Lecture 15: MATH 342W: Introduction to Data Science and Machine Learning

Sergio E. Garcia Tapia*

April 1st, 2025 (last updated May 13, 2025)

Introduction to Model Selection

Last lecture, we introduced the notion of *transformed* or *derived* features. Throughout our study, we have used p to denote the number of features. To account for features before and after transformations, we use p_{raw} to denote the number of features before transformations, and p to denote the number of features after transformations, which includes the raw features and the derived features. Our motivation was to reduce misspecification error by allowing for a more comprehensive set $\mathcal{H}$ of candidate functions (for example, to fit parabolas when lines are insufficient).

The problem we now face is that there are too many ways to derive features (with interactions, without interactions, etc). Moreover, there are too many choices of models, each consisting of an algorithm and a set of candidate functions. How can we select the best model?

Suppose for now that there are M pre-specified algorithm-candidate set pairs:

$$(\mathcal{A}_1, \mathcal{H}_1), (\mathcal{A}_2, \mathcal{H}_2), \ldots, (\mathcal{A}_M, \mathcal{H}_M)$$

For example, we might have:

$$\begin{aligned}
&\mathcal{A}_1 = \text{OLS}, \quad \mathcal{H}_1 = \{\, w_0 + w_1 x \mid \boldsymbol{w} \in \mathbb{R}^2 \,\} \\
&\mathcal{A}_2 = \text{OLS}, \quad \mathcal{H}_2 = \{\, w_0 + w_1 x + w_2 x^2 \mid \boldsymbol{w} \in \mathbb{R}^3 \} \\
&\mathcal{A}_3 = \text{OLS}, \quad \mathcal{H}_3 = \{\, w_0 + w_1 x + w_2 \ln(x) \mid \boldsymbol{w} \in \mathbb{R}^3 \} \\
&\mathcal{A}_4 = \text{OLS or logistic}, \quad \mathcal{H}_4 = \{\ldots\}
\end{aligned}$$

We wish to:

(i) select the best among the M candidates;

(ii) compute a conservative estimate of future performance for the chosen candidate;

(iii) output a prediction function g_{final}.

*Based on lectures of Dr. Adam Kapelner at Queens College. See also the course GitHub page.

Example. Model selection is a fundamental problem of science. For example, there were originally a few classical models of gravitation:

(1) $F = G\frac{m_1 m_2}{r^2}$

(2) $F = G_1\frac{m_1 m_2}{r^2} + G_2\frac{m_1 m_2}{r^3}$

(3) $F = G_1\frac{m_1 m_2}{r^2}e^{-G_2 r}$

Model (2) might have been a better fit, but there may have been other trade-offs that caused it losing favor over (1).

No Cross-Validation (CV)

Previously, we started with our set of historical data $\mathbb{D}$

$$\underbrace{\begin{bmatrix} \\ \mathbb{D} \\ \\ \end{bmatrix}}_{n \times p}$$

Then, we split (partitioned) it into $\mathbb{D}_{\text{train}}$ and $\mathbb{D}_{\text{test}}$:

$$\underbrace{\begin{bmatrix} \\ \mathbb{D}_{\text{train}} \\ \\ \end{bmatrix}}_{n_{\text{train}} \times p} \quad \underbrace{\begin{bmatrix} \mathbb{D}_{\text{test}} \end{bmatrix}}_{n_{\text{test}} \times p}, \quad n_{\text{train}} + n_{\text{test}} = n$$

We did this to obtain honest estimate of out-of-sample performance. A natural extension of this idea is to run each of the M algorithms on $\mathbb{D}_{\text{train}}$ with their respective candidate sets, and compute the out-of-sample performance metric of each on $\mathbb{D}_{\text{test}}$:

$$\begin{aligned}
g_1 = \mathcal{A}_1(\mathbb{D}_{\text{train}}, \mathcal{H}_1) &\implies \text{predict on } \mathbb{D}_{\text{test}} \implies \boldsymbol{e}_{\text{test},1} \implies oosRMSE_1 \\
g_2 = \mathcal{A}_2(\mathbb{D}_{\text{train}}, \mathcal{H}_2) &\implies \text{predict on } \mathbb{D}_{\text{test}} \implies \boldsymbol{e}_{\text{test},2} \implies oosRMSE_2 \\
&\vdots \\
g_M = \mathcal{A}_M(\mathbb{D}_{\text{train}}, \mathcal{H}_M) &\implies \text{predict on } \mathbb{D}_{\text{test}} \implies \boldsymbol{e}_{\text{test},M} \implies oosRMSE_M
\end{aligned}$$

Then, we can pick the algorithm-candidate set pair for which the out-of-sample RMSE metric is minimized:

$$m_* = \underset{m \in \{1,2,\ldots,M\}}{\text{argmin}} \{oosRMSE_m\}$$

Hence, according to this scheme, the best model is $(\mathcal{A}_{m_*}, \mathcal{H}_{m_*})$. In expectation (that is, on average), this is a good way to choose the best model. However, what we do not get out of this is an estimate of future predictive performance. If M is large, it's possible that we "get lucky", meaning we get an algorithm-candidate set pair that optimizes for $\mathbb{D}_{\text{train}}$ and $\mathbb{D}_{\text{test}}$. That is, we may get a pair $(\mathcal{A}_k, \mathcal{H}_k)$ that "accidentally" does well on

$\mathbb{D}_{\text{train}}$ and $\mathbb{D}_{\text{test}}$, but still performs poorly out-of-sample. This is a form of overfitting. We need to be careful to treat $\mathbb{D}_{\text{test}}$ as an opaque box that we open only once.

To address this, we further partition $\mathbb{D}$ so that we have three pieces:

$$\underbrace{\left[\ \mathbb{D}_{\text{train}}\ \right]}_{n_{\text{train}} \times p} \quad \underbrace{\left[\ \mathbb{D}_{\text{select}}\ \right]}_{n_{\text{select}} \times p} \quad \underbrace{\left[\ \mathbb{D}_{\text{test}}\ \right]}_{n_{\text{test}} \times p}$$

where $n = n_{\text{train}} + n_{\text{select}} + n_{\text{test}}$ is the total number of observations and

$$\begin{aligned}
n_{\text{test}} &= \frac{1}{K_{\text{test}}} \cdot n = \text{Number of rows in } \mathbb{D}_{\text{test}} \\
n - n_{\text{train}} &= \left(1 - \frac{1}{K_{\text{test}}}\right) \cdot n = \text{Number of rows in } \mathbb{D}_{\text{train}} \cup \mathbb{D}_{\text{select}} = \mathbb{D} \setminus \mathbb{D}_{\text{test}} \\
n_{\text{select}} &= \frac{1}{K_{\text{select}}}(n - n_{\text{train}}) = \frac{1}{K_{\text{select}}}\left(1 - \frac{1}{K_{\text{test}}}\right) n = \text{Number of rows in } \mathbb{D}_{\text{select}}
\end{aligned}$$

Then we repeat the process from earlier, replacing $\mathbb{D}_{\text{test}}$ with $\mathbb{D}_{\text{select}}$:

- **Step 1**: Train all M models on $\mathbb{D}_{\text{train}}$ and compute their respective out-of-sample metrics on $\mathbb{D}_{\text{select}}$:

$$\begin{aligned}
g_1 = \mathcal{A}_1(\mathbb{D}_{\text{train}}, \mathcal{H}_1) &\implies \text{predict on } \mathbb{D}_{\text{select}} \implies \boldsymbol{e}_{\text{select},1} \implies oosRMSE_1 \\
g_2 = \mathcal{A}_2(\mathbb{D}_{\text{train}}, \mathcal{H}_2) &\implies \text{predict on } \mathbb{D}_{\text{select}} \implies \boldsymbol{e}_{\text{select},2} \implies oosRMSE_2 \\
&\vdots \\
g_M = \mathcal{A}_M(\mathbb{D}_{\text{train}}, \mathcal{H}_M) &\implies \text{predict on } \mathbb{D}_{\text{select}} \implies \boldsymbol{e}_{\text{select},M} \implies oosRMSE_M
\end{aligned}$$

- **Step 2**: Select the algorithm and candidate set $(\mathcal{A}_{m*}, \mathcal{H}_{m_*})$ for which the out-of-sample metric was the smallest:

$$m_* = \underset{m \in \{1,2,\ldots,M\}}{\text{argmin}} \{oosRMSE_m\}$$

- **Step 3**: Run $\mathcal{A}_{m_*}$, but now on $\mathbb{D}_{\text{train}} \cup \mathbb{D}_{\text{select}}$ to obtain a new prediction function:

$$g_{m_{**}} = \mathcal{A}_{m_*}(\mathbb{D}_{\text{train}} \cup \mathbb{D}_{\text{select}}, \mathcal{H}_{m_*})$$

  (in class we overloaded the g_{m*} name, but here I will use double-star to distinguish $g_{m_{**}}$ obtained by running $\mathcal{A}_{m_*}$ on $\mathbb{D}_{\text{train}} \cup \mathbb{D}_{\text{select}}$ from g_{m_*} which was obtained by running $\mathcal{A}_{m_*}$ only on $\mathbb{D}_{\text{train}}$).

- **Step 4**: Compute the out-of-sample metric for $g_{m_{**}}$ by predicting on $\mathbb{D}_{\text{test}}$:

$$g_{m_{**}} \implies \text{predict on } \mathbb{D}_{\text{test}} \implies \boldsymbol{e_{m_{**}}} \implies oosRMSE_{m_{**}}$$

- **Step 5**: We run algorithm m_* now on the entire data set:

$$g_{\text{final}} = \mathcal{A}_{m_*}\left(\underbrace{\mathbb{D}_{\text{train}} \cup \mathbb{D}_{\text{select}} \cup \mathbb{D}_{\text{test}}}_{\mathbb{D}}, \mathcal{H}_{m_*}\right)$$

  The value of $oosRMSE_{m_{**}}$ computed in the previous step is now a conservative estimate of the future performance of g_{final}.

We refer to this algorithm as **No Cross-Validation (No CV)**. We might view this as a meta-algorithm, where each algorithm $\mathcal{A}_m$ is viewed as a feature. If M is too large, we are at risk of overfitting.

Select Cross-Validation (CV)

In the No-CV approach, each of $\mathbb{D}_{\text{train}}$, $\mathbb{D}_{\text{select}}$, and $\mathbb{D}_{\text{test}}$ was fixed. We can improve upon that approach by fixing $\mathbb{D}_{\text{test}}$, but applying a K_{select}-fold cross-validation on each of the M models.

- **Step 1**: Fix a value for K_{test}, denoting the proportion of $\mathbb{D}$ to use for $\mathbb{D}_{\text{test}}$, so that $\mathbb{D}_{\text{test}}$ is fixed for the rest of the algorithm, which will be used in the final validation step. Hence, $\mathbb{D} \setminus \mathbb{D}_{\text{test}}$ is also fixed.
- **Step 2**: Fix a value of K_{select}, denoting the proportion $\mathbb{D} \setminus \mathbb{D}_{\text{test}}$ that will be used for testing each of the M algorithms. Note that K_{select} is fixed but $\mathbb{D}_{\text{select}}$ is not.
- **Step 3**: Shuffle the data set $\mathbb{D} \setminus \mathbb{D}_{\text{test}}$.
- **Step 4**: For $m \in \{1, 2, \ldots, M\}$, that is, for each pair $(\mathcal{A}_m, \mathcal{H}_m)$:
  - For $k \in \{1, 2, \ldots, K_{\text{select}}\}$:
    - Take the next piece (the kth piece) of proportion K_{select} from $\mathbb{D} \setminus \mathbb{D}_{\text{test}}$ to use as $\mathbb{D}_{\text{select},k}$ (say, we start from the bottom of $\mathbb{D} \setminus \mathbb{D}_{\text{test}}$). What remains is $\mathbb{D}_{\text{train},k}$.
    - Compute a prediction function $g_k^{(m)} = \mathcal{A}_m(\mathbb{D}_{\text{train},k}, \mathcal{H}_m)$.
    - Predict on $\mathbb{D}_{\text{select},k}$ using $g_k^{(m)}$, yielding error vector $\boldsymbol{e}_k^{(m)}$.
  - After doing K_{select} folds for the mth model, stack the residual vectors into a vector of length $n_{\text{train}} + n_{\text{select}}$:

$$\boldsymbol{e}_m = \begin{bmatrix} \boldsymbol{e}_{K\text{select}}^{(m)} \\ \vdots \\ \boldsymbol{e}_2^{(m)} \\ \boldsymbol{e}_1^{(m)} \end{bmatrix}$$

  - Compute $oosRMSE_m$ using $\boldsymbol{e}_m$. (How? Is it the mean?)
- **Step 5**: Now we have $oosRMSE_1, oosRMSE_2, \ldots, oosRMSE_M$. Take m_* to be

$$m_* = \underset{m \in \{1,2,\ldots,M\}}{\text{argmin}} \{oosRMSE_m\}$$

- **Step 6**: Train model $(\mathcal{A}_{m_*}, \mathcal{H}_{m_*})$ on $\mathbb{D} \setminus \mathbb{D}_{\text{test}}$ to get $g_* = \mathcal{A}_{m_*}(\mathbb{D} \setminus \mathbb{D}_{\text{test}}, \mathcal{H}_{m_*})$.
- **Step 7**: Predict using g_* on $\mathbb{D}_{\text{test}}$ to obtain $oosRMSE_*$.
- **Step 8**: Train model $(\mathcal{A}_{m_*}, \mathcal{H}_{m_*})$ on $\mathbb{D}$ to get $g_{\text{final}} = \mathcal{A}_{m_*}(\mathbb{D}, \mathcal{H}_{m_*})$. We use $oosRMSE_*$ as a conservative estimate of the future out-of-sample performance of g_{final}.

There is a trade-off here in that this requires more computation, but we may get a better model.

Select and Test CV

A third approach we can use is called **select and test CV**, or **nested resampling**. This approach reduces the variation of $oosRMSE_*$ in g_{final} by varying $\mathbb{D}_{\text{test}}$ rather than keeping it fixed. We pay in computational expense, which is determined by $K_{\text{test}} \cdot K_{\text{select}} \cdot M$. However, we are more sure that we pick the best of our M models. Note the algorithm for select-CV had a nested loop; this algorithm would add yet another level of nesting K_{test}.

Click here to navigate to a page with an example to visualize nested resampling.

Applying Model Selection Procedures

The three model selection procedures we just discussed (no CV, select CV, and select and test CV) are useful in many settings. The three we will study are:

(1) Pick one model among M pre-specified modeling precedures, provide a conservative estimate of future performance, and compute g_{final}.

(2) Greedy Forward Stepwise Modeling.

(3) Hyperparameter Selection.

Recall that there are infinitely-many candidate sets. Therefore (1) can be unrealistic. (2) provides an alternative.

Greedy Forward Stepwise Modeling

- **Step 0**: Create a very large set of derived features so that $\mathcal{H}$ can be very expansive (large). For example:

$$\begin{aligned}&\{x_1, \ldots, x_p, x_1^2, \ldots, x_p^2, x_1^3, \ldots, x_p^3, \ln(x_1), \ldots, \ln(x_p)\\ &x_1x_2, \ldots, x_1x_p, \ldots, x_2x_3, \ldots, x_2x_p, \ldots, x_{p-1}x_p\\ &x_1x_2x_3, \ldots, x_{p-2}x_{p-1}x_p, \sin(x_1), \ldots, \sin(x_p)\}\end{aligned}$$

  Note that $(p+1) \gg n$. We could build a design matrix X that tracks these features, but it would not be invertible, and hence we would not be able to apply OLS. How many OLS models do we have with such a large set? A total of $2^{p+1}!$ We will search this space greedily (not optimally), which will give us a decent solution.

  Begin with some $g_0 \in \mathcal{H}_0$ (e.g. the null model), where $\mathcal{H}_0 = \{w_0 : w_0 \in \mathbb{R}\}$ (this is our "seed").

- **Step 1**: For each of the p features, fit OLS to

$$\mathcal{H} = \{\, w_0 + w_1 \underbrace{()}_{\text{feature}} \mid \boldsymbol{w} \in \mathbb{R}^2 \,\}$$

  using $\mathbb{D}_{\text{train}}$. Then predict using $\mathbb{D}_{\text{select}}$ to obtain an $oosRMSE$ for each. Record the best feature j_* based on the best $oosRMSE$.

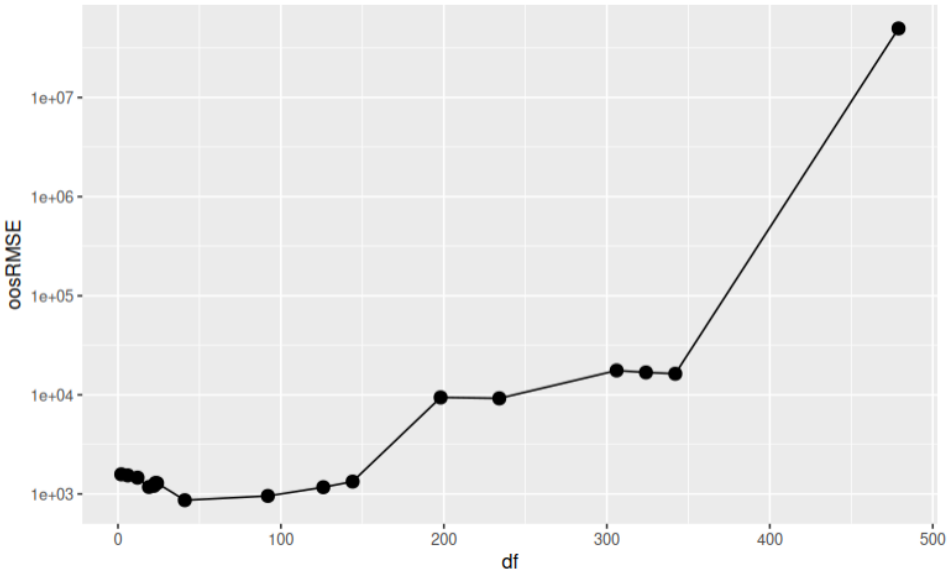

Figure 1: Greedy Forward Stepwise Modeling, tracking the out-of-sample RMSE across iterations

- **Step 2**: For each of the remaining $p-1$ features, fit OLS to

$$\mathcal{H} = \{\, w_0 + w_1(\text{feature}_{j_{*_1}}) + w_2 \underbrace{()}_{\text{feature}} \mid \boldsymbol{w} \in \mathbb{R}^3 \,\}$$

  using $\mathbb{D}_{\text{train}}$. Then predict using $\mathbb{D}_{\text{select}}$ to compute an $oosRMSE$ for each. Record the best feature j_{*_2} based on the best $oosRMSE$. Note that it cannot be j_{*_1} because that feature is not under consideration in the remaining $p-1$ features.

- **Step F**: After F steps, where F is a pre-determined maximum, or at each step, look at the results of the $oosRMSE$ from $\mathbb{D}_{\text{select}}$, and when you are sure that the $oosRMSE$ is increasing, stop. The reason it may start to increase is due to overfitting. Suppose this occurs at iteration t_* (see Figure 1). Then compute a model

$$g_* = \mathcal{A}(\mathbb{D}_{\text{train}} \cup \mathbb{D}_{\text{select}}, \mathcal{H}_{t_*})$$

  Use g_* to predict on $\mathbb{D}_{\text{test}}$ to get $\boldsymbol{e}$ and compute $oosRMSE$, an estimate of future performance. Then output

$$g_{\text{final}} = \mathcal{A}(\mathbb{D}, \mathcal{H}_{t_*})$$

Hyperparameter Selection

Recall the SVM algorithm with the Vapnik objective function:

$$\mathcal{A}_\lambda : \boldsymbol{b} = \underset{\boldsymbol{w}\in\mathbb{R}^{p+1}}{\text{argmin}} \left\{ \frac{1}{n}\sum_{i=1}^{n} H_i + \lambda\|\boldsymbol{w}\| \right\}$$

As before, $\|\boldsymbol{w}\|$ is the size of the wedge, which we are trying to maximize. For each i, H_i denotes the ith hinge error, which is 0 for the points that are correctly classified and

otherwise it is a positive number denoting the distance of a misclassified point from the hyperplane across the wedge defined by the hyperplane. The value λ is the hyperparameter of the algorithm, which needs to be specified numerically in advance. Hence, each different value of λ results in a different algorithm because the objective function changes, so that we have a family of algorithms $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{R}}$.

Suppose we create a grid of M values that λ can take on. For example:

$$\lambda_{\text{grid}} = \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$$

Now we can use no-CV, select-CV, or nested resampling to pick the best $\lambda \in \lambda_{\text{grid}}$. The weakness in this approach is having to pre-specify M models, which in this setting translates in pre-specifying M hyperparameters. To do this effectively, we need a reasonable range of λ values to consider. If our algorithm has more than one hyperparameter, our λ_{grid} might be two-dimensional, i.e., a set of tuples.