CSDS503 / COMP552 – Advanced Machine Learning

Faizad Ullah

Loss Function

Loss functions

- Calculate the average error of h in predicting y.
- Smaller is better
  - 0 loss: No error
  - 100% loss: Could not even get one instance right
  - 50% loss: Your h is as informative as a coin toss

0/1 Loss

$$L_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n} \delta_{h(x_i)\neq y_i}, \text{where } \delta_{h(x_i)\neq y_i} = \begin{cases} 1, if\ h(x_i) \neq y_i \\ 0, otherwise \end{cases}$$

- Counts the average number of mistakes in predicting y
- Returns the training error rate
- Used to evaluate classifiers in binary/multiclass settings

Squared loss

$$L_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$$

- Typically used in regression settings
- The loss is always non-negative
- The loss grows quadratically
- If a prediction is very close to be correct, the square will be tiny

Absolute loss

$$L_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n}|h(x_i) - y_i|$$

- Typically used in regression settings
- The loss is always non-negative
- The loss grows linearly
- Better suited for noisy data

Comparison

y	h(x)	Square loss	Abs loss
100.00	101.00	1.00	1.00
90.00	90.01	0.0001	0.01
100.00	200.00	10,000.00	100.00
100.00	1,000.00	810,000.00	900.00
Overall		**205,000.25**	**250.25**

y	h(x)	Square loss	Abs loss
100.00	101.00	1.00	1.00
90.00	91.00	1.00	1.00
100.00	101.00	1.00	1.00
20.00	21.00	1.00	1.00
30.00	29.00	1.00	1.00
40.00	41.00	1.00	1.00
30.00	31.00	1.00	1.00
10.00	11.00	1.00	1.00
12.00	13.00	1.00	1.00
16.00	17.00	1.00	1.00
100.00	1,000.00	810,000.00	900.00
Overall		**73,637.27**	**82.73**

y	h(x)	Square loss	Abs loss
100.00	0.00	10,000.00	100.00
90.00	0.00	8,100.00	90.00
100.00	0.00	10,000.00	100.00
20.00	0.00	400.00	20.00
30.00	0.00	900.00	30.00
40.00	0.00	1,600.00	40.00
30.00	0.00	900.00	30.00
10.00	0.00	100.00	10.00
12.00	0.00	144.00	12.00
16.00	0.00	256.00	16.00
1,000.00	1,000.00	0.00	0.00
Overall		**2,945.45**	**40.73**

The elusive h

$$\boldsymbol{h = argmin_{h \in H} L(h)}$$

- So, we need an h with a low loss on D?

Reducing Loss

- How about reducing the loss like this?

$$h(x) = \begin{cases} y_i, & if\ \exists (x_i, y_i) \in D, \quad s.t. \quad x = x_i \\ 0, & otherwise \end{cases}$$

- What would be the loss of this h on the training set?
- What would be the loss of this h on an unseen test set?

The memorizer!

- Why is it bad?
- How to prevent this from happening?

Generalization

$$\epsilon = E_{(x,y)\sim P}[l(x,y)|h]$$

- That the expected loss should be calculated on any data point sampled from the distribution P, not necessarily those present in D
- How to get a new datapoint $x, y \sim P$?
- All we have are the n data points!
- We estimate ϵ by splitting the D into Train and Test sets.
- We train on D_{TR} and test on D_{TE} only once!
- Don't train on test inadvertently! (e.g., repeated testing)

Data Splits

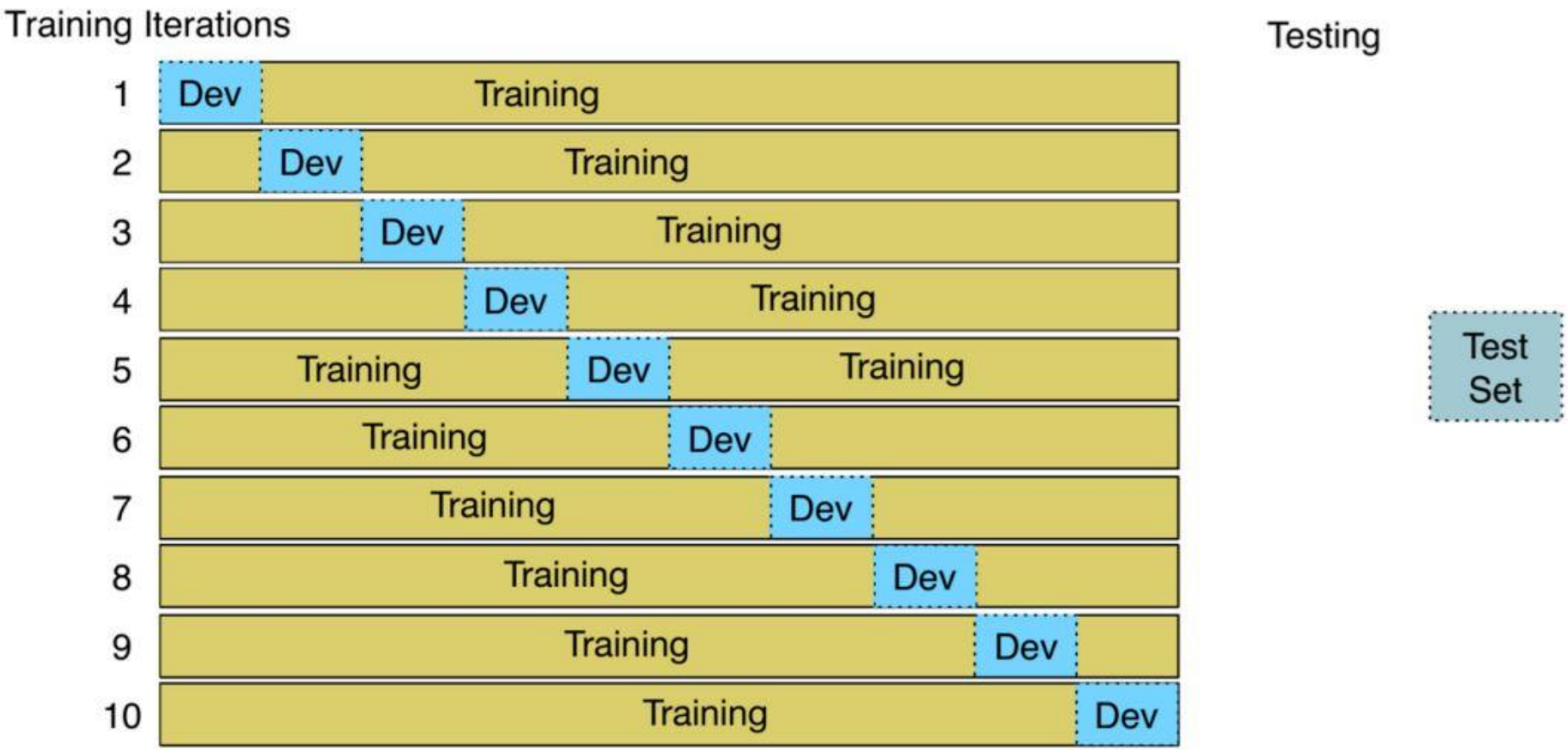
Data Splits

- Usually 80:20 or 70:30 splits
- Making sure the splits make sense
- Time series: Split by time
- i.i.d: Uniformly at random
  - Make sure you don't split the same datapoint between DTR and DTE
  - Make sure the same data does not get repeated on both sides e.g. spam
- We train only using the DTR and only use the DTE once
- Then the test error approximates the generalization loss
- **How do we evaluate the model, if we do not have access to the test data while training?**

Validation Set

70% **10%** **20%**

K-fold Cross Validation



References

- Murphy Chapter 1
- Alpaydin Chapter 1
- TM Chapter 1
- Lectures of Andrew Ng., Dr. Ali Raza, and “Machine Learning for Intelligent Systems (CS4780/CS5780)”, Kilian Weinberger.
- This disclaimer should serve as adequate citation.