Evaluating ML systems

Statistical Natural Language Processing 1

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2024/2025

version: Overlaw 2025-01-09

Overfitting & Underfitting

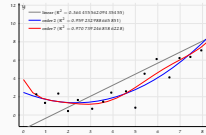
We want our models to *generalize*, perform well on unseen data.

- *Overfitting* occurs when the model learns the idiosyncrasies of the training data
- *Underfitting* occurs when the model is not flexible enough for solving the problem at hand

We want simpler models, but not too simple for the task at hand.

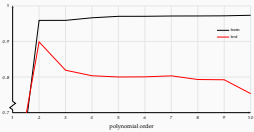
Preventing underfitting

example with polynomial basis functions



Overfitting

demonstration through polynomial regression



Preventing overfitting

- A straightforward approach is to chose a simpler model (family), e.g., by reducing the number of predictors
- More/diverse training data helps: it is *less likely* to overfit if number of training instances are (much) larger than the parameters
- There are other methods (one is coming on the next slide)
- We will return to this topic frequently during later lectures

Regularized parameter estimation

- *Regularization* is a general method for avoiding overfitting
- The idea is to constrain the parameter values in addition to minimizing the training error
- For example, the regression estimation becomes:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{k} w_j^2$$

- The new part is called the regularization term,
- λ is a *hyperparameter* that determines the strength of the regularization
- In effect, we are preferring small values for the coefficients
- Note that we do not include w_0 in the regularization term

L2 regularization

The form of regularization, where we minimize the regularized cost function,

$$\mathsf{E}(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2$$

is called L2 regularization.

- Note that we are minimizing the L2-norm of the weight vector
- In statistic literature L2-regularized regression is called *ridge regression*
- The method is general: it can be applied to other ML methods as well
- The choice of λ is important
- Note that the scale of the input also becomes important

L1 regularization

In L1 regularization we minimize

$$\mathsf{E}(\boldsymbol{w}) + \lambda \sum_{j=1}^{k} |w_j|$$

- The additional term is the L1-norm of the weight vector (excluding w_0)
- In statistics literature the L1-regularized regression is called *lasso*
- The main difference from L2 regularization is that L1 regularization forces some values to be 0 – the resulting model is said to be 'sparse'

Regularization as constrained optimization

L1 and L2 regularization can be viewed as minimization with constraints

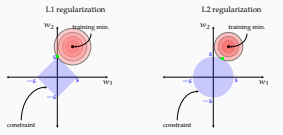
L2 regularization

$$\text{Minimize} \quad \mathsf{E}(\boldsymbol{w}) \quad \text{with constraint} \quad \|\boldsymbol{w}\|^2 < s$$

L1 regularization

$$\text{Minimize} \quad \mathsf{E}(\boldsymbol{w}) \quad \text{with constraint} \quad \|\boldsymbol{w}\|_1 < s$$

Visualization of regularization constraints



Regularization: some remarks

- Regularization prevents overfitting
- The *hyperparameter* λ needs to be determined
  - best value is found typically using a *grid search*, or a *random search*
  - it is tuned on an additional partition of the data, *development* set
  - development set cannot overlap with training or test set
- The regularization terms can be interpreted as *priors* in a Bayesian setting
- Particularly, L2 regularization is equivalent to a normal prior with zero mean

Bias and variance

Bias of an estimate is the difference between the value being estimated, and the expected value of the estimate

$$\mathsf{B}(\hat{\boldsymbol{w}}) = \mathsf{E}[\hat{\boldsymbol{w}}] - \boldsymbol{w}$$

- An *unbiased* estimator has 0 bias

Variance of an estimate is, simply its variance, the value of the squared deviations from the mean estimate

$$\text{var}(\hat{\boldsymbol{w}}) = \mathsf{E}\left[(\hat{\boldsymbol{w}} - \mathsf{E}[\hat{\boldsymbol{w}}])^2\right]$$

$\boldsymbol{w}$ is the parameter (vector) that defines the model

Bias–variance relationship is a trade-off:
models with low bias result in high variance.

Bias–variance, underfitting–overfitting

- Bias and variance are properties of estimators
- We want estimators with low bias, low variance
- Complex models tend to overfit – and exhibit high variance
- Simple models tend to have low variance, but likely to have (high) bias

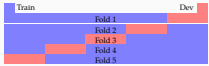
Model selection & hyperparameter tuning

- Our aim is to reduce the error on unseen data
- The evaluation practice should reflect that
- We can estimate the test error on a *development* set (*validation* or *held-out* data):
  - Split the data at hand as *training* and *development* set
  - Train alternative models (different hyperparameters) on the training set
  - Choose the model with best development set performance

Cross validation

- To avoid overfitting, we want to tune our models on a *development set*
- But (labeled) data is valuable
- Cross validation is a technique that uses all the data, for both training and tuning with some additional effort
- Besides tuning hyper-parameters, we may also want to get 'average' parameter estimates over multiple folds

K-fold Cross validation



- At each fold, we hold part of the data for testing, train the model with the remaining data
- The special case where k equal to the number of data points is called *leave-one-out cross validation*

The choice of k in k-fold CV

- Increasing k
  - reduces the bias: the estimates converge to true value of the measure (e.g., R^2) in the limit
  - increases the variance: smaller held-out sets produce more varied parameter estimates
  - is generally computationally expensive
- 5- or 10-fold cross validation is common practice (and found to have a good balance between bias and variance)

Comparing with a baseline

- The performance measures are only meaningful if we have something to compare against

random does the model do anything useful at all?
majority class does the classifier work better than predicting the majority class all the time?
state-of-the-art how does your model compare against known (non-trivial) models?

- In comparing different models we use another split of the data, *test set*
- Ideally test set is used only once – we want to avoid tuning the system on the test data
- Differences between models are exactly repeatable when the same test set is used (by different studies)
- Differences are reliable if your test set size is large enough
- Use statistical tests when comparing different models/methods

Summary

The first principle is that you must not fool yourself and you are the easiest person to fool. – Richard P. Feynman

- We want models with low bias and low variance
- Evaluating ML system requires special care:
  - Tuning your system on a development set
  - Cross-validation allows efficient use of labeled data during tuning
  - A test set is often used when comparing results obtained by different models

Next:

- Classification