

# CSC 561: Neural Networks and Deep Learning

Loss, Overfitting, Model Selection

Marco Alvarez

Department of Computer Science and Statistics  
University of Rhode Island

Spring 2024



## Loss functions

### The “supervised learning” problem

- Finding a **hypothesis** (classifier/regressor) that best approximates a **target function**  $f(\mathbf{x})$ 
  - ✓ the target function maps inputs  $\mathbf{x}$  to outputs  $y$  from a **data-generating** distribution  $P$

for  $h_w \in \mathcal{H}$  and  $\forall (x_i, y_i) \sim P$ , we want  $h_w(\mathbf{x}_i) \approx f(\mathbf{x}_i)$

Machine Learning algorithms use **search** and **optimization methods for finding**  $h_w$

### Empirical risk minimization

- **True risk** (a.k.a. expected loss, cost function)

$$J^*(\mathbf{w}) = \mathbb{E} \left[ l(h_w, \mathbf{x}_i, y_i) \right]_{(\mathbf{x}_i, y_i) \sim P}$$

- ✓ where  $l$  is the **per-example loss** (“error”) between the predicted output  $h_w(\mathbf{x}_i)$  and the target output  $y_i$

Note that the expectation is taken across the data-generating distribution  $P$  rather than over a finite training set.

## Empirical risk minimization

### • Empirical risk

- ✓ we can't solve the true risk directly as we do not know  $P$  but only have a training set  $\mathcal{D}$  of samples

$$\mathbb{E} \left[ l(h_w, \mathbf{x}_i, y_i) \right]_{(\mathbf{x}_i, y_i) \sim \mathcal{D}} = \frac{1}{n} \sum_{i=1}^n l(h_w, x_i, y_i) = J(\mathbf{w})$$

Rather than minimizing the **true risk** directly, we optimize the **empirical risk** hoping that the **true loss** decreases significantly as well.

5

## Empirical risk minimization

- The goal of a machine learning algorithm (supervised learning) is to reduce the **true risk**
  - ✓ as the true distribution  $P$  is unknown we can minimize the empirical risk instead
- The training process based on minimizing the **empirical risk** is known as **empirical risk minimization**

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} J(\mathbf{w})$$

6

## 0/1 loss

$$l_{0/1}(h_w, \mathbf{x}_i, y_i) = I(h_w(\mathbf{x}_i) \neq y_i)$$

indicator  
function

Prediction	Target
5	5
1	9
2	2
7	7
8	0
0	0
0	8
3	3
6	6
4	4

Empirical risk?

7

## Practice

X0	X1	X2	Y
1	0	0	-1
1	1	0	+1
1	1	1	+1
1	0	1	+1

$$h_w(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

$$\sigma(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z \leq 0 \end{cases}$$

zero-one loss for  $\mathbf{w}_a = [0,0,0]^T$ ?

zero-one loss for  $\mathbf{w}_b = [0,1,0]^T$ ?

zero one loss for  $\mathbf{w}_c = [-1,2,2]^T$ ?

8

## Squared loss

$$l_{sq}(h_w, \mathbf{x}_i, y_i) = (h_w(\mathbf{x}_i) - y_i)^2 \text{ penalizes big mistakes}$$

Prediction	Target
1.2	1.4
2.3	2.3
1.1	1.2
3.4	4.1
2.3	2.5
1.1	1.1
2.5	2.6
3.1	3.2
1.7	1.8
2.3	2.3

Empirical risk?

9

## Absolute loss

$$l_{abs}(h_w, \mathbf{x}_i, y_i) = |h_w(\mathbf{x}_i) - y_i|$$

Prediction	Target
1.2	1.4
2.3	2.3
1.1	1.2
3.4	4.1
2.3	2.5
1.1	1.1
2.5	2.6
3.1	3.2
1.7	1.8
2.3	2.3

Empirical risk?

10

## Remarks

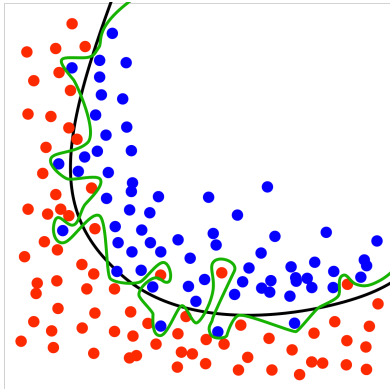
- Empirical risk minimization is prone to **overfitting**
  - ✓ high-capacity models can simply memorize the training set
  - ✓ can introduce regularization to improve generalization
- Modern machine learning is based on **gradient descent**
  - ✓ requires loss functions to be differentiable (e.g., can't use 0/1 loss)
- In the context of deep learning, we must go beyond pure empirical risk minimization
  - ✓ the quantity that we actually optimize may differ from the quantity we truly want to optimize
  - ✓ e.g. SGD does not directly optimize the empirical risk, rather a loss function that approximates it

11

# Overfitting

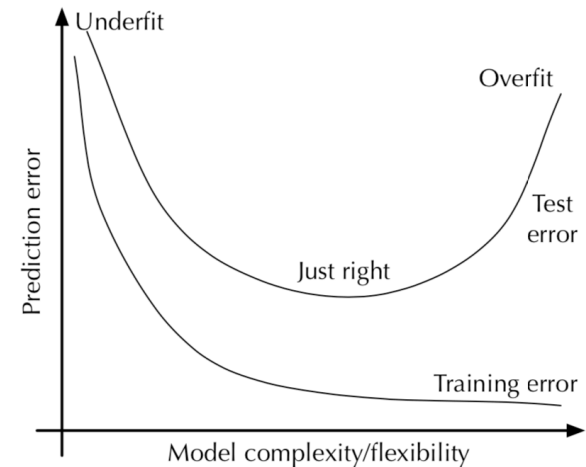
## Overfitting

- Learning a model that “knows” the training data very well but does not generalize



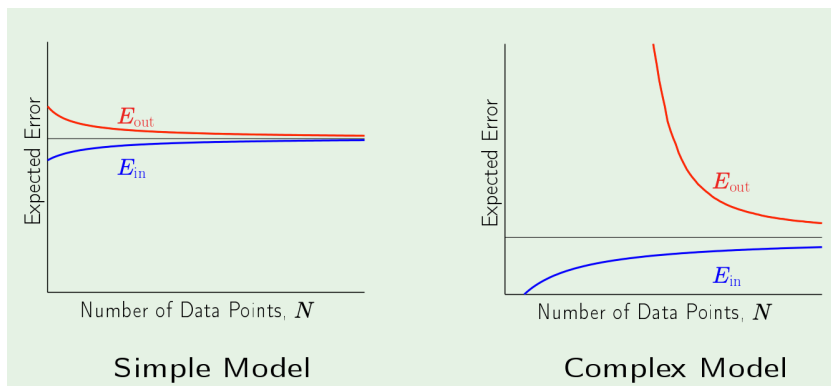
13

## Model complexity



14

## Number of data instances



15

## Overfitting

- Reasons
  - model is too complex
  - model is fitting noise present in the training data
  - training data is not a representative sample of the distribution
- How to prevent?
  - use more training data
  - use fewer features
  - regularize your model

16

## Generalization

- We can use a ML method to calculate:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} J(\mathbf{w})$$

- Problem: it may overfit the training data
  - ✓ we want better generalization
- Solution: split your data in train, validation, test
  - ✓ use train and validation to select the best hypothesis
  - ✓ use test for final evaluation and report

17

## Model selection

## Train and test

TRAIN SET

TRAIN SET

TEST SET

19

## Train, validation, and test

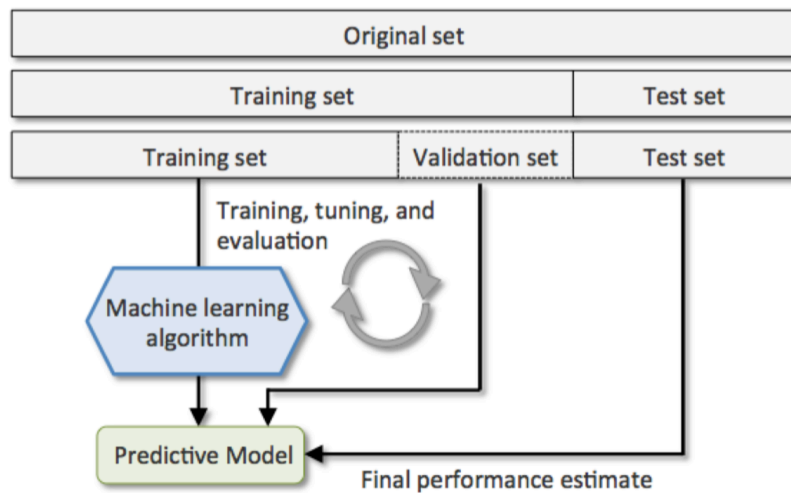
TRAIN SET

TRAIN SET

VALID SET

TEST SET

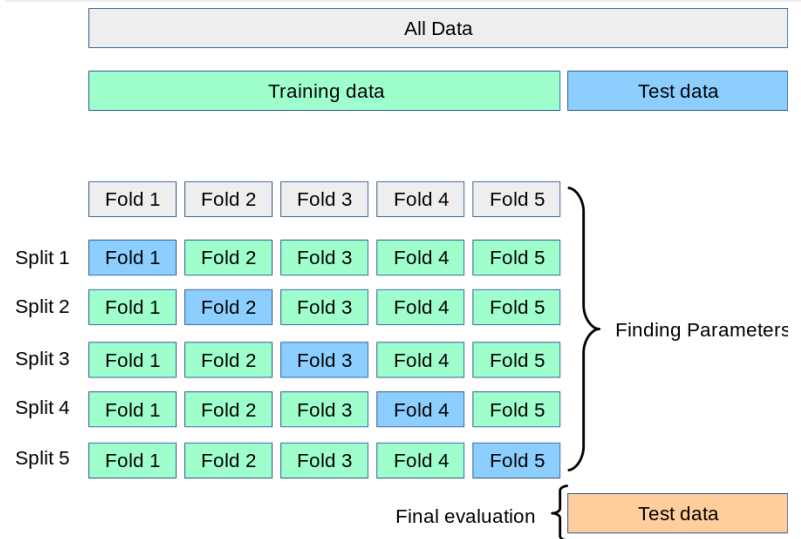
20



from INFO-4604: Applied Machine Learning, Fall 2017, Michael Paul, Univ. of Colorado

21

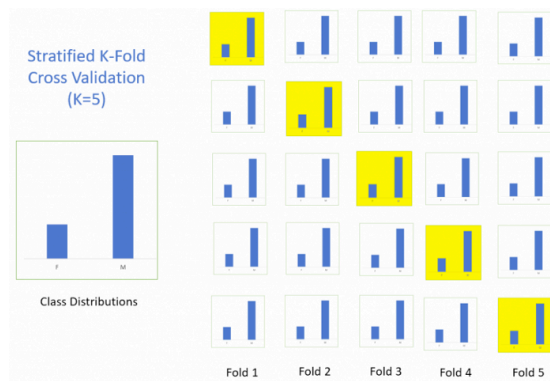
## k-fold cross validation



[https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)

22

## Stratified cross validation



**Stratified cross validation** aims at having the same class distribution within each fold

<https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85>

23

## Evaluation

## Confusion matrix (2 classes)

		Predicted condition	
		Positive (PP)	Negative (PN)
Actual condition	Total population = P + N		
	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

[https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)

25

## Confusion matrix (example)

		Predicted condition	
		Cancer	Non-cancer
Actual condition	Total 8 + 4 = 12	7	5
	Cancer 8	6	2
	Non-cancer 4	1	3

[https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)

26

## Evaluation metrics (2 classes)

### accuracy (ACC)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

### F1 score

is the harmonic mean of precision and sensitivity

$$F_1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

### Matthews correlation coefficient (MCC)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

[https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)

27

## Confusion matrix (example >2 classes)

**CIFAR-10 Confusion Matrix**

True Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck		
	923	4	21	8	4	1	5	5	23	6	92.3%	7.7%
	5	972	2					1	5	15	97.2%	2.8%
	26	2	892	30	13	8	17	5	4	3	89.2%	10.8%
	12	4	32	826	24	48	30	12	5	7	82.6%	17.4%
	5	1	28	24	898	13	14	14	2	1	89.8%	10.2%
	7	2	28	111	18	801	13	17		3	80.1%	19.9%
	5		16	27	3	4	943	1	1		94.3%	5.7%
	9	1	14	13	22	17	3	915	2	4	91.5%	8.5%
	37	10	4	4		1	2	1	931	10	93.1%	6.9%
	20	39	3	3			2	1	9	923	92.3%	7.7%
											88.0%	93.9%
											85.8%	79.0%
											91.4%	89.7%
											91.0%	94.1%
											94.1%	94.8%
											95.0%	
											12.0%	6.1%
											14.2%	21.0%
											8.6%	10.3%
											8.4%	5.9%
											5.2%	5.0%
											airplane	automobile
											bird	cat
											deer	dog
											frog	horse
											ship	truck
											Predicted Class	

<https://www.mathworks.com/help/deeplearning/ref/confusionchart.html>

28