DDA5001 Machine Learning

Cross Validation & Regularization

Zhongxiang DAI

School of Data Science
The Chinese University of Hong Kong, Shenzhen

Recap: Overfitting and its Catalysts

Overfitting

Fitting the data more than is needed

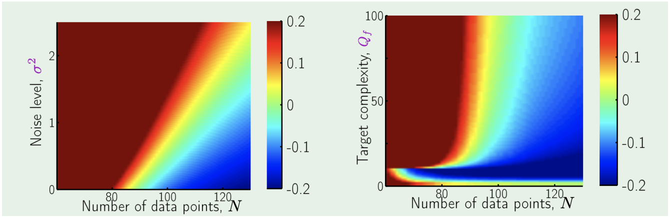


Figure: Color means overfitting level.

Overfitting: Catalysts

- **Number of training samples** increase, overfitting decreases.
- **Noise** in data increase, overfitting increases.
- **Model complexity** increases, overfitting increases.

Recap: Validation

Validation technique tries to estimate the out-of-sample error:

$$\underbrace{\mathrm{Er}_{\mathrm{out}}(f)}_{\text{validation estimates this quantity}} \leq \mathrm{Er}_{\mathrm{in}}(f) + \text{overfit penalty.}$$

Validation is used for model selection for avoiding overfitting.

The idea

Split the training set to another 'training set' and validation set.

Then, use the validation set for estimating $\mathrm{Er}_{\mathrm{out}}$.

Recap: Validation Error and Approximation of $\mathrm{Er}_{\mathrm{out}}$

Validation error:

$$\boxed{\mathrm{Er}_{\mathrm{val}}(f') = \frac{1}{k}\sum_{i=1}^{k} e(f'(\boldsymbol{x}_i), y_i)}$$

Estimate $\mathrm{Er}_{\mathrm{out}}$:

$$\mathrm{Er}_{\mathrm{out}}(f') \leq \mathrm{Er}_{\mathrm{val}}(f') + \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

Restoring: After we have used the validation set to estimate the out-of-sample error, re-train on the whole data set to get $\widehat{f}$. Using a reasonable guess from VC analysis, we have

$$\boxed{\mathrm{Er}_{\mathrm{out}}(\widehat{f}) \leq \mathrm{Er}_{\mathrm{out}}(f') \leq \mathrm{Er}_{\mathrm{val}}(f') + \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).}$$

Recap: Validation for Model Selection

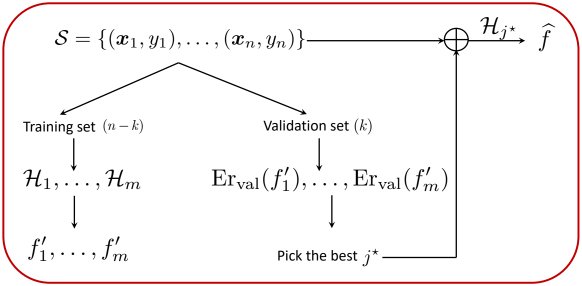
- Setup: Suppose we have m candidate hypothesis (can also be m different learning rate choices, etc)

  $$\mathcal{H}_1, \ldots, \mathcal{H}_m.$$

  We can use the validation set to estimate the out-of-sample error by using $\mathrm{Er}_{\mathrm{val}}(f'_j)$ for each f'_j learned from those model spaces.

- Selection: Choose $j^\star$ such that $\mathrm{Er}_{\mathrm{val}}(f'_{j^\star}) \leq \mathrm{Er}_{\mathrm{val}}(f'_j)$ for all j.

- Restoring: Train f on the whole set using model space $\mathcal{H}_{j^\star}$, get $\widehat{f}$.



$\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$
$\mathcal{H}_{j^\star}$
$\widehat{f}$
Training set $(n-k)$
Validation set (k)
$\mathcal{H}_1, \ldots, \mathcal{H}_m$
$\mathrm{Er}_{\mathrm{val}}(f'_1), \ldots, \mathrm{Er}_{\mathrm{val}}(f'_m)$
$f'_1, \ldots, f'_m$
Pick the best $j^\star$

Agenda

Validation vs. Testing

- We call this "validation", but how is it different from "testing"?
- Typically, validation is used to make learning choices, i.e., choosing hyper-parameters to avoid overfitting.

However,

The test data can never influence the training phase in any way.

If it impacts the learning process, i.e., which final $\widehat{f} \in \mathcal{H}$ we choose, then it is no longer a test set,

it becomes a validation set.

Validation Dilemma

Validation relies on the following chain of reasoning:

$$\mathrm{Er}_{\mathrm{out}}(\hat{f}) \underset{\substack{\big\uparrow \\ \text{small } k}}{\approx} \mathrm{Er}_{\mathrm{out}}(f') \underset{\substack{\big\uparrow \\ \text{large } k}}{\approx} \mathrm{Er}_{\mathrm{val}}(f')$$

- All we need to do is set k so that it is simultaneously small and large... A dilemma we face for choosing k.

Is it possible? Yes.

Cross Validation

Cross Validation: Leave One Out

- We need k to be small, so set $k = 1$.

$$\mathcal{S}^j_{\text{train}} = \{(\boldsymbol{x}_1, y_1), \ldots, \cancel{(\boldsymbol{x}_j, y_j)}, \ldots, (\boldsymbol{x}_n, y_n)\}$$

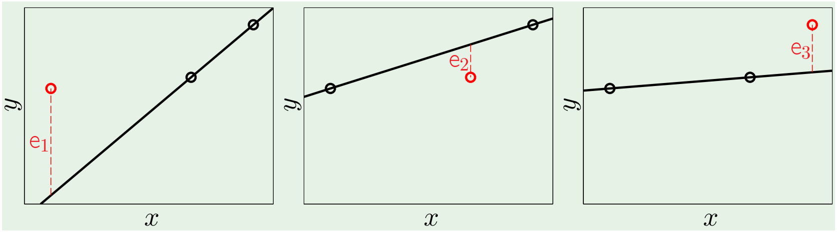
- Learn f'_j using $\mathcal{S}^j_{\text{train}}$. f'_j should has the almost the same quality as that of $\widehat{f}$.
- Validation error: $\mathrm{Er}_{\mathrm{val}}(f'_j) = e(f'_j(\boldsymbol{x}_j), y_j) := e_j$.
- Since $k = 1$, $\mathrm{Er}_{\mathrm{val}}(f'_j)$ is a terrible estimate of $\mathrm{Er}_{\mathrm{out}}(f'_j)$.

The idea: Repeat this for all possible choices of j, and then average them, giving the cross validation error:

$$\mathrm{Er}_{\text{cv}} = \frac{1}{n}\sum_{j=1}^{n} e_j.$$

This approach is called leave-one-out cross validation.

Cross Validation: Example



$$\mathrm{Er}_{\mathsf{cv}} = \frac{1}{3}(e_1 + e_2 + e_3)$$

- The hope is that the n validation errors together (i.e, $\mathrm{Er}_{\mathsf{cv}}$) is somehow equivalent to estimating $\mathrm{Er}_{\mathrm{out}}$ using the whole data set of size n, while at the same time train f_j' on $n-1$ data points.

Cross Validation for Model Selection

- Setup: Suppose that we have m candidate model spaces $\mathcal{H}_1, \ldots, \mathcal{H}_m$ (can also be m different learning rate choices, etc).
- We use cross validation to estimate $\mathrm{Er}_{\mathrm{out}}$ of $\mathcal{H}_i$ for $i = 1, \ldots, m$ by computing $\mathrm{Er}_{\mathsf{cv}}$ of using $\mathcal{H}_i$.
- Choose $i^\star$ that has the smallest $\mathrm{Er}_{\mathsf{cv}}$ over all i. Obtain $\widehat{f} = f'_{i^\star}$, as there is no need to do restoring (only one data point difference).

What is the potential drawback of leave-one-out cross validation?

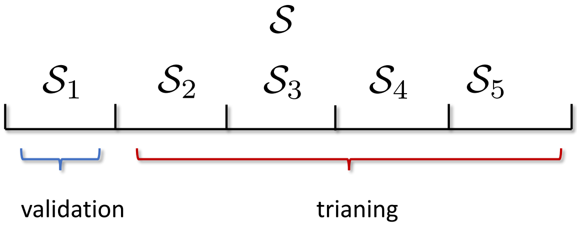
- For obtaining each $\mathrm{Er}_{\mathsf{cv}}$ of using $\mathcal{H}_i$, we need to train n times on $n-1$ samples each.
- In addition, for selecting the model $\mathcal{H}_{i^\star}$, we need to repeat it for m times, requiring around mn rounds of training on $n-1$ data points.
- When n is quite large, it can be computationally prohibitive.

Cross Validation: Leave More Out

k-fold cross validation: Choose a batch of data points for validation rather than one point.

In k-fold cross validation, k is the number of folds and $k' = \frac{n}{k}$ is the size of the validation set

Example: $k = 5$



- Iterate over all 5 choices of validation set and average. So, we only need to train k times on $n - \frac{n}{k}$ samples each.
- For cross validation with m hypothesis spaces, we need mk rounds of learning on $n - \frac{n}{k}$ samples each.

Common choice: $k = 5, 10$.

Leave More Out: Remarks

- For k-fold cross validation, the estimate depends on the particular choice of partition.
- When using k-fold cross validation for classification, one should ensure that each of the sets $\{\mathcal{S}_j\}$ contain training data from each class in almost the same proportion as in the full data set.
- It is common to form several estimates based on different random partitions.

Agenda

Regularization

- Validation is to estimate $\mathrm{Er}_{\mathrm{out}}$, and then adjust hyper-parameters.

The regularization is another weapon for eliminating overfitting, which penalizes the model complexity using penalty $\Omega(\mathcal{H})$:

$$\boxed{\mathrm{Er}_{\mathrm{out}}(f) \leq \mathrm{Er}_{\mathrm{in}}(f) + \Omega(\mathcal{H}), \quad \forall f \in \mathcal{H}}$$

- From VC analysis, it is better to fit the data using the simplest workable $\mathcal{H}$. However, it is hard to determine such a perfect $\mathcal{H}$.
- One further heuristic extrapolation: How about use a rich/complex enough $\mathcal{H}$, but choose a 'simple' (the simplest workable) f from $\mathcal{H}$. Thus, we can choose a penalty $\Omega(f)$ to penalize the complexity of an individual f.
- Instead of minimizing $\mathrm{Er}_{\mathrm{in}}(f)$ alone, regularization amounts to minimizing simultaneously the training error and the complexity penalty, i.e.,

$$\boxed{\min_{f \in \mathcal{H}} \ \mathrm{Er}_{\mathrm{in}}(f) + \Omega(f).}$$

Least Square Revisited

Learning problem for least squares

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\operatorname{argmin}} \ \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2,$$

where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$.

When the data matrix has full column rank, we have a unique closed-form solution for LS:

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

- How about $n < d$?
- What is such a case? Overfitting occurs as d begins to exceed the number of samples n. We will get zero training error, but large test error.

Regularization Technique I: ℓ_2-regularization

One candidate regularizer:

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2.$$

The ℓ_2-regularized LS is:

$$\boxed{\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^d} \ \|\boldsymbol{X\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2}$$

- The most direct way to reduce complexity is to let $\boldsymbol{\theta}$ have many zeros (recall its 'VC dimension' is d). But this is too hard a constraint. ℓ_2-regularizer is to make some parameters small (close to zero).
- $\lambda > 0$ is the regularization parameter (a hyper-parameter) that controls the trade-off between underfitting and overfitting.
  - Too large λ results in underfitting.
  - Too small λ may lead to overfitting.
- Validation technique can be used to choose this hyper-parameter.
- We can apply ℓ_2-regularization to logistic regression too.

Solution of ℓ_2-regularization

Let

$$\mathcal{L}(\boldsymbol{\theta}) = \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2$$

Expanding the ℓ_2-norms yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^\top(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}) + \lambda\boldsymbol{\theta}^\top\boldsymbol{\theta} \\ &= \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\theta}^\top(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbf{I})\boldsymbol{\theta} - 2\boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{y}\end{aligned}$$

Taking the gradient

$$\nabla\mathcal{L}(\boldsymbol{\theta}) = 2(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbf{I})\boldsymbol{\theta} - 2\boldsymbol{X}^\top\boldsymbol{y}$$

Setting the gradient to zero gives

$$\boxed{\widehat{\boldsymbol{\theta}} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbf{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}}$$

ℓ_2-regularization vs. Vanilla Least Squares

Least squares:

$$\begin{aligned}\widehat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}}\ \|\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}\|_2^2 \\ &= \left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y} - \text{ only for full column rank case}\end{aligned}$$

ℓ_2-regularization:

$$\begin{aligned}\widehat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}}\ \|\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2 \\ &= \left(\boldsymbol{X}^\top\boldsymbol{X}+\lambda\mathbf{I}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}\end{aligned}$$

The advantage of ℓ_2-regularization:

$$\underbrace{\left(\boldsymbol{X}^\top\boldsymbol{X}+\lambda\mathbf{I}\right)^{-1}}_{\text{always invertible}}$$

Algebraically explained why ℓ_2-regularization is useful.

Weight Decay

Weight decay is an important technique in machine learning. It is used almost everywhere in the training of neural networks.

- Weight decay is proposed as a technique for directly decaying the parameter (wight) $\boldsymbol{\theta}$ during the algorithm process. It has the form:

$$\boldsymbol{\theta}_{k+1} = (1-\lambda)\boldsymbol{\theta}_k - \mu \nabla \mathcal{L}(\boldsymbol{\theta}_k),$$

  where λ defines the rate of the weight decay per step.
- It is easy to see that if $1-\lambda \in (0,1)$, the weight parameter $\boldsymbol{\theta}$ is decaying at each iteration, thus the name weight decay.

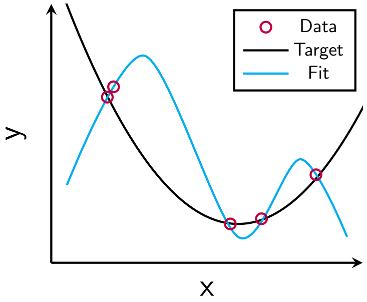
Indeed, we can verify that weight decay is equivalent to applying gradient descent to the ℓ_2-regularized problem:

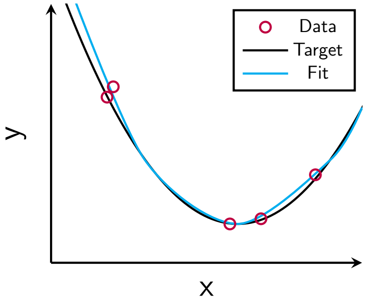
$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}) + \frac{\lambda'}{2}\|\boldsymbol{\theta}\|_2^2, \quad \text{with} \quad \lambda' = \frac{\lambda}{\mu}.$$

However, this is NOT the case in the Adam algorithm (later); see [1].

[1] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. ICLR 2019.

Regularization as a Cure for Overfitting





- Left: Using fourth-order polynomial without regularization.
- Right: Using fourth-order polynomial with regularization (weight decay).

$\rightsquigarrow$ Next lecture: Another regularization technique using ℓ_1-norm and concluding overfitting section.