

Lecture 11: Learning Parametric Models

- We have until now looked at two simple parametric models
 - linear regression
 - logistic regression
 - generalized linear models
- Parametric models assume a functional form described a fixed number of parameters
- **Learning** a parametric model implies tuning the parameters to fit the training dataset
- In the next 2 lectures, we will discuss basic principles for learning these models

— Key components of any parametric ML algorithm:

- The data $\{ \underline{x}^{(i)}, y^{(i)} \}_{i=1}^N$
- A model with parameters $\underline{\Theta}$ [e.g. $\hat{y} = \underline{x}^T \underline{\Theta}$]
- A loss function $L(y, \hat{y})$ to quantify goodness of model predictions during training
- An optimization algorithm to tune the model's parameters so as to minimize the loss

— Key components of any parametric ML algorithm:

- The data $\{\underline{x}^{(i)}, y^{(i)}\}_{i=1}^N$

→ • A model with parameters $\underline{\theta}$ [e.g. $\hat{y} = \underline{x}^T \underline{\theta}$]

- A loss function $L(y, \hat{y})$ to quantify goodness of model predictions during training
- An optimization algorithm to tune the model's parameters so as to minimize the loss

Recall linear regression model

$$y = \underline{x}^T \underline{\theta} + \epsilon$$

$$\begin{cases} \bullet \text{ linear in model parameters } \underline{\theta} \\ \bullet \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2) \end{cases}$$

Assumptions:

(a) Linear model

$$f(\underline{x}; \underline{\theta}) = \underline{x}^T \underline{\theta} \quad (\text{linear in parameters})$$

(b)

Noise is additive \rightarrow

$$\underline{x}^T \underline{\theta} + \epsilon \quad \checkmark$$

$$\text{not multiplicative} \rightarrow \underline{x}^T \underline{\theta} \cdot \epsilon \quad \times$$

(c)

Noise distribution is assumed $\rightarrow \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$

to be Gaussian

Nonlinear Parametric Functions (Regression)

- A nonlinear regression model

$$y = \underline{f_{\underline{\theta}}}(\underline{x}) + \epsilon$$

sometimes also
written as $f(\underline{x}; \underline{\theta})$

- $\underline{f_{\underline{\theta}}}(\underline{x})$ could be any nonlinear function, which depend on some model parameter $\underline{\theta}$ that control the shape of the function
- Different values of $\underline{\theta}$ result in different functions $\underline{f_{\underline{\theta}}}(\cdot)$
- In mathematical terms, we say we have a **parametric family** of functions

$$\{ \underline{f_{\underline{\theta}}}(\cdot) \text{ s.t. } \underline{\theta} \in \underline{\Theta} \}$$

space of all possible
parameter vectors

Nonlinear Parametric Functions (Regression)

A nonlinear regression model

$$y = f_{\underline{\theta}}(\underline{x}) + \epsilon$$

- If noise ϵ is taken to be Gaussian with zero mean and variance σ_{ϵ}^2 we obtain a Gaussian likelihood

$$p(y | \underline{x}; \underline{\theta}) = \mathcal{N}(f_{\underline{\theta}}(\underline{x}), \sigma_{\epsilon}^2)$$

- Note that in the linear case, $f_{\underline{\theta}}(\underline{x}) = \underline{x}^T \underline{\theta}$ and

$$p(y | \underline{x}; \underline{\theta}) = \mathcal{N}(\underline{x}^T \underline{\theta}, \sigma_{\epsilon}^2)$$

Nonlinear Parametric Functions (Classification)

- Recall linear classification i.e. logistic regression

- Logit : $z = \underline{x}^T \underline{\theta}$

- $p(y=1|\underline{x}) : h(z) = \frac{e^z}{1+e^z} = \frac{e^{\underline{x}^T \underline{\theta}}}{1+e^{\underline{x}^T \underline{\theta}}}$

- Nonlinear classification can be constructed by considering $z = f_{\theta}(\underline{x})$ as a generalization of the logistic regression

$$p(y=1|\underline{x}) = \frac{e^{f_{\theta}(\underline{x})}}{1 + e^{f_{\theta}(\underline{x})}}$$

- Similarly, we could also have a multi-class nonlinear classifier by generalizing multi-class logistic regression model (recall softmax)

— Key components of any parametric ML algorithm:

- The data $\{\underline{x}^{(i)}, y^{(i)}\}_{i=1}^N$

- A model with parameters $\underline{\theta}$ [e.g. $\hat{y} = \underline{x}^T \underline{\theta}$]

→ • A loss function $L(y, \hat{y})$ to quantify goodness of model predictions during training

- An optimization algorithm to tune the model's parameters so as to minimize the loss

Loss functions

- After choosing a certain parametric model class, the next step is to "learn" the model — find suitable values of parameters so that the model describes the true (but often unknown) input-output relationship as accurately as possible
- For parametric models, "learning" is typically framed as an optimization

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmin}} \underbrace{\frac{1}{N} \sum_{i=1}^N \overbrace{L(y^{(i)}, f_{\underline{\theta}}(x^{(i)}))}^{\text{loss function}}}_{\text{Cost function } J(\underline{\theta})}$$

Usually solved using some numerical optimization method

- We seek to minimize a cost function which is the average of some user-defined loss function L evaluated on the training data

Loss function is a proxy of Generalization

- Natural Idea: Find the value of $\underline{\theta}$ that fits the training data well
- However, our ultimate goal is **not** to fit the training data very well but rather to find a model that can **generalize to new data**
 - In other words, we are actually interested in solving:

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmin}} \underbrace{E_{\text{new}}(\underline{\theta})}_{\rightarrow E_*[E(y, \hat{y}(x^*; \underline{\theta}))]}$$

- Issue is $E_{\text{new}}(\underline{\theta})$ is unknown since $p(x, y)$ is unknown
However, it is still important to keep in mind that

$$\text{Training objective } \hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N L(y^{(i)}, f_{\underline{\theta}}(x^{(i)}))$$

is only a proxy for the actual objective

→ Viewing the training objective as a proxy will help us in choosing how we approach the optimization problem

- Optimization accuracy vs Statistical accuracy

It is not meaningful to optimize $J(\underline{\theta})$ with greater accuracy than the statistical error in the estimate.

↪ difficult to determine though

- Choice of loss function

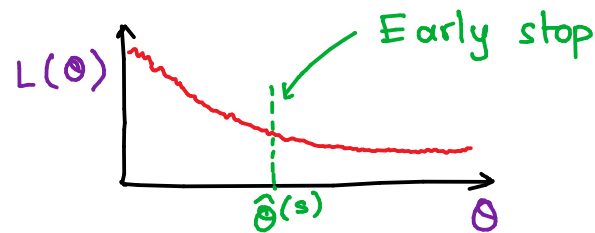
One should choose a loss function with the aim of making the optimization problem

easier to solve

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmin}} \quad \frac{1}{N} \sum_{i=1}^N L(y^{(i)}, f_{\underline{\theta}}(x^{(i)}))$$

(e.g. use convex loss functions)

- Early stopping



$$\hat{\theta}^{(0)} \rightarrow \hat{\theta}^{(1)} \rightarrow \hat{\theta}^{(2)} \dots \hat{\theta}^{(s)} \dots \rightarrow$$

- Explicit regularization by adding a parameter-penalty term in the cost function (e.g. L_2 -regularization)

Different loss functions (for regression)

- Squared error loss: $L(y, \hat{y}) = (y - \hat{y})^2$ ← default choice for linear regression
 - Maximum likelihood perspective: Noise ϵ is Gaussian, $\epsilon \sim N(0, \sigma_\epsilon^2)$
- Absolute error loss: $L(y, \hat{y}) = |y - \hat{y}|$ ← robust to outliers since it grows more slowly for large errors
 - Maximum likelihood perspective: Noise ϵ is Laplace, $\epsilon \sim \mathcal{L}(0, b_\epsilon)$
 $\hookrightarrow \mathcal{L}(x | \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right)$
- ϵ -insensitive loss
(extension of absolute error loss)
$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| < \epsilon \\ |y - \hat{y}| - \epsilon & \text{otherwise} \end{cases}$$
 ← ϵ chosen by user
- Huber loss (hybrid between squared error loss and absolute error loss)
$$L(y, \hat{y}) = \begin{cases} \frac{1}{2} (y - \hat{y})^2 & \text{if } |y - \hat{y}| < 1 \\ |y - \hat{y}| - \frac{1}{2} & \text{otherwise} \end{cases}$$

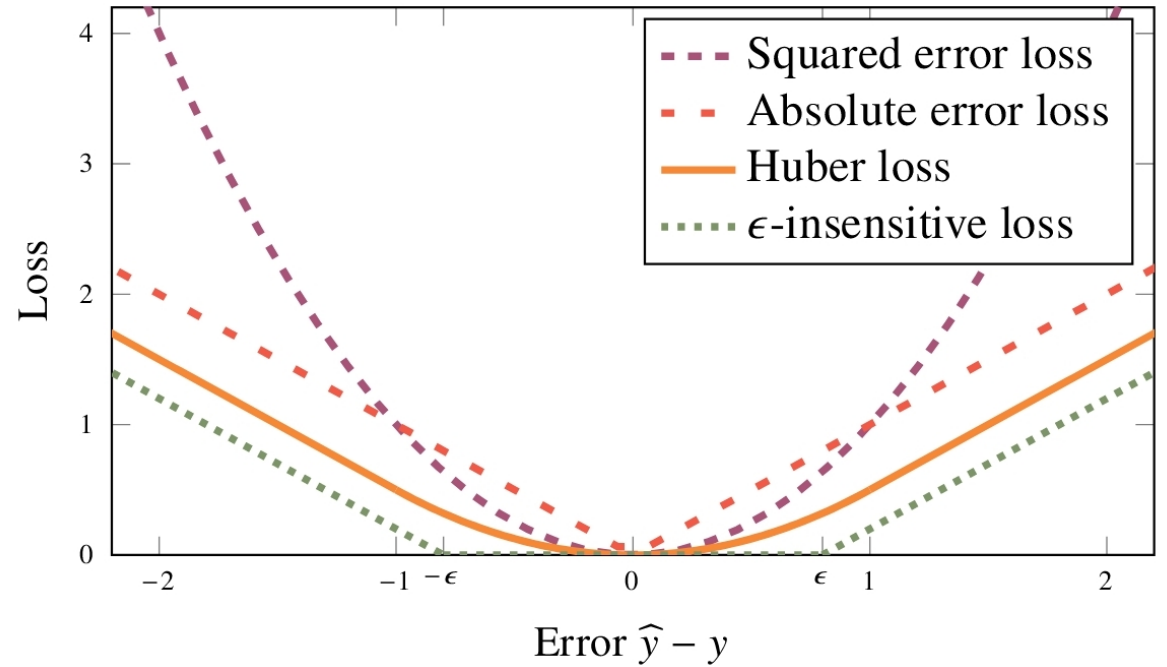
Different loss functions (for regression)

Squared error loss

$$L(y, \hat{y}) = (y - \hat{y})^2$$

Absolute error loss

$$L(y, \hat{y}) = |y - \hat{y}|$$



ϵ -insensitive loss

$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| < \epsilon \\ |y - \hat{y}| - \epsilon & \text{otherwise} \end{cases}$$

ϵ -insensitive loss will turn out to be useful for support vector regression (SVR)

Huber loss

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2} (y - \hat{y})^2 & \text{if } |y - \hat{y}| < 1 \\ |y - \hat{y}| - \frac{1}{2} & \text{otherwise} \end{cases}$$

Different loss functions (for classification)

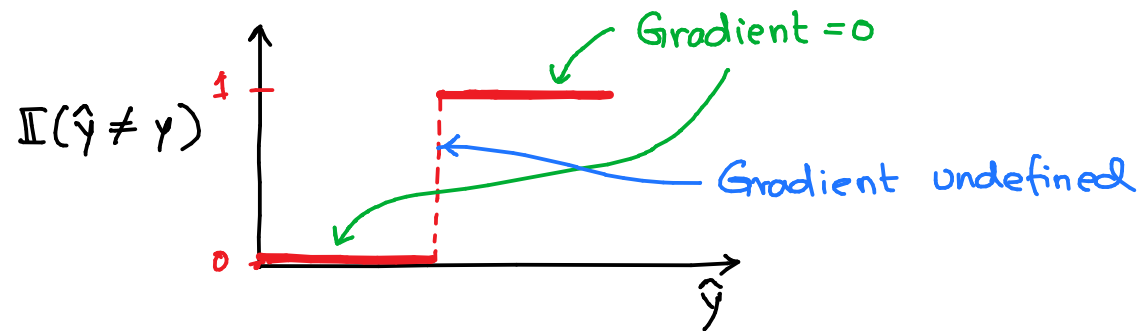
let's look at loss functions for binary classification first

- An intuitive loss function for is the **misclassification loss**

$$L(y, \hat{y}) = \mathbb{I}(\hat{y} \neq y) = \begin{cases} 0 & y = \hat{y} \\ 1 & y \neq \hat{y} \end{cases}$$

↑
indicator function

Although intuitive, this loss is rarely used in practice, because it is hard to optimize \rightarrow has zero gradients



Different loss functions (for classification)

- Cross-entropy loss forms a natural choice for a binary classifier that predicts class probabilities $p(y=1|\underline{x})$ in terms of $g(\underline{x})$

$$L(y, \hat{y}) = \begin{cases} \ln g(\underline{x}) & \text{if } y = 1 \\ 1 - \ln g(\underline{x}) & \text{if } y = -1 \end{cases}$$

\uparrow
 $g(\underline{x})$

- Another useful class of loss functions can be defined using the concept of margins
- Many classifiers can be constructed by thresholding some real-valued function $f(\underline{x}; \underline{\theta})$ at 0. We can write the class prediction as

$$\hat{y}(\underline{x}) = \text{sign} \{ f(\underline{x}; \underline{\theta}) \} \quad \begin{matrix} \text{(for binary classes)} \\ \{-1, 1\} \end{matrix}$$

E.g. Logistic regression can be brought into this form by $f(\underline{x}) = \underline{x}^T \underline{\theta}$

Concept of margin for (binary) classifiers

- The decision boundary of any classifier of the form

$$\hat{y}(\underline{x}) = \text{sign} \{ f(\underline{x}; \underline{\theta}) \}$$

$$\hat{y}(\underline{x}) \begin{cases} \rightarrow +1 \\ \rightarrow -1 \end{cases}$$

is given by the values of \underline{x} for which $f(\underline{x}) = 0$

- The **margin of a classifier** for a data point (\underline{x}, y) is $y \cdot f(\underline{x})$

$$\left. \begin{array}{l} f(\underline{x}) \rightarrow + \\ y \rightarrow + \end{array} \right\} \rightarrow y \cdot f(\underline{x}) \rightarrow +ve \text{ margin}$$

$$\left. \begin{array}{l} f(\underline{x}) \rightarrow - \\ y \rightarrow - \end{array} \right\} \rightarrow y \cdot f(\underline{x}) \rightarrow +ve \text{ margin}$$

- If classification is correct, margin is positive

- If y and $f(\underline{x})$ have different signs, margin is negative

(meaning incorrect classification)

- Data points with small margins are closer to decision boundary

Margin-based perspective of logistic loss

- In the lecture on logistic regression, we started out with a **class probability perspective**, modelling using $p(y=1 | \underline{x}) = g(\underline{x})$, then arrived at cross-entropy loss, and later for $g(\underline{x})$ modelled using the logistic function, we obtained the logistic loss

$$L(y, \hat{y}) = \ln \left(1 + e^{-y \cdot (\underline{x}^T \underline{\theta})} \right)$$

- Without linking the probabilistic perspective, we could consider the logistic loss as a generic margin-based loss

$$L(y, f(\underline{x})) = \ln \left(1 + e^{-\underbrace{y \cdot f(\underline{x})}_{\text{margin of the classifier}}} \right)$$

Hence,

- we postulate a classifier according to $\hat{y}(\underline{x}) = \text{sign} \{ f(\underline{x}; \underline{\theta}) \}$, and
- then learn the parameters of $f(\underline{x}; \underline{\theta})$ by minimizing $L(y, f(\underline{x}))$

Other margin-based loss functions for classification

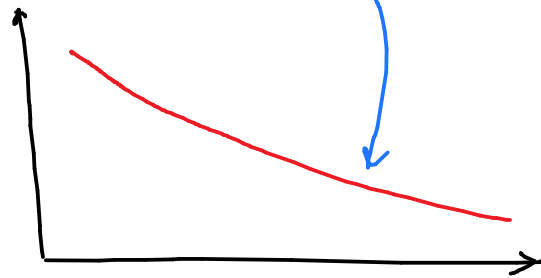
- Misclassification loss as margin-based loss

Misclassification loss

$$L(y, \hat{y}) = \mathbb{I}(\hat{y} \neq y) = \begin{cases} 0 & y = \hat{y} \\ 1 & y \neq \hat{y} \end{cases}$$

$$L(y, f(x)) = \begin{cases} 1 & \text{if } y \cdot f(x) < 0 \\ 0 & \text{otherwise} \end{cases}$$

- In principle, any DECREASING function is a candidate loss function



- e.g. Exponential loss

$$L(y, f(x)) = \exp(-y \cdot f(x))$$

- Not very robust to outliers, due to the exponential growth for negative margins

- Hinge loss (will be used in support vector machine)

$$L(y, f(x)) = \begin{cases} 1 - y \cdot f(x) & \text{for } y \cdot f(x) \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

[no probabilistic interpretation possible]

- Squared hinge loss (has probabilistic interpretation)

$$L(y, f(x)) = \begin{cases} (1 - y \cdot f(x))^2 & \text{for } y \cdot f(x) \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- less robust to outliers

- Huberized squared hinge loss

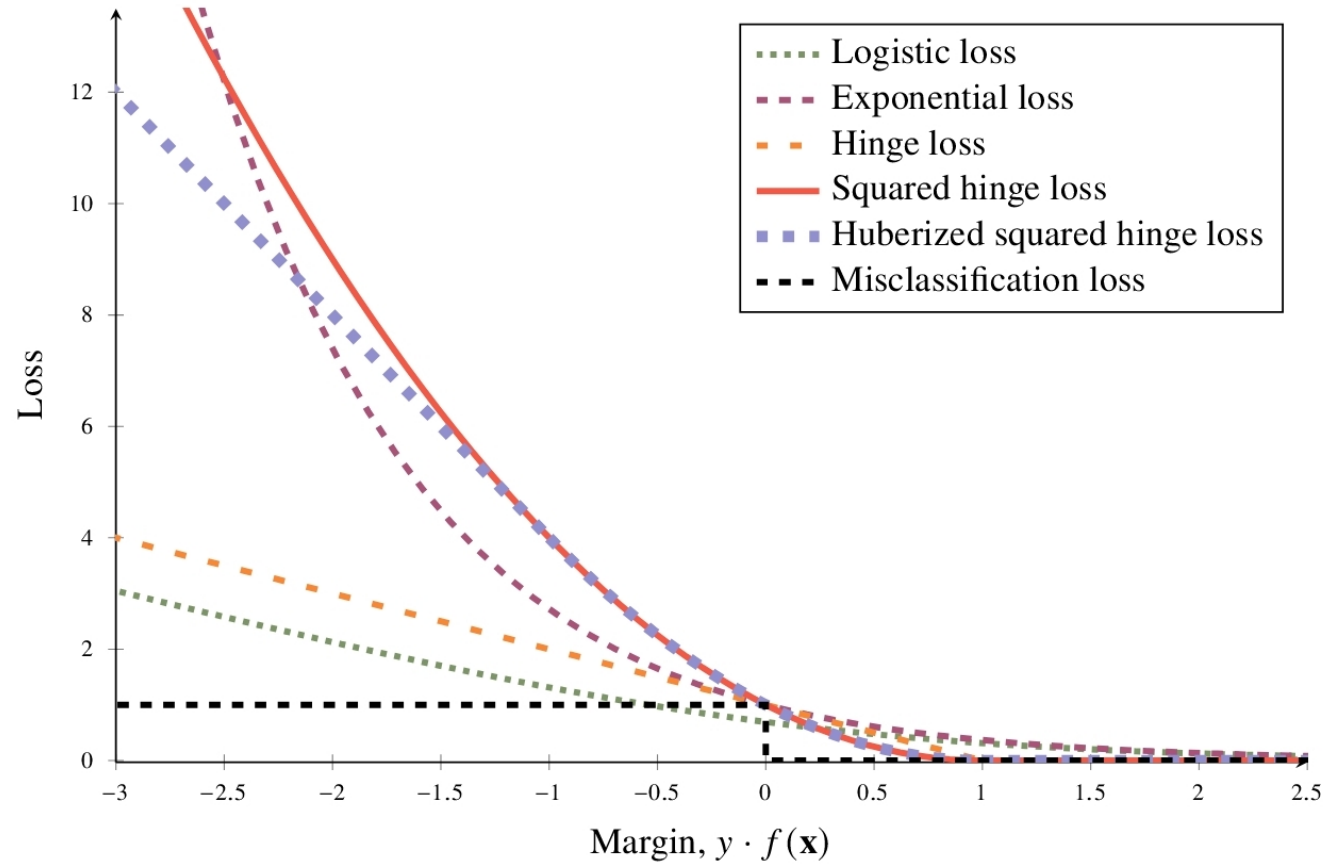
$$L(y, f(x)) = \begin{cases} -4 y \cdot f(x) & \text{for } y \cdot f(x) \leq -1 \quad (\text{linear margin}) \\ (1 - y \cdot f(x))^2 & \text{for } -1 \leq y \cdot f(x) \leq 1 \quad (\text{squared hinge loss}) \\ 0 & \text{otherwise} \end{cases}$$

Misclassification loss

$$L(y, f(x)) = \begin{cases} 1 & \text{if } y \cdot f(x) < 0 \\ 0 & \text{otherwise} \end{cases}$$

Exponential loss

$$L(y, f(x)) = \exp(-y \cdot f(x))$$



Hinge loss

$$L(y, f(x)) = \begin{cases} 1 - y \cdot f(x) & \text{for } y \cdot f(x) \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Squared hinge loss

$$L(y, f(x)) = \begin{cases} (1 - y \cdot f(x))^2 & \text{for } y \cdot f(x) \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Huberized squared hinge loss

$$L(y, f(x)) = \begin{cases} -4 y \cdot f(x) & \text{for } y \cdot f(x) \leq -1 \\ (1 - y \cdot f(x))^2 & \text{for } -1 \leq y \cdot f(x) \leq 1 \\ 0 & \text{otherwise} \end{cases}$$