Many methods can be seen as:

- decision trees
- logistic functions
- linear functions
- kernel functions
- lower dimensional subspace

E.g., neural networks, regression trees, random forest, ...

Some combinations don’t help.
Handling Overfitting

- Overfitting occurs when the system finds regularities in the training set that are not in the test set.
- Prefer simpler models. How do we trade off simplicity and fit to data?
- Test it on some hold-out data.
Bayes Rule:

\[ P(h|d) \propto P(d|h)P(h) \]

\[ \arg \max_h P(h|d) = \arg \max_h P(d|h)P(h) \]

\[ = \arg \max_h (\log P(d|h) + \log P(h)) \]

\[ \log P(d|h) \text{ measures fit to data} \]

\[ \log P(h) \text{ measures model complexity} \]
Bayes Rule:

\[ P(h|d) \propto P(d|h)P(h) \]

\[ \arg \max_h P(h|d) = \arg \max_h P(d|h)P(h) \]

\[ = \arg \max_h (\log P(d|h) + \log P(h)) \]

- \( \log P(d|h) \) measures fit to data
- \( \log P(h) \) measures model complexity
Regularization

Logistic regression:

\[
\text{minimize } Error_E(\overline{w}) = \sum_{e \in E} \left( Y(e) - f\left( \sum_i w_i X_i(e) \right) \right)^2.
\]

L2 regularization:

\[
\text{minimize } \sum_{e \in E} \left( Y(e) - f\left( \sum_i w_i X_i(e) \right) \right)^2 + \lambda \sum_i w_i^2
\]

L1 regularization:

\[
\text{minimize } \sum_{e \in E} \left( Y(e) - f\left( \sum_i w_i X_i(e) \right) \right)^2 + \lambda \sum_i |w_i|
\]

\(\lambda\) is a parameter to be learned.
Cross Validation

Idea: split the training set into:

- new training set
- validation set

Use the new training set to train on. Use the model that works best on the validation set.

- To evaluate your algorithm, the test should not be used for training or validation.
- Many variants: k-fold cross validation, leave-one-out cross validation, ...