Introduction to Learning

Learning as a meta-algorithm

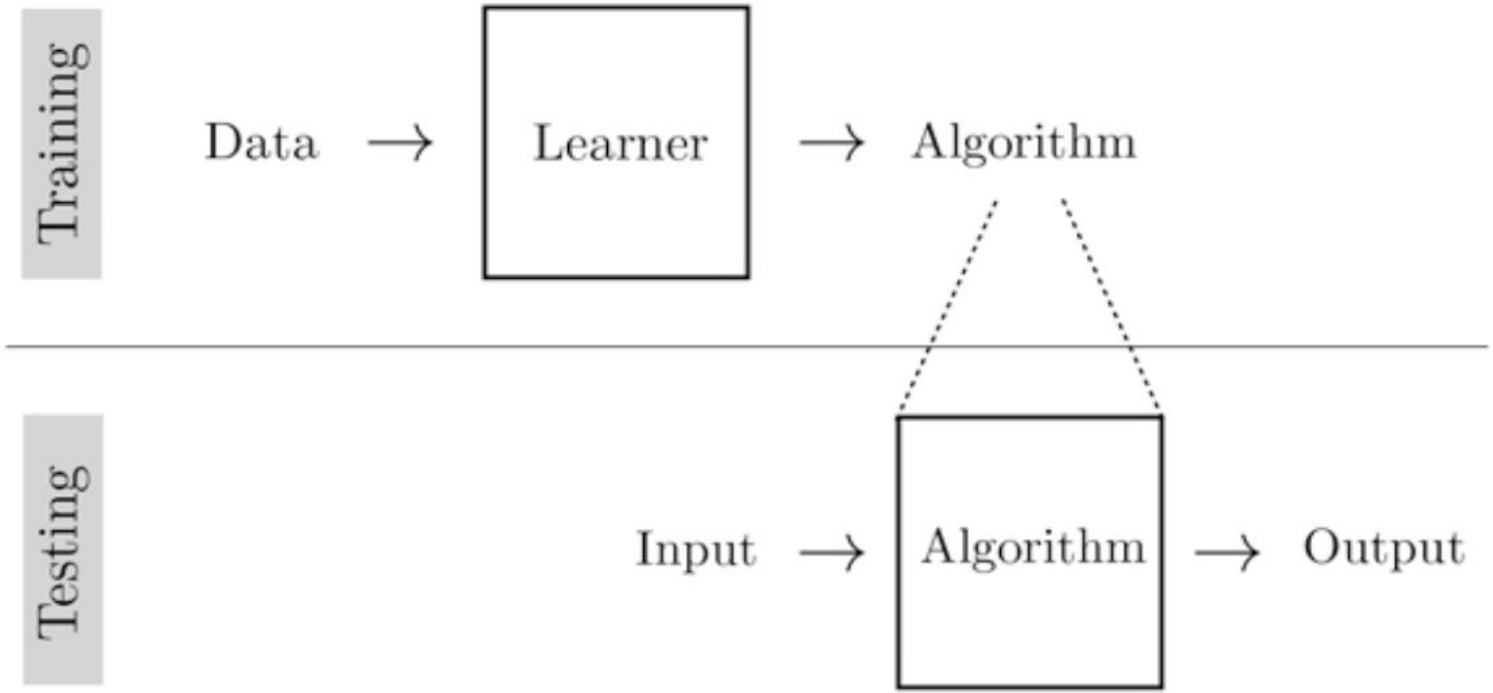


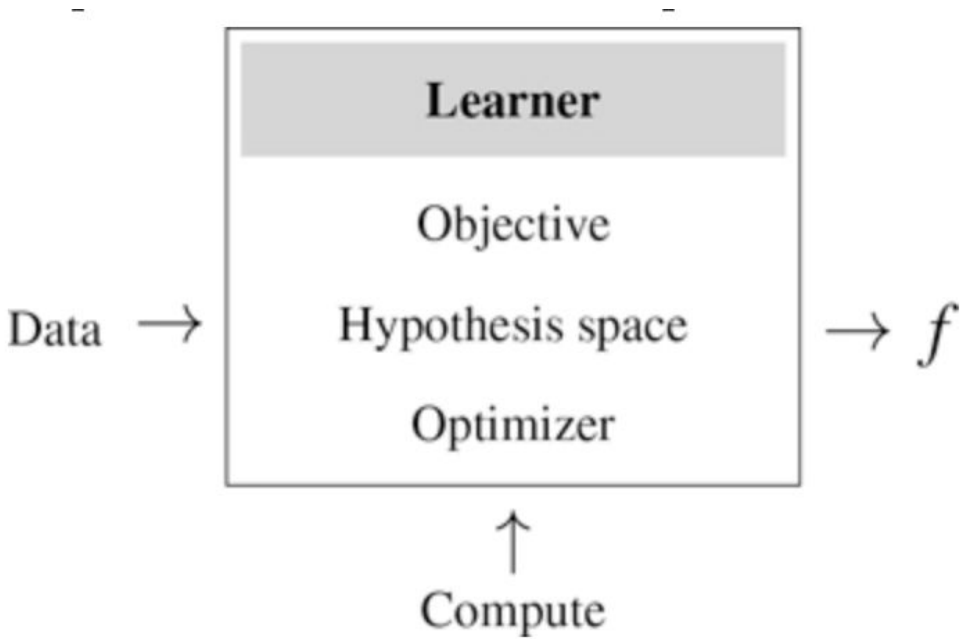
Figure 9.1: Learning is an algorithm that outputs algorithms.

Types of learning

1. Supervised - examples
2. Unsupervised - objective function
3. Reinforcement - reward (no training data)

Key ingredients

1. **Objective**: What does it mean for the learner to succeed, or, at least, to perform well?
2. **Hypothesis space**: What is the set of possible mappings from inputs to outputs that we will search over?
3. **Optimizer**: How, exactly, do we search the hypothesis space for a specific mapping that maximizes the objective?

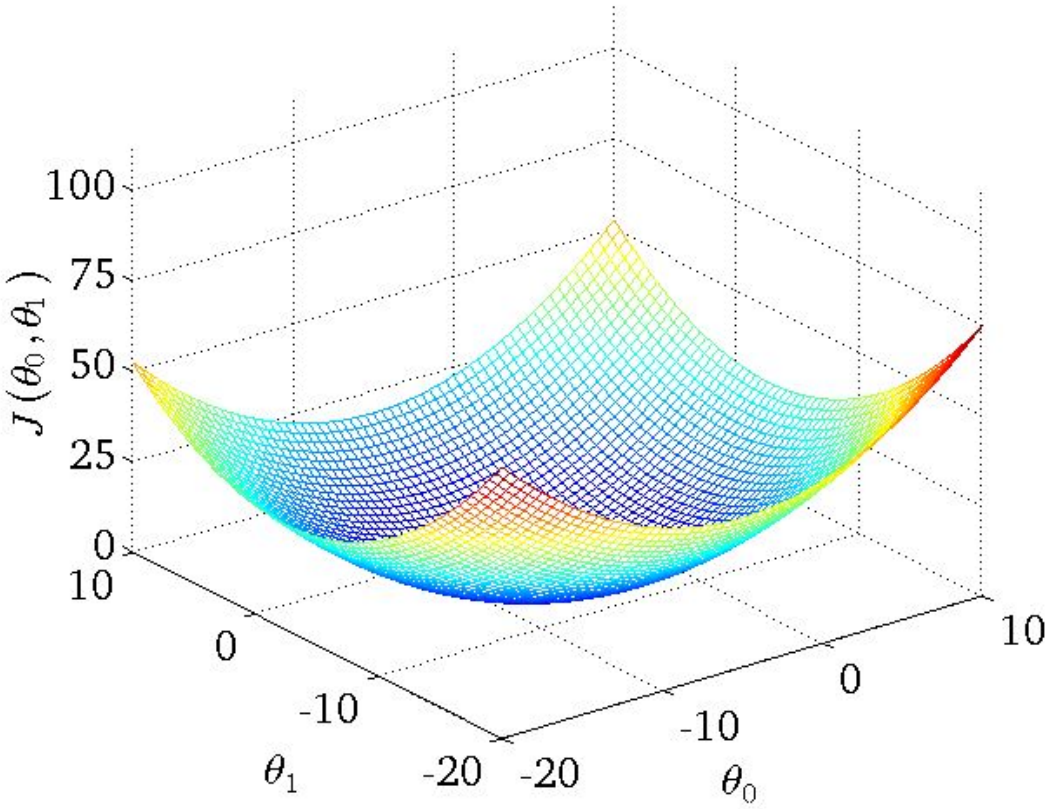

Learner
Objective
Data
Hypothesis space
f
Optimizer
Compute

Parameterization

1. Hypothesis space
    a. All mappings from $\mathbb{R}^2 \rightarrow \mathbb{R}$
    b. $\mathbb{R}x\mathbb{R} \rightarrow \mathbb{R}_{>0}$ (Distance Matrix)
2. Parameterized Hypothesis space
    a. $\mathbb{R} \rightarrow \mathbb{R}$
        i. $y = \theta_1 x + \theta_0$
        ii. $y = \theta_2 \theta_1 x + \theta_0$

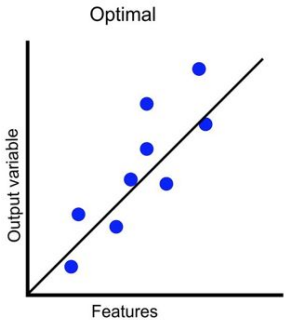
Objective Function

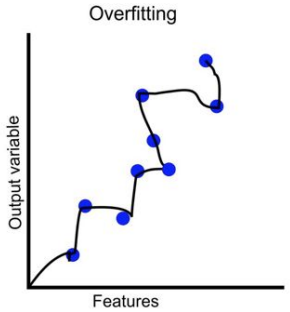
1.



Overparameterization

1. What if the number of parameters are equal to the number of training examples?
2. The model has the ability to memorise each example!
3. So we should have a lot more examples than parameters





Overparameterization

1. What about deep learning?
    a. DL Models have millions of parameters
2. Why do they work so well? Some explanations
    a. Architectural constraints (e.g. CNNs are designed to recognize spatial hierarchies of features)
    b. Overparametrization can make the optimization landscape smoother
    c. In high-dimensional spaces, sharp minima occupy a smaller volume than wide minima.
    d. Stochastic Gradient Descent (SGD) uses noisy gradient updates due to sampling mini-batches of data rather than the entire dataset
    e. SGD exhibits a bias toward minimizing the norm of the weights (e.g., $// W //$),

Overparameterization

1. Consider a deep neural network trained on image classification:
    a. Initially, the network fits simple, global features (e.g., edges, color blobs).
    b. As training progresses, it begins to capture finer details.
    c. If training is stopped early or if the dataset contains noise, SGD's dynamics guide the network to prioritize these simple global features, ignoring noise or overly complex patterns.

Empirical Risk Minimization

1. Empirical Risk Minimization

$$\arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) \quad \triangleleft \quad \text{ERM} \tag{9.1}$$

2. Maximum Likelihood

   a. “we are trying to infer the hypothesis f that assigns the highest probability to the data”

$$\arg\max_{f} p\left(\{\mathbf{y}^{(i)}\}_{i=1}^{N} | \{\mathbf{x}^{(i)}\}_{i=1}^{N}, f\right) \quad \triangleleft \quad \text{Max likelihood learning} \tag{9.2}$$

Empirical Risk Minimization

1. Maximum A Posteriori
    a. “infers the most probable hypothesis given the training data”

$$\arg\max_f p\left(f \mid \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N\right) \quad \triangleleft \quad \text{MAP learning} \tag{9.3}$$

$$= \arg\max_f p\left(\{\mathbf{y}^{(i)}\}_{i=1}^N \mid \{\mathbf{x}^{(i)}\}_{i=1}^N, f\right) p(f) \quad \triangleleft \quad \text{by Bayes' rule} \tag{9.4}$$

    b. Maximum Likelihood * Prior

Least Squares Regression

1. The learning problem

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2. \tag{9.7}$$

2. Objective function

$$J(\theta) = \sum_{i=1}^{N} (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2. \tag{9.8}$$

Classification and Softmax Regression

1. 0-1 loss

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \mathbb{1}(\hat{\mathbf{y}} \neq \mathbf{y}), \tag{9.15}$$

2. Cross-entropy loss

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k \quad \triangleleft \quad \text{cross-entropy loss} \tag{9.16}$$

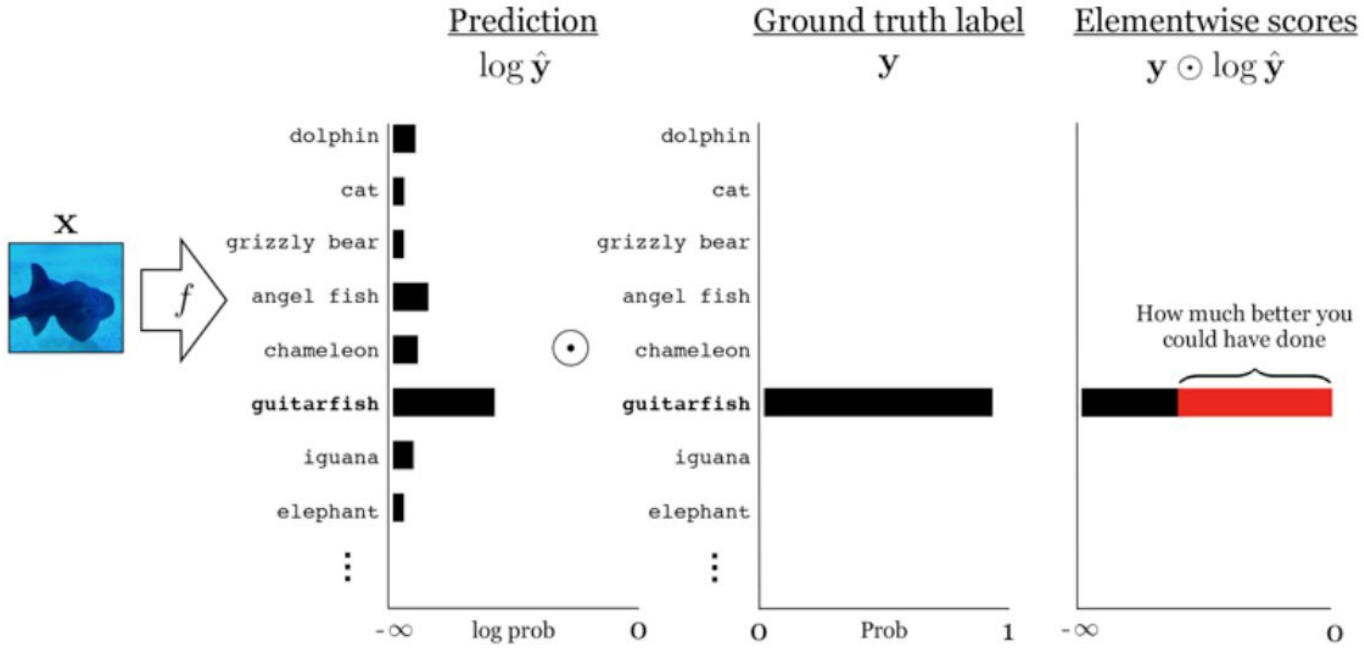
$$\mathbf{z} = z_\theta(\mathbf{x}) \tag{9.17}$$

$$\hat{\mathbf{y}} = \texttt{softmax}(\mathbf{z}) \tag{9.18}$$

$$\hat{y}_j = \frac{e^{-z_j}}{\sum_{i=1}^{K} e^{-z_k}}. \tag{9.19}$$

Classification and Softmax Regression

1.



References

1. Foundations of Computer Vision - Chapter 9