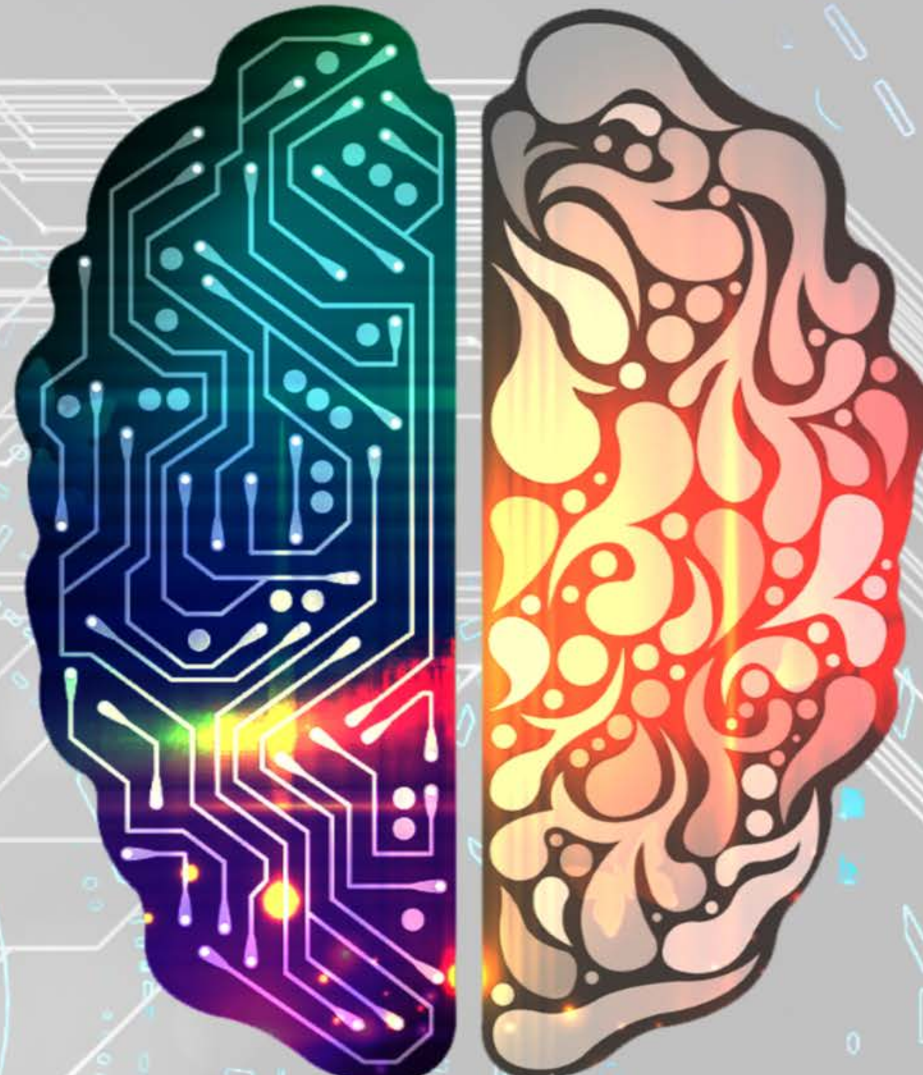
CMPS 460 – Spring 2022

MACHINE LEARNING

Tamer Elsayed



7.a

Linear Models: Optimization Framework

Chapter 7: 7.1-7.2

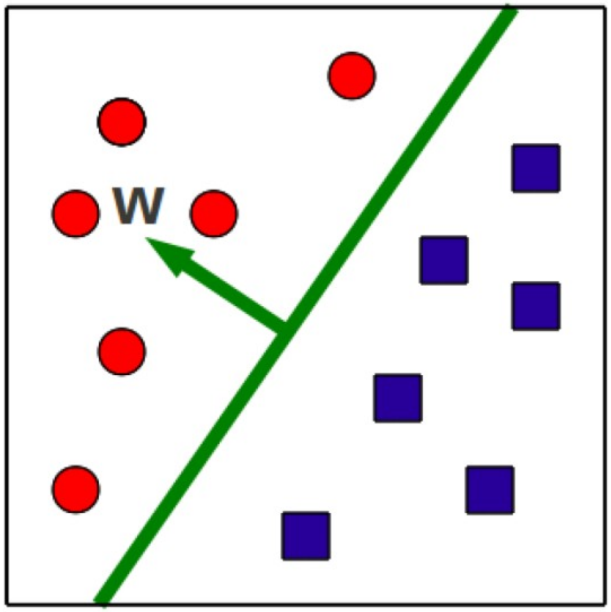
Roadmap …

- Linear Models
  - Loss functions
  - Regularization
- Gradient Descent
- Calculus refresher
  - Convexity
  - Gradients & subgradients

Binary Classification via Hyperplanes

- A classifier is a hyperplane (w, b)
- At test time, we check on what side of the hyperplane examples fall

$$\hat{y} = sign(w^T x + b)$$



- This is a **linear model**
  - Because the prediction is a linear combination of feature values x.

Recall: Perceptron Algorithm

Algorithm 5 PERCEPTRONTRAIN($\mathbf{D}$, $MaxIter$)

$w_d \leftarrow 0$, for all $d = 1 \dots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \dots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \dots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \dots, w_D, b$

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

$a \leftarrow \sum_{d=1}^{D} w_d \, \hat{x}_d + b$ // compute activation for the test example
return SIGN(a)

Perceptron is a ***model*** and ***algorithm*** in one

Let's separate model definition from training algorithm

Optimization Framework for Linear Models

Learning a Linear Classifier as an *Optimization Problem*

Optimization Framework

Objective function

Loss function measures how well classifier fits training data

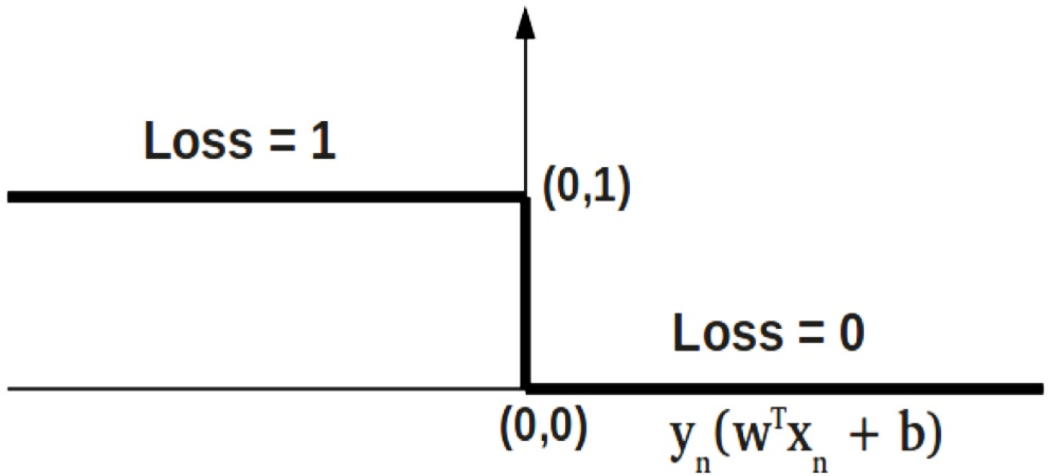
$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0)$$

0-1 loss

$\mathbb{I}(.)$: Indicator function: 1 if (.) is true, 0 otherwise

- **Two problems!**
  1. **If linearly inseparable, no efficient optimization**
     - The 0-1 loss above is NP-hard to optimize exactly/approximately in general

The 0-1 Loss Function



- Small changes in w, b can lead to big changes in the loss value
- 0-1 loss is non-smooth, non-convex

Optimization Framework

Objective function

Loss function measures how well classifier fits training data

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0)$$

0-1 loss

$\mathbb{I}(.)$: Indicator function: 1 if (.) is true, 0 otherwise

- **Two problems!**
  1. If linearly inseparable, no efficient optimization
     - The 0-1 loss above is NP-hard to optimize exactly/approximately in general
  2. **Subject to overfitting**

Solution:

Different loss functions **&** ***regularizers***

Optimization Framework

Objective function

Loss function measures how well classifier fits training data

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T \cancel{\mathbf{x}_n} + b) < 0)$$

$$l(y_n, \hat{y}_n)$$

- Different loss function approximations
  - easier to optimize

Optimization Framework

Objective function

Loss function measures how well classifier fits training data

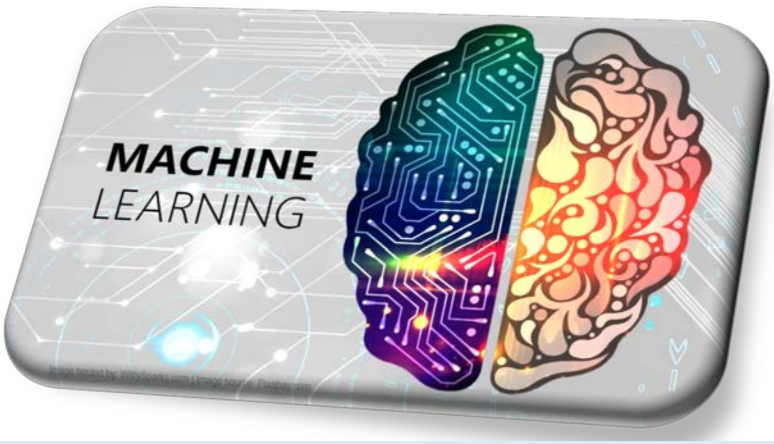
Regularizer prefers solutions that generalize well

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \cancel{\mathbb{I}(y_n(\mathbf{w}^T x_n + b) < 0)} + \lambda R(\mathbf{w}, b)$$

$l(y_n, \hat{y}_n)$

λ: parameter that controls the importance of the regularization term

- Different loss function approximations
  - easier to optimize
- Regularizer
  - prevents overfitting/prefers simple models.

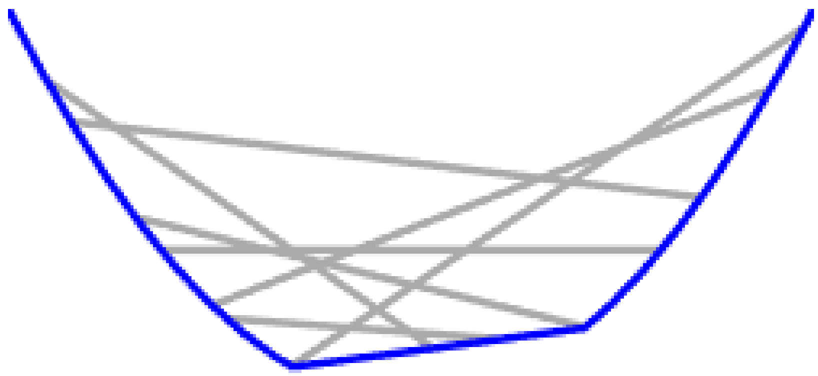


Convex Surrogate Loss Functions

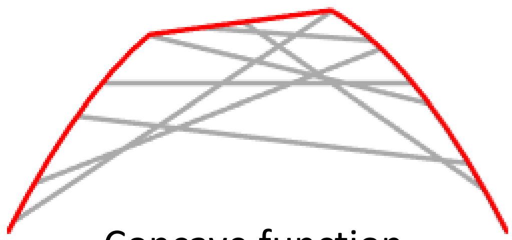
Convex vs. Non-Convex Functions

- A line segment between any two points on the graph of the function lies above or on the graph.
- Convex functions are easy to minimize.

Convex function



Non-convex functions



Concave function

Searching for
convex surrogate loss functions …

Surrogate Loss …

- 0-1 loss is hard to optimize
  ➔ optimize something else
- Convex functions are easy to optimize
  ➔ approximate 0-1 loss with a convex function
- This approximating function will be called a **surrogate loss**.
- The surrogate losses we construct will always be upper bounds on the true loss function.

Surrogate loss functions

- 0-1 Loss

  $$\ell^{(0/1)}(y,\hat{y}) = \mathbf{1}[y\hat{y} \le 0]$$

- Hinge Loss

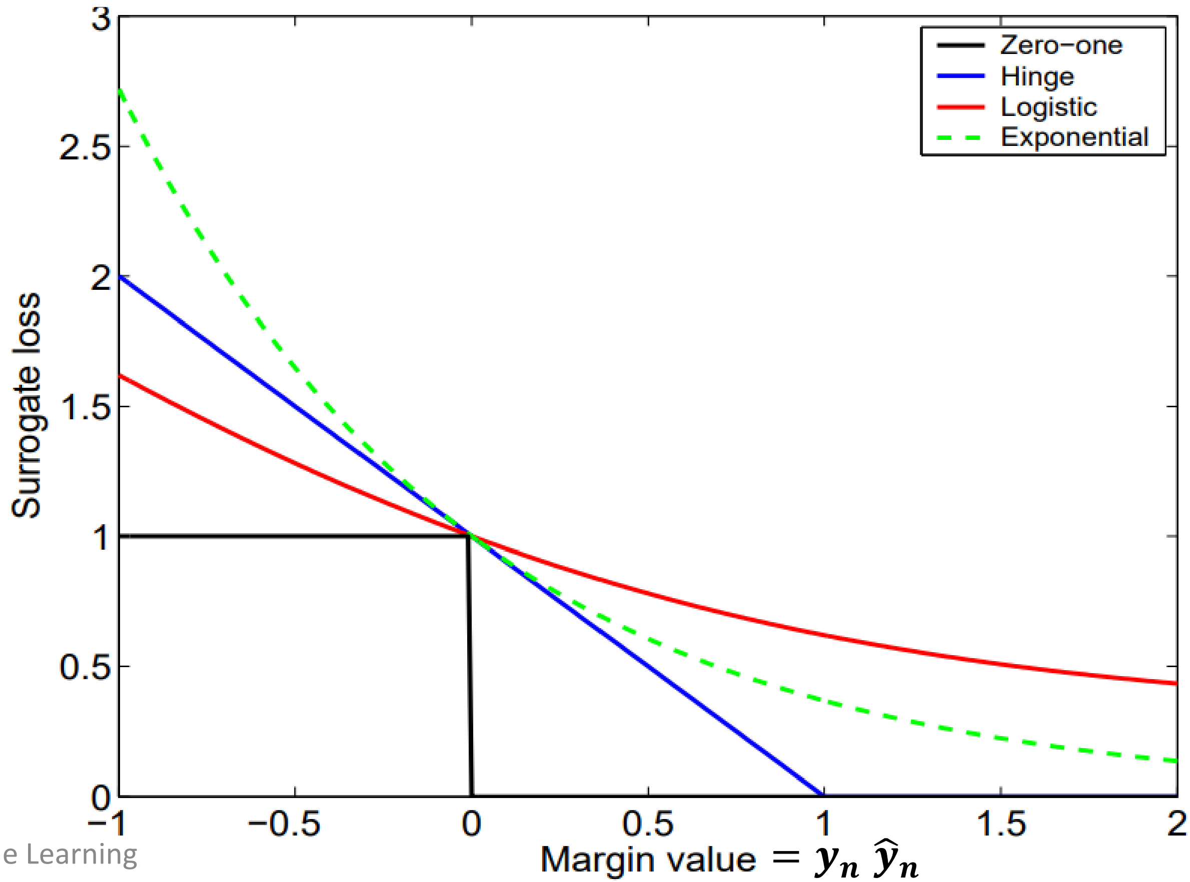
  $$\ell^{(\text{hin})}(y,\hat{y}) = \max\{0, 1 - y\hat{y}\}$$

- Logistic Loss

  $$\ell^{(\log)}(y,\hat{y}) = \frac{1}{\log 2}\log\left(1 + \exp[-y\hat{y}]\right)$$

- Exponential loss

  $$\ell^{(\exp)}(y,\hat{y}) = \exp[-y\hat{y}]$$



Surrogate loss functions

- 0-1 Loss

$$\ell^{(0/1)}(y, \hat{y}) = \mathbf{1}[y\hat{y} \le 0]$$

- Hinge Loss

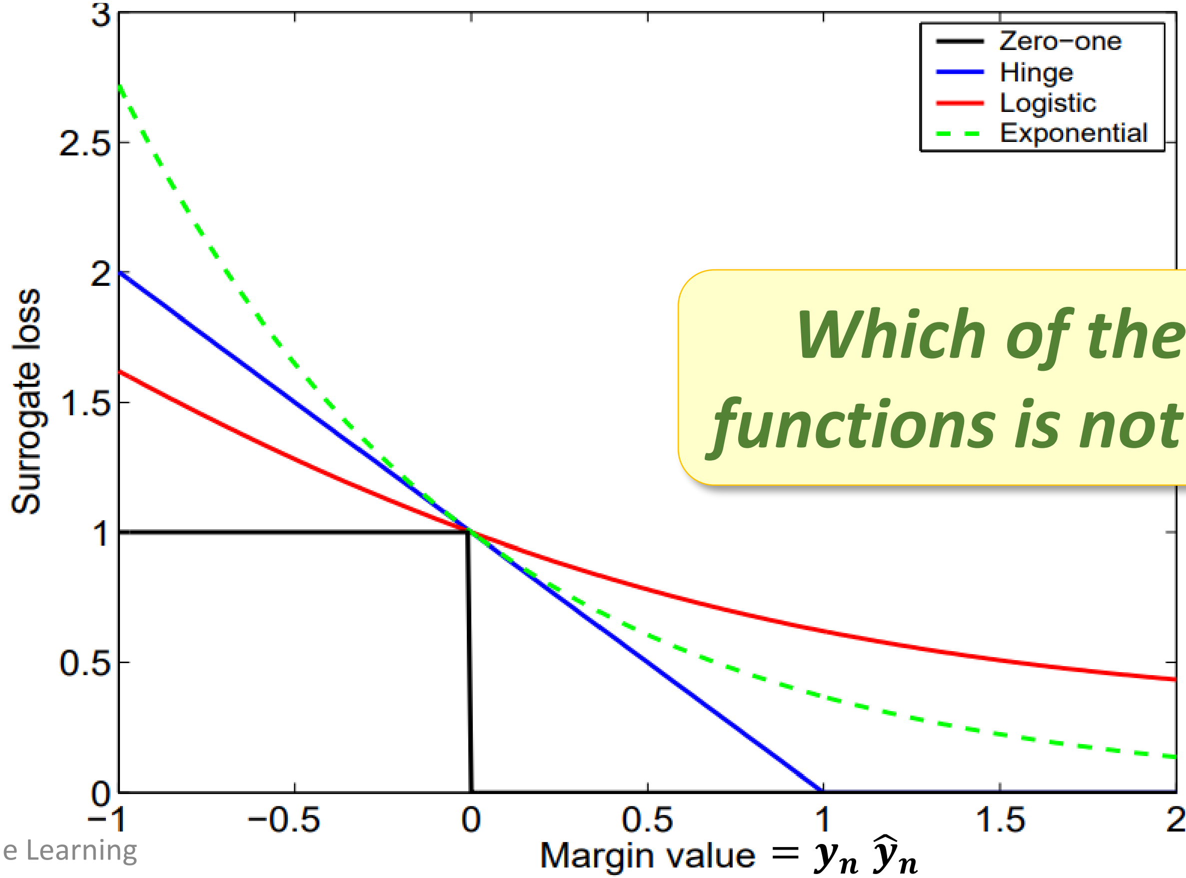
$$\ell^{(\text{hin})}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$$

- Logistic Loss

$$\ell^{(\log)}(y, \hat{y}) = \frac{1}{\log 2} \log\left(1 + \exp[-y\hat{y}]\right)$$

- Exponential loss

$$\ell^{(\exp)}(y, \hat{y}) = \exp[-y\hat{y}]$$



Which of these loss functions is not smooth?

Surrogate loss functions

- 0-1 Loss

$$\ell^{(0/1)}(y,\hat{y}) = \mathbf{1}[y\hat{y} \leq 0]$$

- Hinge Loss

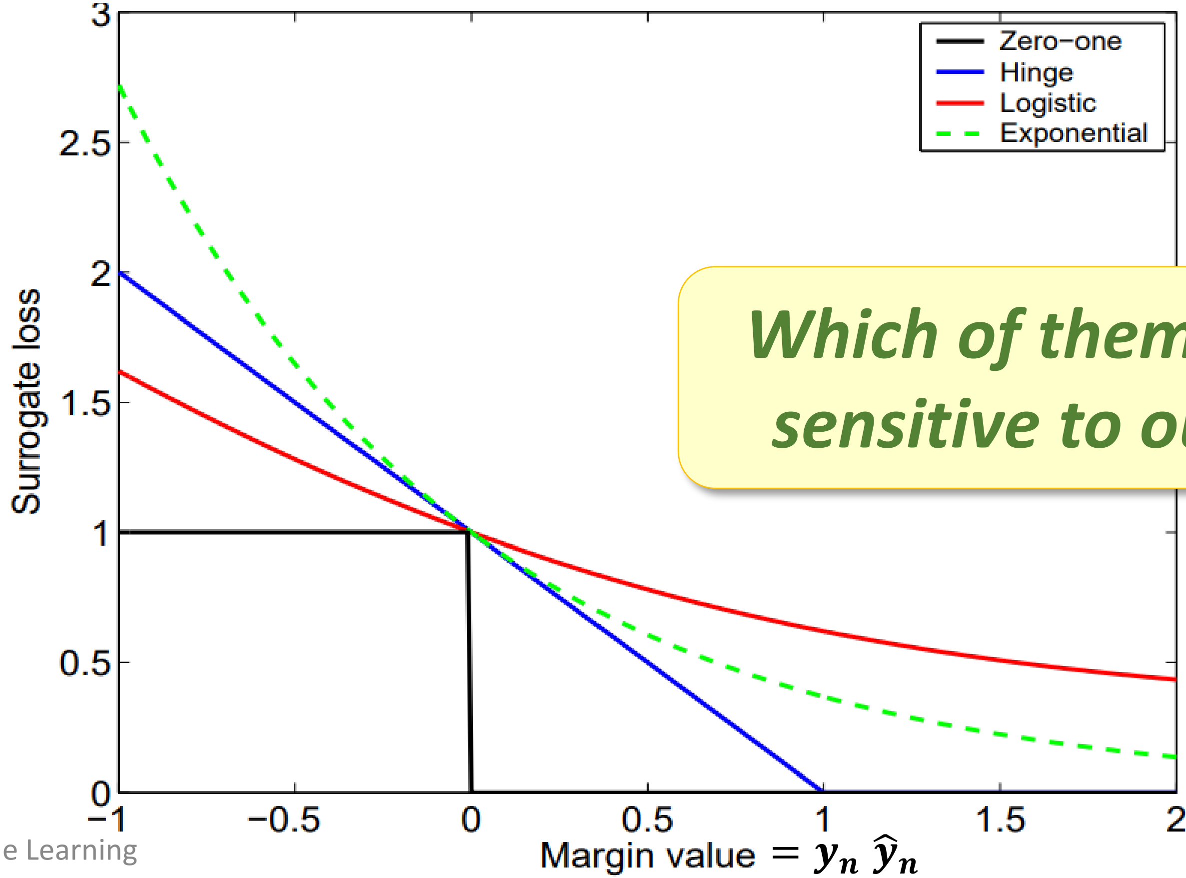
$$\ell^{(\text{hin})}(y,\hat{y}) = \max\{0, 1 - y\hat{y}\}$$

- Logistic Loss

$$\ell^{(\log)}(y,\hat{y}) = \frac{1}{\log 2}\log\left(1 + \exp[-y\hat{y}]\right)$$

- Exponential loss

$$\ell^{(\exp)}(y,\hat{y}) = \exp[-y\hat{y}]$$



Which of them is most sensitive to outliers?