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

Different loss functions (for classification)

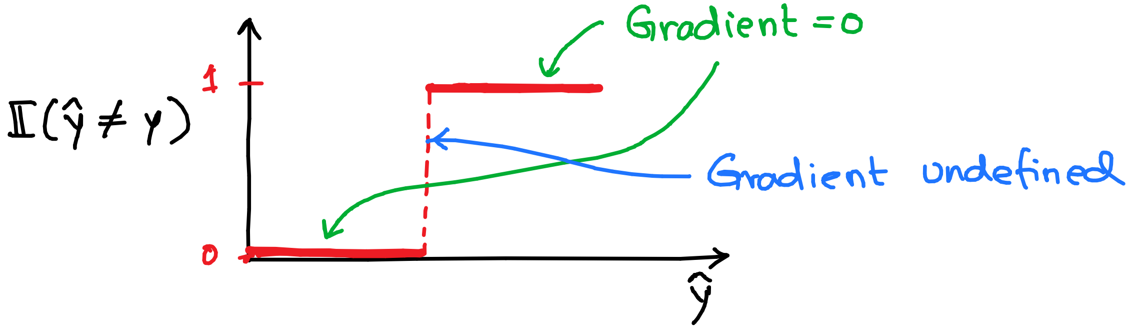
Let's look at loss functions for binary classification first

- 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 $\longrightarrow$ 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 $g(\underline{x})$

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

- 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

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

(for binary classes) $\{-1, 1\}$

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

$$\hat{y}(\underline{x}) = \text{sign}\{ f(\underline{x}; \underline{\theta}) \} \qquad \hat{y}(x) \begin{cases} \nearrow +1 \\ \searrow -1 \end{cases}$$

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 $y \cdot f(\underline{x})$

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

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

  - 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 $p(y=1 \mid \underline{x}) = g(\underline{x})$, then arrived at cross-entropy loss, and later for $g(x)$ 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

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

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

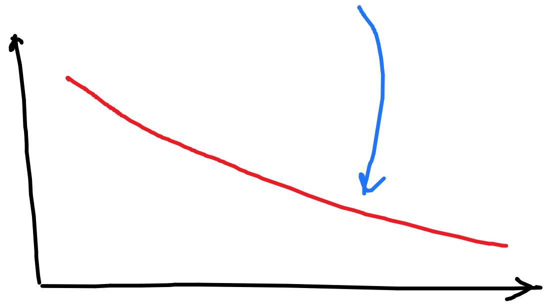
- Misclassification loss as margin-based loss

Misclassification loss

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

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

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



- e.g. Exponential loss

$$L(y, f(\underline{x})) = \exp\left(-y \cdot f(\underline{x})\right)$$

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

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

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

[no probabilistic interpretation possible]

- Squared hinge loss (has probabilistic interpretation)

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

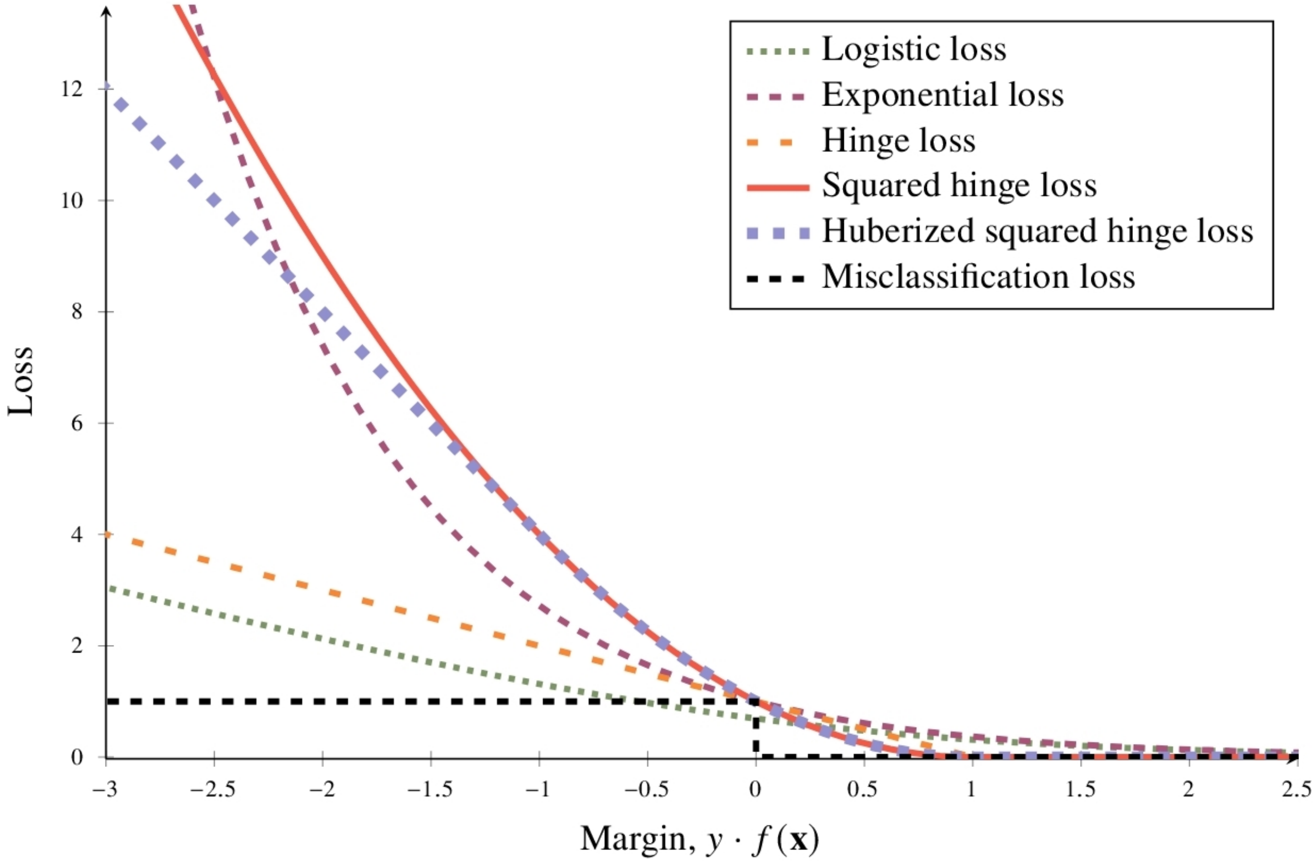
  - less robust to outliers

- Huberized squared hinge loss

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

Misclassification loss

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



Exponential loss

$$L(y, f(\underline{x})) = \exp\left(-y \cdot f(x)\right)$$

Hinge loss

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

Huberized squared hinge loss

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

Squared hinge loss

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

Regularization

- The idea of regularization in a parametric model is to
"keep the parameters $\underline{\hat{\theta}}$ small unless the data really convinces us otherwise"

- Two types of regularization

  → Explicit regularization, e.g L_2-regularization

  → Implicit regularization, e.g. early stopping

Explicit regularization

L_2-regularization	L_1-regularization (LASSO)

L_2-regularization

$$\hat{\underline{\theta}} = \arg\min_{\underline{\theta}} \frac{1}{N} \| \underline{y} - \underline{\underline{X}}\,\underline{\theta} \|_2^2 + \lambda \| \underline{\theta} \|_2^2$$

- Admits closed-form solution

$$\hat{\underline{\theta}} = \left(\underline{\underline{X}}^T \underline{\underline{X}} + N\lambda \underline{\underline{I}} \right)^{-1} \underline{\underline{X}}^T \underline{y}$$

- Typically does not produce sparse solution

L_1-regularization (LASSO)

$$\hat{\underline{\theta}} = \arg\min_{\underline{\theta}} \frac{1}{N} \| \underline{y} - \underline{\underline{X}}\,\underline{\theta} \|_2^2 + \lambda \| \underline{\theta} \|_1$$

$$\| \underline{\theta} \|_1 = |\theta_0| + |\theta_1| + \cdots + |\theta_p|$$

- No closed-form solution available

  Have to do numerical optimization

- Produces sparse solutions, where only a few of the parameters are non-zero

  In a sense, L_1-regularization can "switch-off" some inputs (by setting the corresponding parameter θ_k to zero)

Implicit Regularization

- There are alternative ways to achieve regularization without explicitly modifying the cost function

- One such way is Early Stopping

  ↳ aborting an iterative numerical optimization before it has reached the minimum of the cost function

- Set aside some hold-out validation data for computing $E_{\text{hold-out}}$ and use it to determine the stopping point

$E_{\text{hold-out}}$ ← Validation error

E_{train} ← training error