Machine Learning: Generalization 3

Hao Tang

March 3, 2026

Fundamental theorem of statistical learning

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{finite}\\ \text{VC dimension}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Uniform}\\ \text{convergence}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{PAC}\\ \text{learnable}\end{array}}
\end{tikzcd}$$

VC generalization bounds

- With probability $1-\delta$, for all $h \in \mathcal{H}$

$$L_{\mathcal{D}}(h) \leq L_S(h) + 2\sqrt{\frac{8d\log(en/d) + 2\log(4/\delta)}{n}} \tag{1}$$

- d is the VC dimension.
- For linear classifiers $\mathcal{H}_{\text{lin}} = \{x \mapsto w^\top x : w \in \mathbb{R}^p\}$, VC-dim$(\mathcal{H}_{\text{lin}}) = p+1$.
- For multilayer perceptrons with p edges, VC-dim$(\mathcal{H}) = O(p\log p)$.

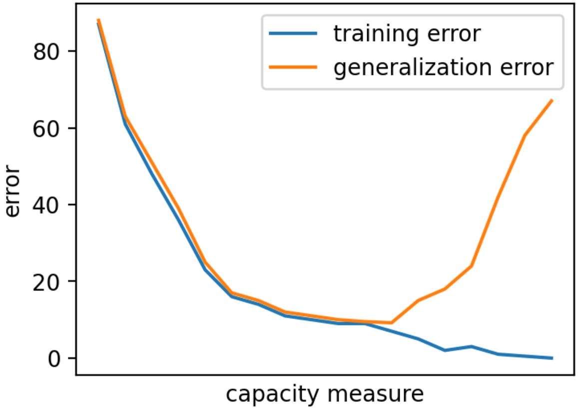
VC generalization bounds

- With probability $1 - \delta$, for all $h \in \mathcal{H}$

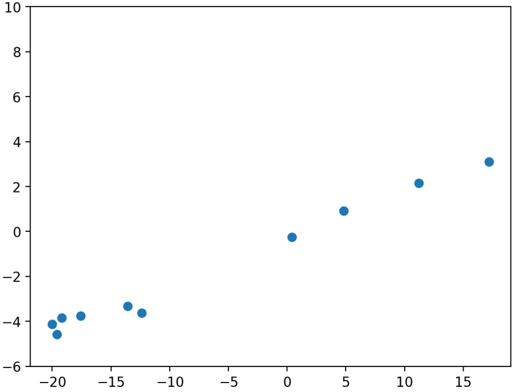
$$L_{\mathcal{D}}(h) \leq L_S(h) + 2\sqrt{\frac{8d \log(en/d) + 2 \log(4/\delta)}{n}} \tag{1}$$

- d is the VC dimension.
- For linear classifiers $\mathcal{H}_{\text{lin}} = \{x \mapsto w^\top x : w \in \mathbb{R}^p\}$, VC-dim$(\mathcal{H}_{\text{lin}}) = p + 1$.
- For multilayer perceptrons with p edges, VC-dim$(\mathcal{H}) = O(p \log p)$.
- The result is independent of how ERM is done and has little requirement on what L should be.

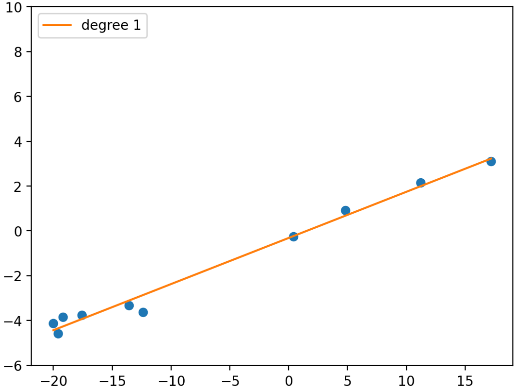
Capacity-generalization tradeoff



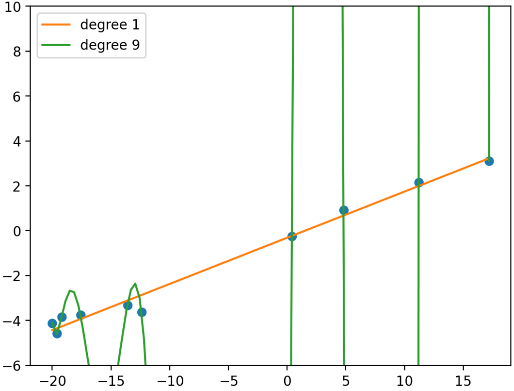
Capacity-generalization tradeoff



Capacity-generalization tradeoff



Capacity-generalization tradeoff



Optimization

- We can only do ERM for a limited number of cases, for example, $w = (X^\top X)^{-1} X^\top y$ in linear regression.
- Recall that the convergence of an optimization algorithm tells us how many iterations we need (how large t should be) to get to

$$L_S(h_t) - \min_{h \in \mathcal{H}} L_S(h) < \epsilon. \tag{2}$$

- But what is L_S?

Optimization

- We care about generalization of zero-one loss, not the cross entropy or the log likelihood.
- Cross entropy or the log likelihood are called **surrogate losses**.
- Surrogate losses are easier to optimize than the task loss, and usually have some connection to the task loss.
- For example, log loss is easier to optimize than zero-one loss, and is a smooth approximation of zero-one loss.

Error decomposition

- Optimization error
  - Mismatch between the surrogate loss and the task loss
  - Controlled by the optimization algorithm
- Estimation error
  - Controlled if we do ERM and have uniform convergence
  - Controlled by the capacity of $\mathcal{H}$ and the size of the training set
- Approximation error
  - Controlled by the capacity of $\mathcal{H}$

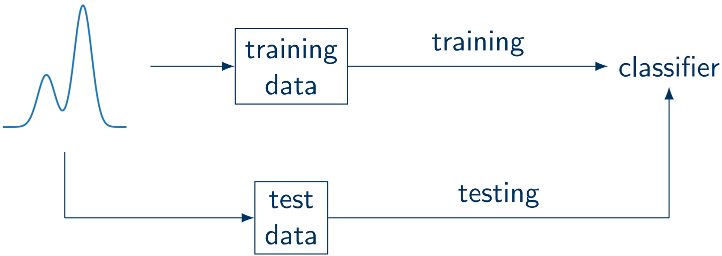
In practice

- We minimize a *surrogate* loss on the training set S, i.e., doing ERM.
- We can only do ERM approximately most of the time, because of optimization difficulty.
- Suppose training gives us $\hat{h}$.
- We use a test set S' and measure *task* loss $L_{S'}(\hat{h})$ to approximate generalization error.
- We hope $L_{\mathcal{D}}(\hat{h})$ is small when $L_{S'}(\hat{h})$ is small.

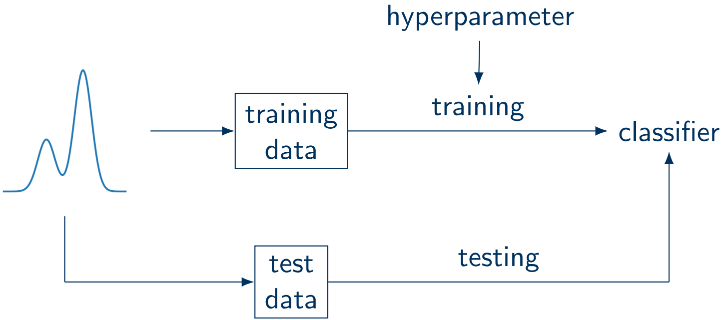
Test set

- Test error on a test set is used to approximate generalization error.
- Test set is supposed to be considered as an indepdent data drawn from the unknown distribution.
- Sometimes we have hyperparameters (not learned from data) we need to tune, for example, the step size in stochastic gradient descent.
- What's the problem of using the test set to tune hyperparameters?

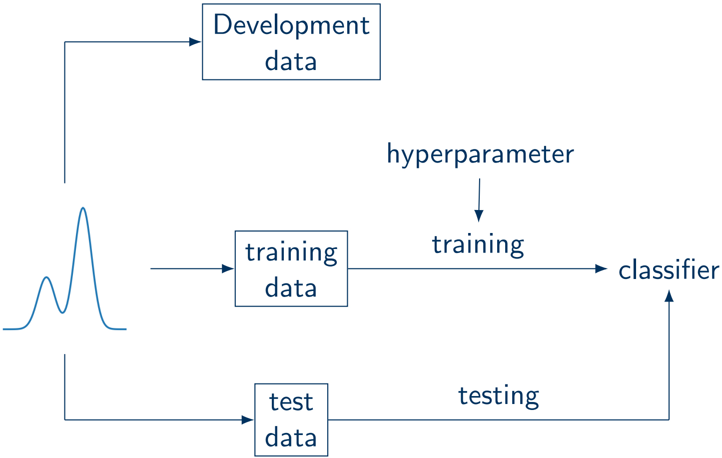
Generalization



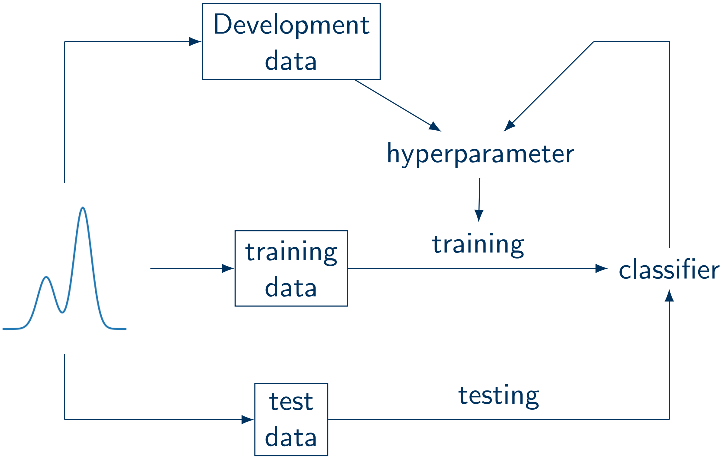
Generalization



Generalization



Generalization



Reusing test sets

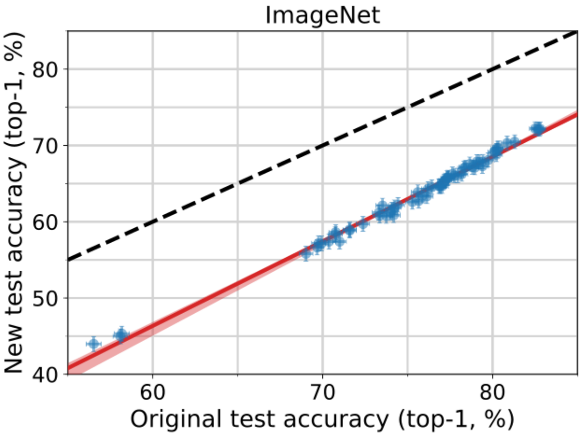


Image credit: (Recht *et al.*, 2019)

Underfitting

Underfitting

- A model is **underfitting** if there is another model that has a lower training.

Underfitting

- A model is **underfitting** if there is another model that has a lower training.
- A model h is underfitting if there is f such that $L_S(f) < L_S(h)$.
- The better f is unknown unless we find it.
- All models are underfitting with respect to ERM.
- When people say a model is underfitting, they simply mean there is room to improve the training error.

Overfitting

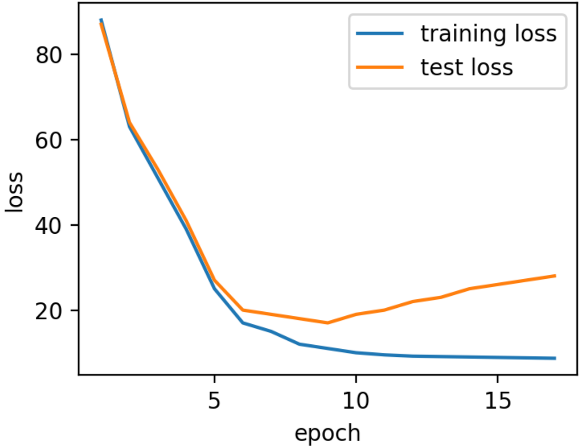
Overfitting

- A model is **overfitting** if there is another model that has a higher training error but a lower test eror.

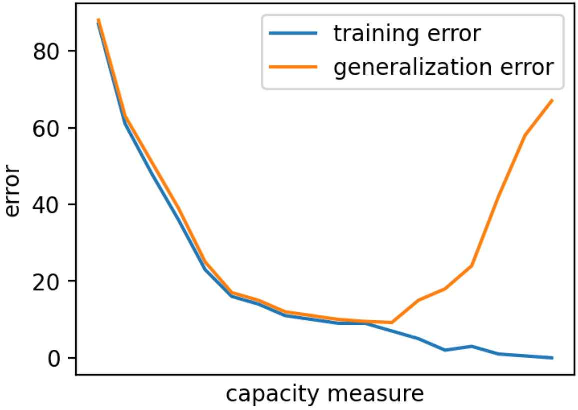
Overfitting

- A model is **overfitting** if there is another model that has a higher training error but a lower test eror.
- A model h is overfitting if there is f such that $L_S(f) > L_S(h)$ and $L_{S'}(f) < L_{S'}(h)$.
- The better f is unknown unless we find it.
- Models can overfit even when the gap $|L_S(h) - L_{S'}(h)|$ between training and test is not large.
- When people say a model is overfitting, they simply mean there is a large gap between the training and test error.

Overfitting



Capacity-generalization tradeoff



Large hypothesis classes

- Compare

$$\mathcal{H}_1 = \text{the set of two-layer neural networks with 512 hidden units} \tag{3}$$

$$\mathcal{H}_2 = \text{the set of all two-layer neural networks} \tag{4}$$

Large hypothesis classes

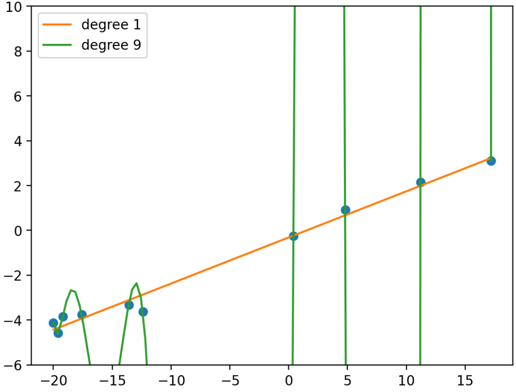
- Compare

$$\mathcal{H}_1 = \text{the set of two-layer neural networks with 512 hidden units} \tag{3}$$

$$\mathcal{H}_2 = \text{the set of all two-layer neural networks} \tag{4}$$

- $\mathcal{H}_1$ has a finite VC dimension, while the VC dimension of $\mathcal{H}_2$ is infinite!
- It is much easier (and tempting) to reduce the training error by increasing the hypothesis class.

Overfitting



Overfitting

- Compare

  $w_2 = [0.206, -0.317]$

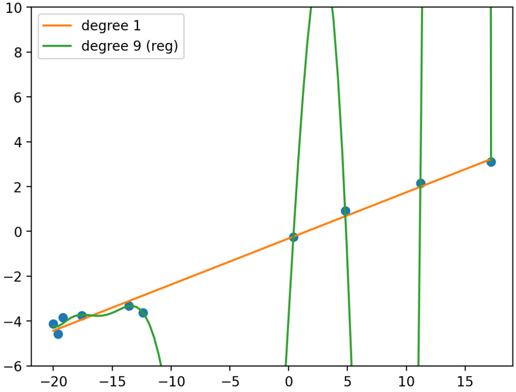
  $w_9 = [-30.69, 93.27, -2.65, -3.29, -0.124, 0.0248, 0.0017, 0.0000245,$
  $-0.00000423, -0.0000000857]$

- The learned weights are either too large or too small for degree 9.

- What if instead we optimize

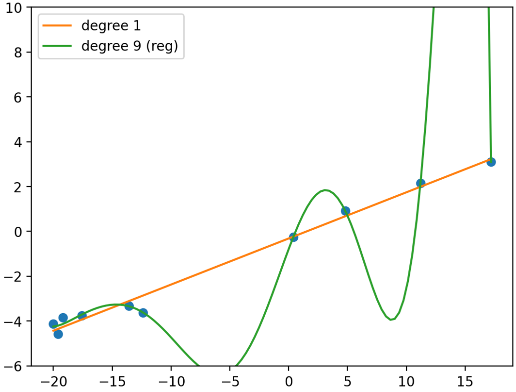
$$\min_{w \in \mathcal{H}} L_S(w) + \frac{\lambda}{2} \|w\|_2^2 \tag{5}$$

Regularization



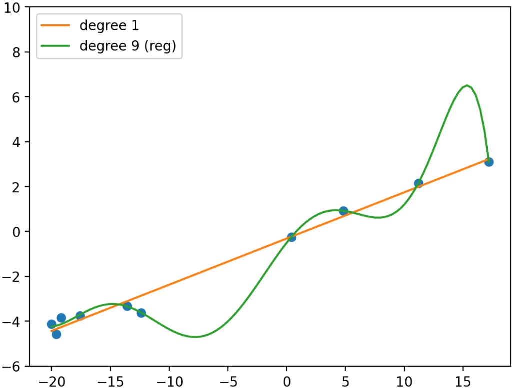
$$\lambda = 0.002$$

Regularization



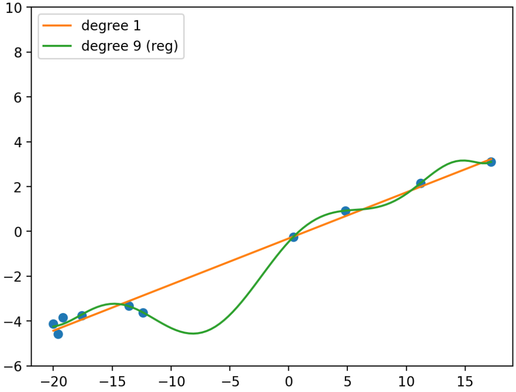
$$\lambda = 0.02$$

Regularization



$\lambda = 0.1$

Regularization



$\lambda = 0.2$

L_2 Regularization

- The term $\frac{\lambda}{2}\|w\|_2^2$ is called an L_2 regularizer.
- It is also known as weight decay.
- The expression

$$L_S(w) + \frac{\lambda}{2}\|w\|_2^2 \tag{6}$$

is the Lagrangian of

$$\min_w \quad L_S(w) \tag{7}$$

$$\text{s.t.} \quad \|w\|_2 \leq B \tag{8}$$

L_2 Regularization

- The L_2 regularizer has an effect of controlling the capacity of the hypothesis class.
- Compare

$$\mathcal{H} = \{x \mapsto w^\top x : w \in \mathbb{R}^d\} \tag{9}$$

$$\mathcal{H} = \{x \mapsto w^\top x : \|w\|_2 \leq B\} \tag{10}$$

Generalization bound for bounded linear classifier

- With probability $1-\delta$, for all $h \in \mathcal{H}$,

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{r^2 B^2}{n}} + 3\sqrt{\frac{\log(2/\delta)}{2n}}, \tag{11}$$

where $\|x\|_2 \leq r$ for any $x \in S$ and $\mathcal{H} = \{x \mapsto w^\top x : \|w\|_2 \leq B\}$.

An alternative explanation: Algorithmic stability

- A learning algorithm is **stable** if the learned program does not change much in performance when we change the data set slightly.
- The slight change in data set is by swapping out a data point.

$$S = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\} \tag{12}$$

$$S^i = \{(x_1, y_1), \ldots, (x', y'), \ldots, (x_n, y_n)\} \tag{13}$$

- A learning algorithm is stable is $A(S)$ and $A(S^i)$ is "similar," or

$$\ell(A(S)(x), y) - \ell(A(S^i)(x), y) \tag{14}$$

is small.

Stability

- Stable learning algorithms don't overfit.

$$\mathbb{E}_{S\sim\mathcal{D}^n}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] = \mathbb{E}_{\substack{i\sim U(n)\\ S\sim\mathcal{D}^n\\ (x,y)\sim\mathcal{D}}}[\ell(A(S^i)(x_i), y_i) - \ell(A(S)(x_i), y_i)] \tag{15}$$

- It can be shown that

$$\underset{w}{\operatorname{argmin}}\, L_S(w) + \frac{\lambda}{2}\|w\|_2^2 \tag{16}$$

is stable and hence does not overfit.