



# Chapter 2: Empirical Risk Minimization

Advanced Topics in Statistical Machine Learning

---

Tom Rainforth

Hilary 2024

[rainforth@stats.ox.ac.uk](mailto:rainforth@stats.ox.ac.uk)

# Problem Setting

- For the first part of the course we are going to focus on **discriminative** approaches to **supervised learning**
- We will assume that there is true underlying joint distribution  $P(X, Y)$  over inputs  $X \in \mathcal{X}$  and outputs/labels  $Y \in \mathcal{Y}$
- We do not know  $P(X, Y)$  itself, but we have access to samples from it in the form of our dataset  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ 
  - We will implicitly assume that each datapoint  $(x_i, y_i)$  is **independent and identically distributed** (i.i.d.)
- Our goal is to accurately predict  $Y|X = x$  (or some characterization of  $Y$ , e.g. class probabilities) for new inputs  $x$
- As we are taking a discriminative approach, this is done by directly **learning a function**  $f : \mathcal{X} \rightarrow \mathcal{Y}$ 
  - We will reason about  $P(X, Y)$ , but not direct model it

# Loss Functions

- To learn  $f$ , we need an objective function that quantifies the relative performance of different possible functions
- This requires a **loss function**  $L(y, f(x), x)$  that attributes a cost to making a prediction  $f(x)$  for input  $X = x$  when the true output is  $Y = y$
- A loss function is thus any function that takes the form<sup>1</sup>

$$L : \mathcal{Y} \times \mathcal{Y} \times \mathcal{X} \rightarrow \mathbb{R}^+$$

such that its outputs are strictly non-negative (they can be 0)

- $L(y, f(x), x)$  is measure of the discrepancy between the predicted output  $f(x)$  and the true output  $y$  at the input  $x$
- Most loss function functions do not depend directly on  $x$ , so we will simplify this notation to  $L(y, f(x))$  from now on

---

<sup>1</sup>This definition is not always exactly true as sometimes our predictions are not truly in  $\mathcal{Y}$ , e.g. predicting class probabilities rather than classes directly.

- A loss function is defined for particular instances of  $X$  and  $Y$
- To convert it into an objective we need to consider its value over different possible instances
- By far the most common approach to this is to consider the expectation of the loss over possible input–output pairs

## Risk

For a given loss function  $L$ , the **risk**  $R$  of a predictive function  $f$  is given by the expected loss

$$R(f) := \mathbb{E}_{X,Y} [L(Y, f(X))] .$$

# Hypothesis Space

- The need to choose a machine learning algorithm means that there are generally restrictions on what  $f$  we can learn
- The set of all possible functions is known as the **hypothesis space**  $\mathcal{H}$
- We can now formally define the optimal  $f$  as

$$f^* = \arg \min_{f \in \mathcal{H}} R(f) = \arg \min_{f \in \mathcal{H}} \mathbb{E}_{X,Y} [L(Y, f(X))]$$

- In many cases our function is explicitly parameterized by some parameters  $\theta \in \Theta$  such that our hypothesis space is represented through the form of  $f_\theta$  and the set of allowable parameters  $\Theta$ . We then have

$$f^* = f_{\theta^*} \quad \text{where} \quad \theta^* = \arg \min_{\theta \in \Theta} \mathbb{E}_{X,Y} [L(Y, f_\theta(X))]$$

# Empirical Risk

- The risk  $R(f)$  (sometimes known as the **true risk**) is an expectation with respect to the true (unknown) joint distribution of  $(X, Y)$
- In practice it is thus unknown as we do not have infinite data or computation, but we can compute the **empirical risk**:

$$\hat{R}(f) := \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$$

- For an arbitrary  $f$  that is chosen independently of the data, the empirical risk is an unbiased estimate of the true risk
- ... but if  $f$  is trained using this data it becomes a negatively biased because we have actively chosen  $f$  for which the empirical risk is smaller

# Empirical Risk Minimization

- Ideally we would like to learn  $f^\star$ , but the intractability of  $R(f)$  means that this is impossible
- **Empirical Risk Minimization (ERM)**: minimize the empirical risk instead

$$\hat{f} = \arg \min_{f \in \mathcal{H}} \hat{R}(f) = \arg \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$$

- Training a discriminative machine learning algorithm now corresponds to (approximately) performing this ERM
- Most of supervised machine learning now boils down to the three choices of a) our hypothesis class, b) our loss function, and c) our mechanism for performing the subsequent ERM

## Example Hypotheses Classes

- Linear functions:  $\mathcal{H} := \{f : f(x) = w^\top x + b\}$ , parametrized by  $w \in \mathbb{R}^p$  and  $b \in \mathbb{R}$
- Linear functions with nonlinear mappings:  
 $\mathcal{H} := \{f : f(x) = w^\top \varphi(x) + b\}$  for some predefined **nonlinear feature expansion**  $\varphi : \mathcal{X} \rightarrow \mathbb{R}^D$ , typically with  $D > p$ 
  - If we allow  $D$  to be potentially infinitely dimensional, this results in an important type of hypothesis space: **Reproducing Kernel Hilbert Space (RKHS)**.
- Deep learning: computational graph of parameterized, differential, mappings that starts with  $x$  and ends with  $y$

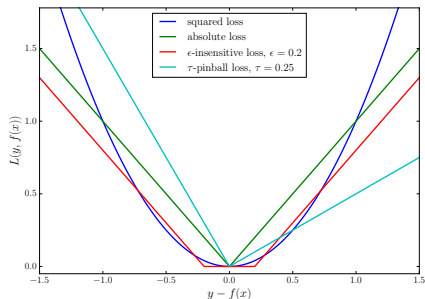


# Example Loss Functions for Regression

**Squared loss:**  $(y - f(x))^2$   
optimal  $f$  is the conditional mean  $\mathbb{E}[Y|X = x]$ ,

**Absolute loss:**  $|y - f(x)|$   
optimal  $f$  is the conditional median  $\text{med}[Y|X = x]$ ,

**$\tau$ -pinball loss** ( $\tau \in (0, 1)$ ):  
 $2 \max\{\tau(y - f(x)), (\tau - 1)(y - f(x))\}$ ,  
optimal  $f$  is the  $\tau$ -quantile of  $P(Y = y|X = x)$ .



**Figure 1:** Loss functions for regression

# Loss Functions for Classification

- Rather than have  $f(x)$  directly predict classes, we normally have  $f(x)$  map to some preference of the different classes, from which we have a fixed mapping to the classes themselves
- For binary classification, we typically denote our classes as  $-1$  and  $+1$  and then predict classes as  $\text{sign}(f(x))$ , such that the magnitude of  $f(x)$  represents the “confidence”
  - We can also derive the class probabilities via

$$P_y(x) = \frac{1}{1 + \exp(-yf(x))}$$

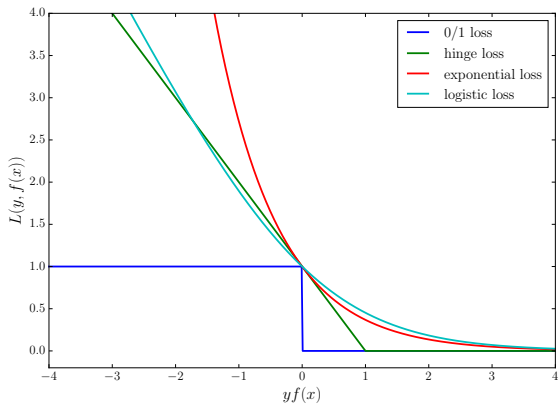
- For multi-class classification, we have  $\mathcal{Y} = \{1, \dots, K\}$  and typically learn a mapping  $f_k(x)$  of each class  $k$  and then predict classes as  $\arg \max_k f_k(x)$ 
  - We can derive the class probabilities via the **softmax function**

$$P_y(x) = \frac{\exp(f_y(x))}{\sum_{k=1}^K \exp(f_k(x))}$$

## Example Loss Functions for Binary Classification

- 0/1 loss (misclassification loss)  $L(y, f(x)) = \mathbb{I}\{yf(x) \leq 0\}$   
Optimal solution is called the **Bayes classifier** and is given by  
$$f(x) = \arg \max_{y \in \{-1, 1\}} P(Y = y | X = x)$$
- Hinge loss  $L(y, f(x)) = \max(0, (1 - yf(x)))$   
Used in **support vector machines**, leads to sparse solutions
- Exponential loss  $L(y, f(x)) = \exp(-yf(x))$   
Used in **boosting** algorithms like Adaboost
- Logistic loss  $L(y, f(x)) = \log(1 + \exp(-yf(x)))$   
Used in **logistic regression**, has generative interpretations

Good losses should generally not just penalize misclassification, they should also depend on the confidence of our predictions



**Figure 2:** Loss functions for binary classification

## Example Loss Functions for Multi-Class Classification

Most common loss for multi-class classification loss is the **cross-entropy** loss which generalizes the logistic loss:

$$\begin{aligned} L(y, f(x)) &= -\log P_y(x) \\ &= -\log \frac{\exp(f_y(x))}{\sum_{k=1}^K \exp(f_k(x))} \end{aligned}$$

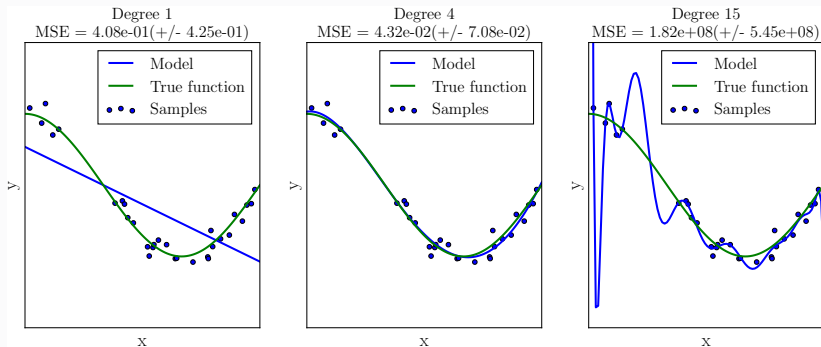
- How complex should we allow functions  $f$  to be? If hypothesis space  $\mathcal{H}$  is “too large”, ERM can lead to **overfitting**:

$$\hat{f}(x) = \begin{cases} y_i & \text{if } x = x_i, \\ 0 & \text{otherwise} \end{cases}$$

will have zero empirical risk, but is useless for generalization.

- For a completely unconstrained  $f$ , it is possible to simultaneously achieve  $\hat{R}(\hat{f}) = 0$  and  $R(\hat{f}) = \infty$  for any dataset!
- Remember that  $\hat{R}(\hat{f})$  is negatively biased estimate for  $R(\hat{f})$
- To avoid overfitting we either need to either restrict our hypothesis class, or apply **regularization**

# Overfitting and Underfitting



**Figure 3:** Underfitting and Overfitting

- Powerful models typically must be flexible and are therefore usually prone to overfitting
- **Regularization**: add a term  $r(f)$ , known as a **regularizer**, to the empirical risk that penalizes complex functions:

$$\hat{f} = \arg \min_{f \in \mathcal{H}} \hat{R}(f) + r(f) = \arg \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) + r(f)$$

- Note that we would not need to regularize the true risk if we could calculate it: the job of the regularizer is to account for the overfitting bias induced by optimizing the empirical risk



# Regularizers

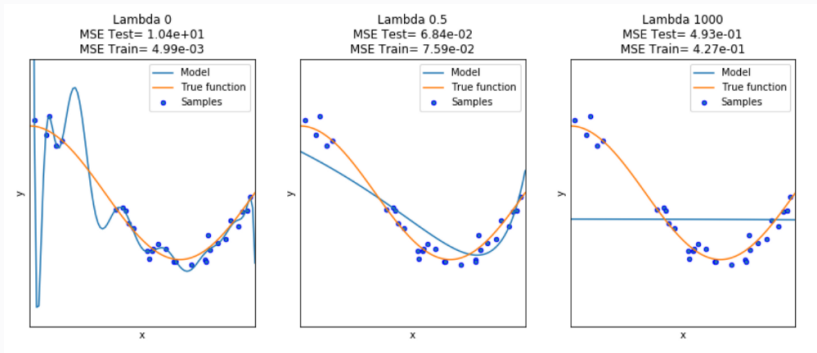
- For parameterized functions, it is common for larger values of the parameters  $\theta$  to correspond to more complex functions
  - Example: larger coefficients in polynomial regression produce steeper gradients and more “wiggly” functions
- This means we can introduce regularization by **penalizing large values of  $\theta$**

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \hat{R}(f_{\theta}) + \lambda \|\theta\|_{\rho}^{\rho}$$

where  $\rho \geq 1$ , and  $\|\theta\|_{\rho} = (\sum_{j=1}^p |\theta_j|^{\rho})^{1/\rho}$  is the  $L_{\rho}$  norm of  $\theta$  (also of interest when  $\rho \in [0, 1)$ , but is no longer a norm)

- Also known as **shrinkage** methods—parameters are shrunk towards 0
- $\lambda$  is a **hyperparameter** that controls the amount of regularisation, and resulting complexity of the model

# Regularization Example



**Figure 4:** Effect of  $L_2$  regularization. Source:

<https://towardsdatascience.com/understanding-regularization-in-machine-learning-d7dd0729dde5>

# Types of Regularization

- **Ridge regression / Tikhonov regularisation:**  $\rho = 2$  (Euclidean norm)
- **LASSO:**  $\rho = 1$  (Manhattan norm)
- **Sparsity-inducing** regularisation:  $\rho \leq 1$  (nonconvex if  $\rho < 1$ )
- **Elastic net** regularisation: mixed  $L_1/L_2$  penalty:

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \hat{R}(f_{\theta}) + \lambda [(1 - \alpha)\|\theta\|_2^2 + \alpha\|\theta\|_1]$$

- Sometimes we can even regularize directly in function space, e.g. using the **Sobolev norm**

$$\|f\|_{W^1}^2 = \int_{-\infty}^{+\infty} f(x)^2 dx + \int_{-\infty}^{+\infty} f'(x)^2 dx, \quad (1)$$

which penalises functions for being “wiggly:” it prefers functions that have small magnitude and small derivatives

- In discriminative supervised learning we look to find the function that minimizes the **risk**  $R(f) := \mathbb{E}_{X,Y}[L(Y, f(X))]$
- As this is intractable we instead use the **empirical risk**  
 $\hat{R}(f) := \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$
- To prevent overfitting and allow generalization, we need to either constrain our **hypothesis class** of functions or apply some form of regularization
- A large portion learning now boils down to how we choose our hypothesis class, loss function, regularizer, and optimizer

## Further Reading

- The “Statistical Machine Learning” course will go into more depth on ERM (particularly in terms of generalization).
- Examples in the course notes
- Chapter 2 of Trevor Hastie, Robert Tibshirani, and Jerome Friedman. **The elements of statistical learning: data mining, inference, and prediction.** Springer Science & Business Media, 2009 (<https://web.stanford.edu/~hastie/ElemStatLearn/>)