mathbb{R}^{d'}} \frac{1}{N}\sum_{i=1}^{N}\left(\theta^\top\phi(x^{(i)}) - y^{(i)}\right)^2 + \lambda\, r(\theta).$$

The additional term is a **regularizer**, $r : \mathbb{R}^{d'} \to \mathbb{R}$, which captures complexity of θ. The scalar $\lambda > 0$ specifies the weight to put on this regularizer versus the empirical risk.

Let us get some more understanding of the regularizer. First, observe that a more complex model (higher K in the polynomial setting) will have lower empirical risk simply because it can fit more complex data. Thus to counter that, the function r has higher values for more complex models. Given such a regularization term, now in order for a complex model to be chosen, the lowering of the empirical risk should be large enough. Otherwise a simpler model is chosen.

Examples:

- $r(\theta) = \sum_{j=1}^{d'} \mathbf{1}\{\theta_j \neq 0\}$ Number of non-zero entries.

  Encourages sparse features, i.e., fewer number of non-zero entries of θ. However, finding arg min is computationally challenging.

- $r(\theta) = \sum_{j=1}^{d'} |\theta_j|$ "LASSO". Also encourages sparsity.

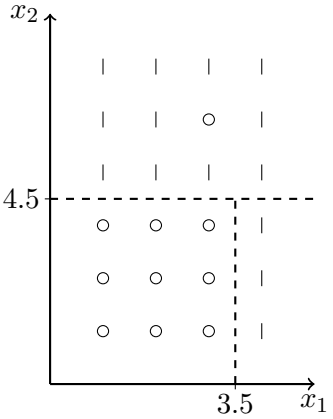
- $r(\theta) = \sum_{j=1}^{d'} \theta_j^2$ "Ridge regression"

Few more comments:

- Exclude bias term θ_0 from the regularizer.
- Preprocessing: Ensure all coordinates have the same units and are on the same scale (e.g., by subtracting the mean and dividing by the variance across all training samples).
- There are also other ways of regularizing that we didn't get time to discuss here (e.g., dropout, early stopping).
- How to choose λ? Cross Validation...

Cross Validation

Recall decision trees. Consider the following training data and two splits:



Should you keep splitting until the outlier point near the top right is classified correctly? An argument for that is that it may contain useful information. Or should we stop earlier? An argument for stopping is that this label may just be noise.

More generally, choice of tree depth, λ, degree of polynomial k are all called **hyperparameters**: ERM is biased towards one type of answer (more complex models) and may not capture the tradeoffs.

Now think about this: What if we had access to some more (labeled) data. We could evaluate both trees (one with further splits and the one without) on this other data and pick the one that has lower error.

Idea: Split training samples (randomly) into a "training set" and a "validation set."

- For each value of the hyperparameter(s):
  - Train on training set.
  - Compute average error of the obtained model on the samples in the validation set.
- Choose the value of the hyperparameter(s) which led to lowest error on validation set.
- Then retrain model under chosen hyperparameters on the entire data.

This is called **"holdout cross validation."** Typically, sizes of train:validation sets is 80:20.

Problem? If original training data doesn't have many samples, this suffers from high variance.

We can instead use k**-fold cross validation**: Here we randomly partition the training data into k equally-sized "folds" $F_1, F_2, \ldots, F_k$. Do the following for each value of the hyperparameter(s). For each fold $i \in \{1, \ldots, k\}$:

- Use F_i as the validation set.
- Use the remaining $k-1$ folds, $\bigcup_{j \neq i} F_j$, as the training set.
- Train the model on the training set and compute the error on the validatino set F_i.

This gives us k different error estimates. Average them to get the cross-validation error as $\frac{1}{k}\sum_{i=1}^{k} \text{error}_i$. Repeat this procedure for each candidate value of the hyperparameter(s) and choose the one with the lowest cross-validation error. Finally, as in holdout crossvalidation, retrain on the full dataset using the chosen hyperparameters.

Common choices for the number of folds are $k = 5$ or $k = 10$. The extreme case $k = N$ is called **leave-one-out cross validation** (LOOCV).

Finally, we make a quick note on a relevant distinction between feature engineering and feature learning. We will revisit and discuss this in later lectures.

Feature Engineering	**Feature Learning**
Humans decide what features may be useful.	Automatically discover useful features/representations from the data.
Humans implement feature extraction algorithms.	Typically needs large datasets.
Higher interpretability.	