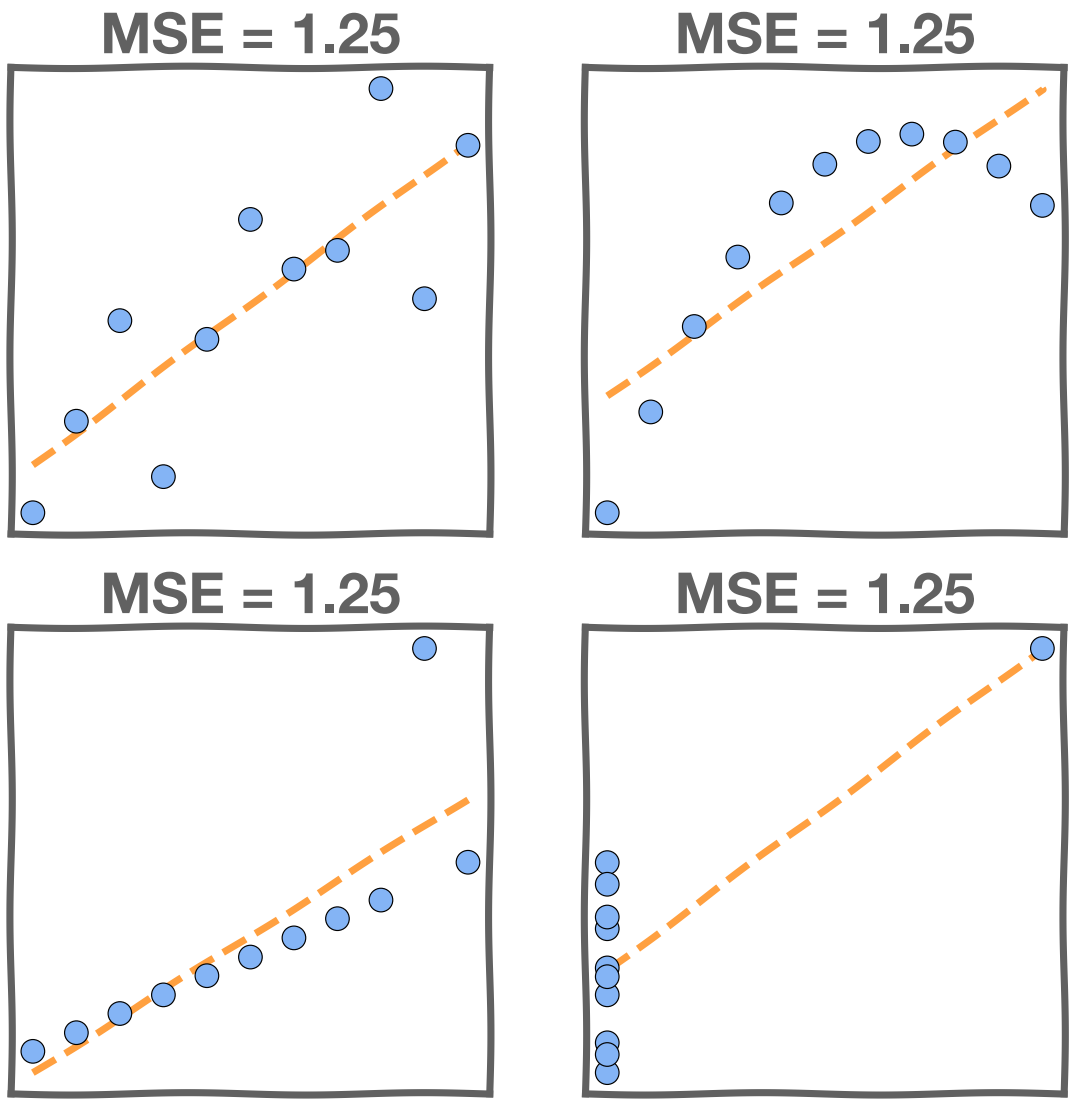
Model Selection

Bookmark this page

Evaluation: Training Error

Just because we find a model that minimizes the squared error doesn't mean that it's a good model. We should also investigate the R^2 as well. Consider these four models:



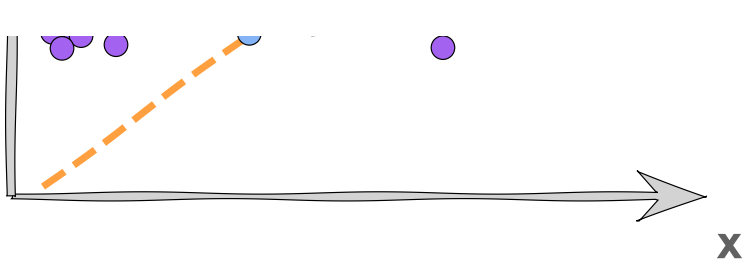
All four have the same MSE, but the fits to the data look very different. If MSE were our only criterion we would have to consider all models equally good. Let's add another criterion to our model selection to help us pick the right one.

Evaluation: Test Error

> **Definition: Generalization**
>
> The ability of a model to do well on *new* data is called **generalization**.

The goal of **model selection** is to choose the model that generalizes the best. To test if our model's performance generalizes to previously unseen examples we need to evaluate it on a *separate* set of data that the model did *not* train on. We call this the **test data**. Here's an example where our model fits the training set well but doesn't fit the test set. This is because the training data contains a strange point – an outlier – which confuses the model.

Calculator



In general, we evaluate the model on both train and test data, because models that do well on training data may do poorly on new data. The training MSE here is 2.0, while the test MSE is 12.3. Evaluating the model on the test data makes it clear that the model doesn't work well for the full data set.

When a model is strongly influenced by aspects of its particular training data that do not generalize to new data we call this **overfitting**.

Model Selection

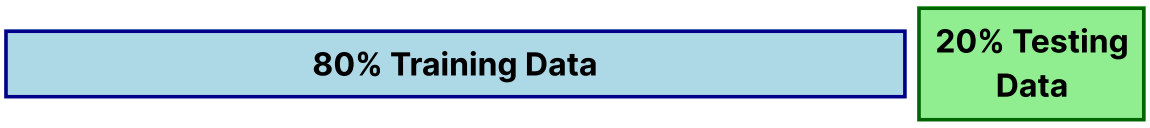
Model selection is the application of a principled method to determine the complexity of the model. This could be choosing a subset of predictors, choosing the degree of the polynomial model, etc.

A strong motivation for performing model selection is to avoid overfitting, which we saw can happen when:

- there are too many predictors
- the feature space has high dimensionality
- the polynomial degree is too high
- too many interaction terms are considered
- the coefficients values are too extreme (we have not seen this yet)

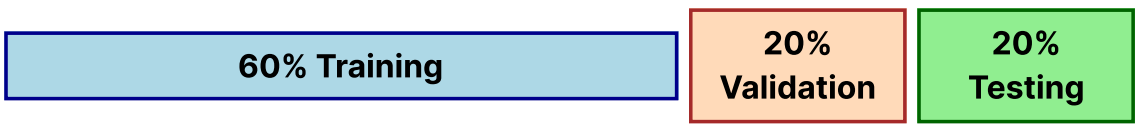
Train-Test Split

So far, we have been using the train split to train a model and the test split to evaluate the model performance.



However, we can improve our methods by splitting the data again.

Train-Validation Split



As before, we use the train split to train a model and the test split to evaluate the final model's performance. But now we'll introduce a new sub-set, called **validation**, and use *this* to select the model.

> ### Can I peek?
>
> Of course you'll use the training data... but what about that test data? How will you know whether you're doing your training right without looking at it? Patrick had that same question. Here's how he resolved the issue. Spoiler warning: Don't peek!

Video

Video

Descargar archivo de video

Transcripciones

Descargar SubRip (.srt) file

Descargar Text (.txt) file

Selection Approach

There are several approaches to model selection:

- Exhaustive search (below)
- Greedy algorithms (below)
- Fine tuning hyper-parameters (later)
- Regularization (later)

Exhaustive Search

Could we simply evaluate models trained on all possible combinations of predictors and select the best?

How many potential models are there when we have J predictors and consider only linear terms?

For example, consider 3 predictors X_1, X_2, X_3:

$$
\left.\begin{aligned}
&\textbf{- Models with 0 predictor:}\\
&\qquad M_0 : \textbf{bias only}\\
&\textbf{- Models with 1 predictor:}\\
&\qquad M_1 : X_1\\
&\qquad M_2 : X_2\\
&\qquad M_3 : X_3\\
&\textbf{- Models with 2 predictors:}\\
&\qquad M_4 : \{X_1, X_3\}\\
&\qquad M_5 : \{X_2, X_3\}\\
&\qquad M_6 : \{X_3, X_1\}\\
&\textbf{- Models with 3 predictors:}\\
&\qquad M_7 : \{X_1, X_2, X_3\}
\end{aligned}\right\} 2^J \textbf{ models}
$$

Calculator

If we were able to evaluate all 2^J models we could be sure that we've selected the best among these candidates. However, this quickly becomes intractable as the number of predictors grows.

Greedy Algorithms

With a greedy algorithm we give up on trying to find the *best* possible model as this is not practical. Instead, we content ourselves with finding a locally optimal solution.

Stepwise Variable Selection and Validation

We then need to define a process for searching and selecting a (locally) optimal sub-set of predictors, including choosing the degree for polynomial models. One approach is stepwise variable selection and validation.

Here we iteratively building an optimal subset of predictors by optimizing a fixed model evaluation metric each time on a validation set. The algorithm is "greedy" in the sense that we are only concerned with optimization at the current iteration. There is a **forward** and a **backward** version of stepwise variable selection.

Stepwise Variable Selection: Forward Method

In forward selection, we find an 'optimal' set of predictors by iterative adding to our set.

1. Starting with the empty set P_0, construct the null model, M_0.
2. For $k = 1, \ldots, J$:

‹ Previous Next ›

Security
Media Kit

Calculator