Generalization

Reading

- Chapters 7.1, 7.2
  - Hastie, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009.
  - Available online: https://hastie.su.domains/Papers/ESLII.pdf
- Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning (still) requires rethinking generalization." *Communications of the ACM* 64, no. 3 (2021): 107-115.

Overview

Rensselaer

- Generalization is a central concept in all areas of ML
- Just because your model works on your training data doesn't mean it will work on your test data
  - One can go further: even if your model works on your test data, it doesn't mean it will work on new test data
    - But that's "better" evidence than working on training data
- Generalization is particularly important in deep learning
  - Neural networks can overfit any dataset we currently have
  - Users need to always be careful about generalization
- We'll discuss how to estimate generalization error and what makes a model more likely to overfit

Training vs Test Data

Rensselaer

- Almost any supervised ML task involves the collection of data
- Typically, once the data is collected, we split it into 3 sets:
  - Training set
  - Validation set
  - Test set
- Historically, datasets weren't large enough for such a split, so researchers had to develop other techniques
  - E.g., cross validation
- But the end goal is the same
  - If we develop a model based on the data we have, how well does this model perform on new data?

Generalization error

- Let D_{tr}, D_v, D_{te} be the training, validation and test sets, respectively
- Each of those sets is drawn IID from the same distribution $\mathcal{D}$
- The error of a model f on a dataset D is defined as:

$$Err_D(f) = \frac{1}{|D|} \sum_{(\boldsymbol{x},y)\in D} \boldsymbol{I}(f(\boldsymbol{x}) \neq y)$$

  - Where $\boldsymbol{I}$ is the indicator function
    - $\boldsymbol{I}(f(\boldsymbol{x}) \neq y) = 1$ when $f(\boldsymbol{x}) \neq y$ and 0, otherwise
- The generalization error of a model f is defined as:

$$GE(f) = \mathbb{P}[f(\boldsymbol{X}) \neq Y]$$

  - where $(\boldsymbol{X}, Y) \sim \mathcal{D}$
  - i.e., it is the probability of making an error on unseen data

Generalization Error, cont'd

Rensselaer

- An unbiased estimate of the generalization error is the classifier's performance on the test set:

$$\widehat{GE}(f) = Err_{D_{te}}(f)$$

- An estimator is said to be unbiased if its expected value is equal to the quantity it is trying to estimate
  - E.g., $\mathbb{E}\left[Err_{D_{te}}(f)\right] = GE(f)$
  - Why?
  - Consider any random test set D
  - All $(\boldsymbol{x}_i, y_i) \in D$ are drawn IID from some distribution $\mathcal{D}$
    - Each corresponds to a random variable $(\boldsymbol{X}_i, Y_i)$

Generalization Error, cont'd

- An unbiased estimate of the generalization error is the classifier's performance on the test set:

$$\widehat{GE}(f) = Err_{D_{te}}(f)$$

- An estimator is said to be unbiased if its expected value is equal to the quantity it is trying to estimate
  - E.g., $\mathbb{E}\left[Err_{D_{te}}(f)\right] = GE(f)$:

$$\mathbb{E}[Err_D(f)] = \frac{1}{|D|}\sum_i \mathbb{E}[\boldsymbol{I}(f(\boldsymbol{X}_i) \neq Y_i)]$$

$$= \frac{1}{|D|}\sum_i \mathbb{P}[f(\boldsymbol{X}_i) \neq Y_i] = \frac{1}{|D|}\sum_i \mathbb{P}[f(\boldsymbol{X}_1) \neq Y]$$

$$= \frac{|D|}{|D|}\mathbb{P}[f(\boldsymbol{X}_1) \neq Y]$$

Generalization Error, cont'd

Rensselaer

- Being unbiased is a nice property but it's not enough
  - The test error on a single point is also an unbiased estimate
  - But if we do well on a larger test set, that is better than doing well on a smaller test set!
  - We can use the Law of Large Numbers and Hoeffding's inequality to further analyze our model's performance

Probability Aside: Law of Large Numbers

- Let $X_1, \ldots, X_n$ be n IID random variables
- Let $S_n = X_1 + \cdots + X_n$
- (Weak) Law of Large Numbers:

$$\mathbb{P}\left[\left|\frac{S_n}{n} - \mathbb{E}[X_1]\right| < \epsilon\right] \to 1 \text{ as } n \to \infty$$

  - for any positive ϵ
- As we collect more data, the sample mean S_n/n converges to the expected mean $\mathbb{E}[X_1]$
  - Since the X_i are IID, $\mathbb{E}[X_1] = \mathbb{E}[X_i]$ for any i

The Benefit of the Law of Large Numbers

- What is the benefit of the Law of Large Numbers?

$$\mathbb{P}\left[\left|\frac{S_n}{n} - \mathbb{E}[X_1]\right| < \epsilon\right] \to 1 \text{ as } n \to \infty$$

- Think of $\frac{S_n}{n}$ as your model's test error

$$Err_D(f) = \frac{1}{|D|}\sum_{(\boldsymbol{x},y)\in D} \boldsymbol{I}(f(\boldsymbol{x}) \neq y)$$

  - Here, $S_n := \sum_{(\boldsymbol{x},y)\in D} \boldsymbol{I}(f(\boldsymbol{x}) \neq y)$ and $n := |D|$
  - So the test error converges to the true expected error as the test set gets large

- Practically speaking, the larger the dataset the better
  - E.g., if your model achieves good accuracy on a large test set, then it will likely work well on new data also

Probability Aside: Hoeffding's Inequality

Rensselaer

- Suppose your model has good accuracy on a test set
  - Is it possible that you just got lucky and your model isn't that great after all?
- Let $X_1, \ldots, X_n$ be n independent random variables
  - Each bounded by $a_i \leq X_i \leq b_i$
- Let $S_n = X_1 + \cdots + X_n$
- Hoeffding's Theorem:

$$\mathbb{P}[S_n - \mathbb{E}[S_n] \geq t] \leq \exp\left\{-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right\}$$

- Two-tailed version:

$$\mathbb{P}[|S_n - \mathbb{E}[S_n]| \geq t] \leq 2\exp\left\{-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right\}$$

Probability Aside: Hoeffding's Inequality, cont'd

Rensselaer

- Hoeffding's Theorem:

$$\mathbb{P}[S_n - \mathbb{E}[S_n] \geq t] \leq \exp\left\{-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right\}$$

- A type of concentration bound
- Given a sample S_n, we can bound its deviation from the true mean
- The larger t is, the higher the probability the mean is within t of the sample
- The smaller the bounds $(b_i - a_i)$, the tighter the bound on S_n

Probability Aside: Hoeffding's Inequality, cont'd

Rensselaer

- Suppose each X_i is Bernoulli, i.e., $X_i \in \{0,1\}$, i.e., $b_i - a_i = 1$
  - E.g., X_i denotes correct or wrong classification on example i
- The bound simplifies to:

$$\mathbb{P}[S_n - \mathbb{E}[S_n] \geq t] \leq \exp\left\{-\frac{2t^2}{n}\right\}$$

- Furthermore, suppose we are interested in bounding the mean
  - E.g., your model's accuracy

$$\mathbb{P}\left[\frac{1}{n}(S_n - \mathbb{E}[S_n]) \geq t\right] =$$
$$= \mathbb{P}[S_n - \mathbb{E}[S_n] \geq nt] \leq \exp\{-2t^2 n\}$$

- For fixed t, the sample mean is less likely to be farther from the expected mean as we collect more data

Importance of the test set accuracy

Rensselaer

- Suppose we have trained a model f
- As far as the test set is concerned, f is now just a function
- Suppose the test set is $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
  - In other words, we have realizations of IID variables $(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)$
- Define the variable $Z_i = 1$ if $f(\boldsymbol{X}_i) = Y_i$ and 0, otherwise
  - Then the Z_i are IID Bernoulli variables
  - The expected value $\mathbb{E}[Z_i]$ is the true accuracy of f
  - The sample mean $\frac{1}{n}\sum_i z_i$ is the accuracy of f on the test set
- How can we bound $\mathbb{E}[Z_i]$ in terms of $\frac{1}{n}\sum_i z_i$?
  - We can directly apply Hoeffding's inequality on the test set

Example: Hoeffding's Inequality

- Suppose our model achieves a test accuracy of 80% over 1000 datapoints
  - What's the probability the true accuracy is less than 70%?
  - Note that $\mathbb{P}\left[\frac{1}{n}(\mathbb{E}[S_n] - S_n) \leq -t\right] = \mathbb{P}\left[\frac{1}{n}(S_n - \mathbb{E}[S_n]) \geq t\right]$
  - Then, using Hoeffding's inequality:

$$\mathbb{P}\left[\frac{1}{n}(S_n - \mathbb{E}[S_n]) \geq t\right] =$$

$$\mathbb{P}\left[\frac{1}{n}(S_n - \mathbb{E}[S_n]) \geq 0.1\right] \leq$$

$$\exp\{-2 * 0.1^2 * 1000\} \approx 2 * 10^{-9}$$

- Even 1000 points give us strong probabilistic guarantees

What about the training set?

- Can we apply Hoeffding's inequality to the training set?
  - Suppose training set is $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
  - Let z_i be the same as before
  - The z_i are no longer independent!
    - f is function of all $(\boldsymbol{x}_i, y_i)$, so the $f(\boldsymbol{x}_i)$ are not independent
- Intuitively, it makes sense that we can't evaluate our model on the training data
  - As with any training task, you eventually remember the task too well (you overfit!)
- There are some cases where we can bound the test set performance in terms of training set performance
  - VC dimension!

What about the validation set?

- In theory, the model f is only trained on the training set
- In practice, we choose different hyper-parameters of f and iterate the training process
  - E.g., number of neighbors in KNN
  - After each iteration, we evaluate the model's accuracy on the validation set only (not the test set!)
  - Why?
  - We can overfit the hyper-parameter values also
- Once we train a good model, we evaluate on the test set
  - If there is a big difference between the test and validation sets, then overfitting is likely to blame

Cross Validation

Rensselaer

- When the test set is not large (a few dozen examples), Hoeffding's inequality provides loose bounds
- Cross validation very useful in this case
  - Split the data randomly into 90% training and 10% testing
  - Train on the training data and record the test accuracy
  - Repeat multiple (e.g., 10) times
  - Take the average test error over all runs
  - A better estimate of generalization error than a single split
- Most modern datasets are big enough such that this is no longer an issue
  - Cross validation is still useful but is not commonly used since it's quite computationally expensive

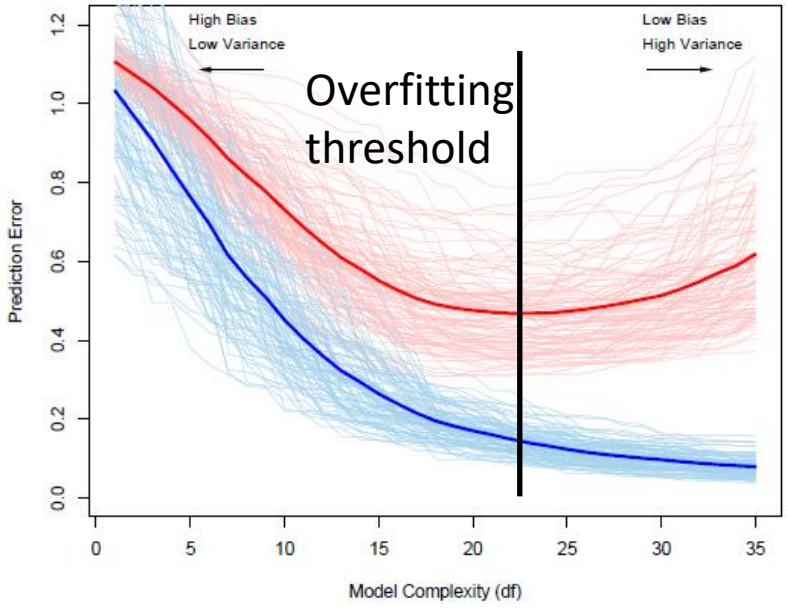
Bias, Variance, and Model Complexity

Rensselaer

- We've already seen examples of models that perfectly overfit the training data without having any generalization capacity
  - E.g., a table with rules
- Turns out this is a general phenomenon that has to do with a model's complexity
  - The more complex a model is, the easier it is to achieve zero training error
  - However, it is also easier to overfit

Model Complexity vs Generalization

- Typically, there exists a point beyond which increasing the model complexity does not bring any generalization benefits



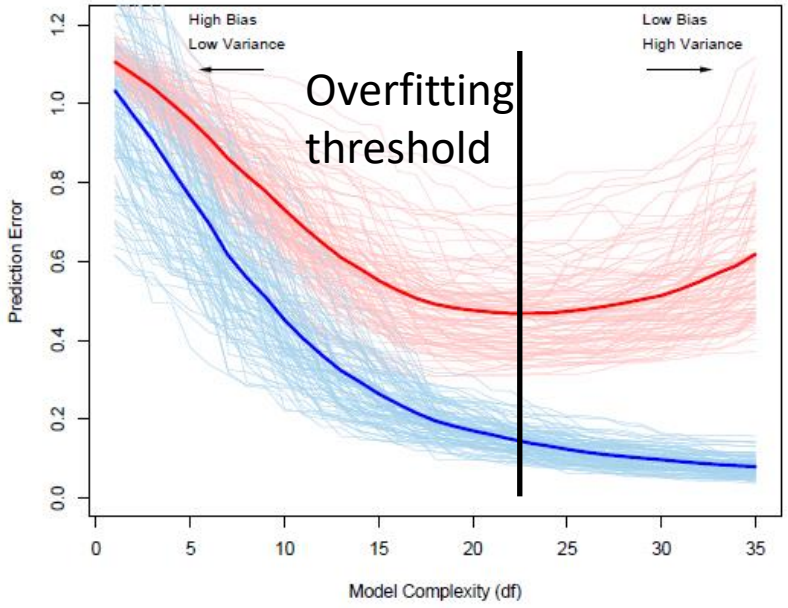
Test error

Train error

  - Book authors trained a LASSO algorithm on simulated data
  - LASSO is a more sophisticated regression technique
    - See book if you are interested

Model Complexity vs Generalization

- Typically, there exists a point beyond which increasing the model complexity does not bring any generalization benefits



Test error

Train error

  - As the model complexity is increased:
    - Train error (bias) decreases, but eventually test error starts increasing (overfitting!)
    - Test error variance increases; models are sensitive to training noise

Understanding deep learning (still) requires rethinking generalization

Overview

Rensselaer

- This paper is actually an updated version of a 2017 paper with the same name (without the ‘still’)
- The authors show that generalization is not a well understood notion
  - In classical learning theory, good performance on the training set should lead to similar performance on the test set (when overfitting precautions have been made)
  - This paper shows that this need not be the case
- The notion of a “distribution” is really not well defined (at least in the case of images)

Theoretical Bounds on Generalization Error

Rensselaer

- The classical approach is to quantify the classifier's expressive power (also known as capacity)
- Intuitively, argument works as follows:
  - Suppose you have a simple classifier and you have correctly classified a "large" training set
    - "Simple" as measured through a complexity measure such as VC dimension or Rademacher complexity
  - Chances are you'll correctly classify new points also (you've already seen a large chunk of the distribution)
- Traditional generalization error arguments don't work for NNs
  - In some ways, it is surprising that they generalize at all
    - There are many methods to fit the training data that don't generalize

Random Labels

- First experiment in the paper
- Randomly shuffle all labels
  - By design, generalization isn't possible
- One way of assessing the NN capacity
  - Is it able to learn (i.e., memorize) even the shuffled labels?
  - Is the learning going to slow down or be otherwise adversely affected by the irregular training set?

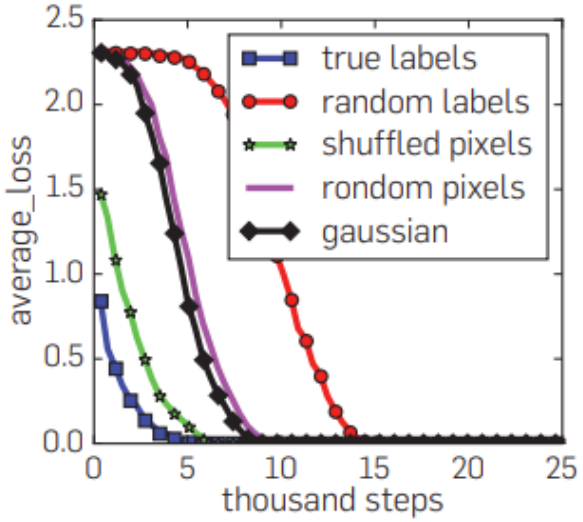
Random Pixels and Shuffled Pixels

Rensselaer

- Second set of experiments
- Random pixels: keep the original labels, but replace all pixels with random noise
  - Once again, generalization isn't possible
- Shuffled pixels: keep the original labels, but shuffle pixels using the same transformation for all images
  - Depending on the transformation, this may add little to significant noise

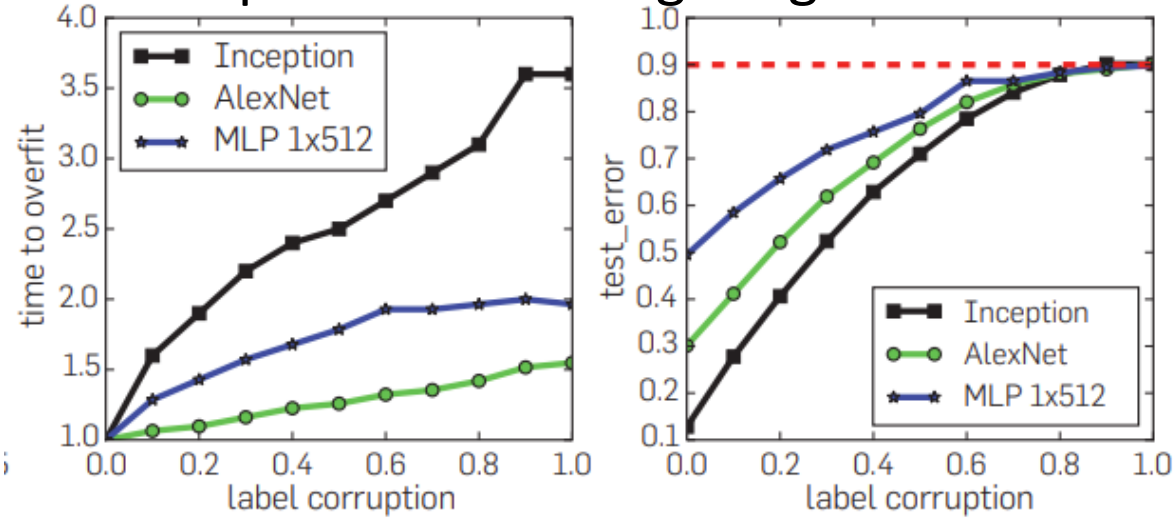
Training Results, CIFAR10

- In all cases, the NN is able to memorize the entire training set!
- Training with random labels takes the longest but it still converges to 0 loss
  - Training with random pixels is faster probably because the data is more separated in space due to the noise



Label corruption, CIFAR10

- In order to assess the effect of label corruption on training and generalization, the authors also try corrupting a fraction of the labels
  - Ranging from 0% to 100% of all labels are corrupted
- Higher label corruption makes it significantly harder to overfit the training set
- Higher label corruption leads to higher generalization error



Training/Test Results Summary, CIFAR10

Rensselaer

- All sufficiently large models can perfectly overfit training data
- Regularization improves generalization
  - But not necessary or sufficient
  - Major overfitting even with generalization
- Both convolution and fully-connected NNs show the same trends

Table 1. The training and test accuracy (in %) of various models on the CIFAR10 dataset.

Model	# params	Random crop	Weight decay	Train accuracy	Test accuracy
Inception	1,649,402	Yes	Yes	100.0	89.05
		Yes	No	100.0	89.31
		No	Yes	100.0	86.03
		No	No	100.0	85.75
(fitting random labels)		No	No	100.0	9.78
Inception w/o BatchNorm	1,649,402	No	Yes	100.0	83.00
		No	No	100.0	82.00
(fitting ran-dom labels)		No	No	100.0	10.12
Alexnet	1,387,786	Yes	Yes	99.90	81.22
		Yes	No	99.82	79.66
		No	Yes	100.0	77.36
		No	No	100.0	76.07
(fitting ran-dom labels)		No	No	99.82	9.86
MLP 3 × 512	1,735,178	No	Yes	100.0	53.35
		No	No	100.0	52.39
(fitting random labels)		No	No	100.0	10.48
MLP 1 × 512	1,209,866	No	Yes	99.80	50.39
		No	No	100.0	50.51
(fitting random labels)		No	No	99.34	10.61

Training/Test Results Summary, ImageNet

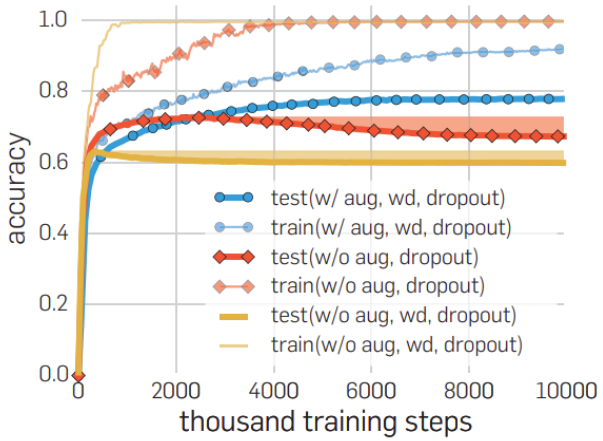
- Similar to CIFAR10
  - Overfitting only 95% of training data (still very surprising!)
- Regularization helps generalization
  - But once again not necessary or sufficient (still major overfitting even with regularization)

data aug	dropout	weight decay	top-1 train	top-5 train	top-1 test	top-5 test
ImageNet 1000 classes with the original labels						
yes	yes	yes	92.18	99.21	77.84	93.92
yes	no	no	92.33	99.17	72.95	90.43
no	no	yes	90.60	100.0	67.18 (72.57)	86.44 (91.31)
no	no	no	99.53	100.0	59.80 (63.16)	80.38 (84.49)
Alexnet (Krizhevsky et al., 2012)			-	-	-	83.6
ImageNet 1000 classes with random labels						
no	yes	yes	91.18	97.95	0.09	0.49
no	no	yes	87.81	96.15	0.12	0.50
no	no	no	95.20	99.14	0.11	0.56

Table 2 shows the performance on Imagenet with true labels and random labels, respectively.

Role of Regularization

- Remember Occam's Razor
  - Usually want the simplest model that can learn the task
  - This is what regularization tries to achieve
- Standard deep learning regularization techniques (dropout, weight decay, batch normalization) do not prevent overfitting in CIFAR10
  - Only works for some smaller models (AlexNet) on ImageNet



(a) Inception on ImageNet

(b) Inception on CIFAR10

NN Expressivity

Rensselaer

- Turns out that it doesn't take a very large NN to perfectly overfit a given training set
- **Theorem**: Given a training set $S = \{(x_1 y_1), \ldots, (x_n, y_n)\}$ of size n, where each $x_i \in \mathbb{R}^d$, there exists a 2-layer NN with ReLU activations and $2n + d$ weights that can perfectly overfit S.
- Proof is not very hard
- This means that even very high-dimensional datasets can be overfit with small NNs
  - Hence we need to rethink generalization

Conclusion

Rensselaer

- Generalization is one of the most important aspects of ML
- It is especially important for expressive models such as neural networks where overfitting is very easy
- The most robust method of establishing your model's generalization performance is through a test set
  - The larger and more diverse, the better!