Training loss vs. prediction loss

Training loss: $J_{\underline{\lambda}}(\underline{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(h_{\underline{\lambda}}(x_i; \underline{\theta}), y_i)$

($\underline{\lambda}$: hyperparameters; $\underline{\theta}$: tuning parameters)

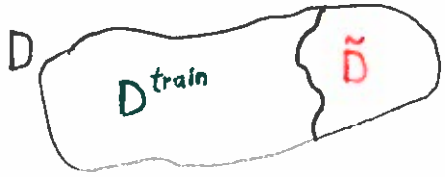
Prediction loss: $J(h_{\hat{\underline{\lambda}}}(.; \hat{\underline{\theta}})) = \mathbb{E}_{\underline{x},y}[\mathcal{L}(h_{\hat{\underline{\lambda}}}(\underline{x}; \hat{\underline{\theta}}), y)]$

Expected prediction loss: $\bar{J}(\mathcal{A}) = \mathbb{E}_{D^{train}}[J(\mathcal{A}(D^{train}))]$

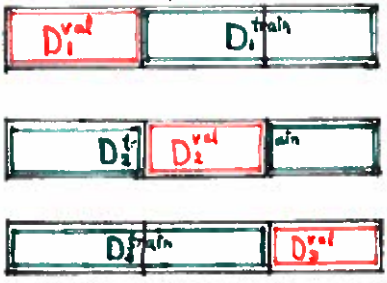
Goal: Avoid overfitting and underfitting

Estimate prediction loss / expected prediction loss

Prediction loss: held-out set



Expected prediction loss: cross validation



$\hat{\bar{J}}(\mathcal{A}) = CV$

Estimating generalisation performance with hyperparameter selection

Idea: - Nest method for estimating generalisation performance to avoid overfitting

- Use held-out or CV in hyperparameter optimisation:

  For given hyperparameters $\underline{\lambda}$, estimate generalisation performance, then we select best $\underline{\lambda}$

  ⇝ now $\underline{\lambda}$ can overfit, so yet another held-out set necessary

Algorithm:

1. Split D into D^{hyp} and D^{test} (e.g. 80:20)
   Lock away D^{test} until step 4.

2. Optimise hyperparameters $\underline{\lambda}$ on D^{hyp}:

Held-out

Split D^{hyp} into D^{train} and D^{val}

$\hat{\underline{\lambda}} = \arg\min_{\underline{\lambda}} PL(\underline{\lambda})$

where

$PL(\underline{\lambda}) = \hat{J}(\hat{h}_{\underline{\lambda}}, D^{val})$

and $\hat{h}_{\underline{\lambda}} = \mathcal{A}(D^{train})$

Cross validation

Split D^{hyp} into k folds and compute CV for each $\underline{\lambda}$

$\hat{\underline{\lambda}} = \arg\min_{\underline{\lambda}} EPL(\underline{\lambda})$

where

$EPL(\underline{\lambda}) = CV_{\underline{\lambda}} = \hat{J}(\mathcal{A}_{\underline{\lambda}})$

3. Reestimate $\underline{\theta}$ using $\hat{\underline{\lambda}}$: $\hat{h} = \mathcal{A}_{\hat{\underline{\lambda}}}(D^{hyp})$

4. Estimate prediction loss: $\hat{J} = \hat{J}(\hat{h}; D^{test})$

5. Reestimate $\hat{h}$ on all data: $\hat{h} = \mathcal{A}_{\hat{\lambda}}(D)$

Do not estimate prediction loss on all data ▽

"Two-times held out"

"CV and held-out"

Loss functions

- Regression:

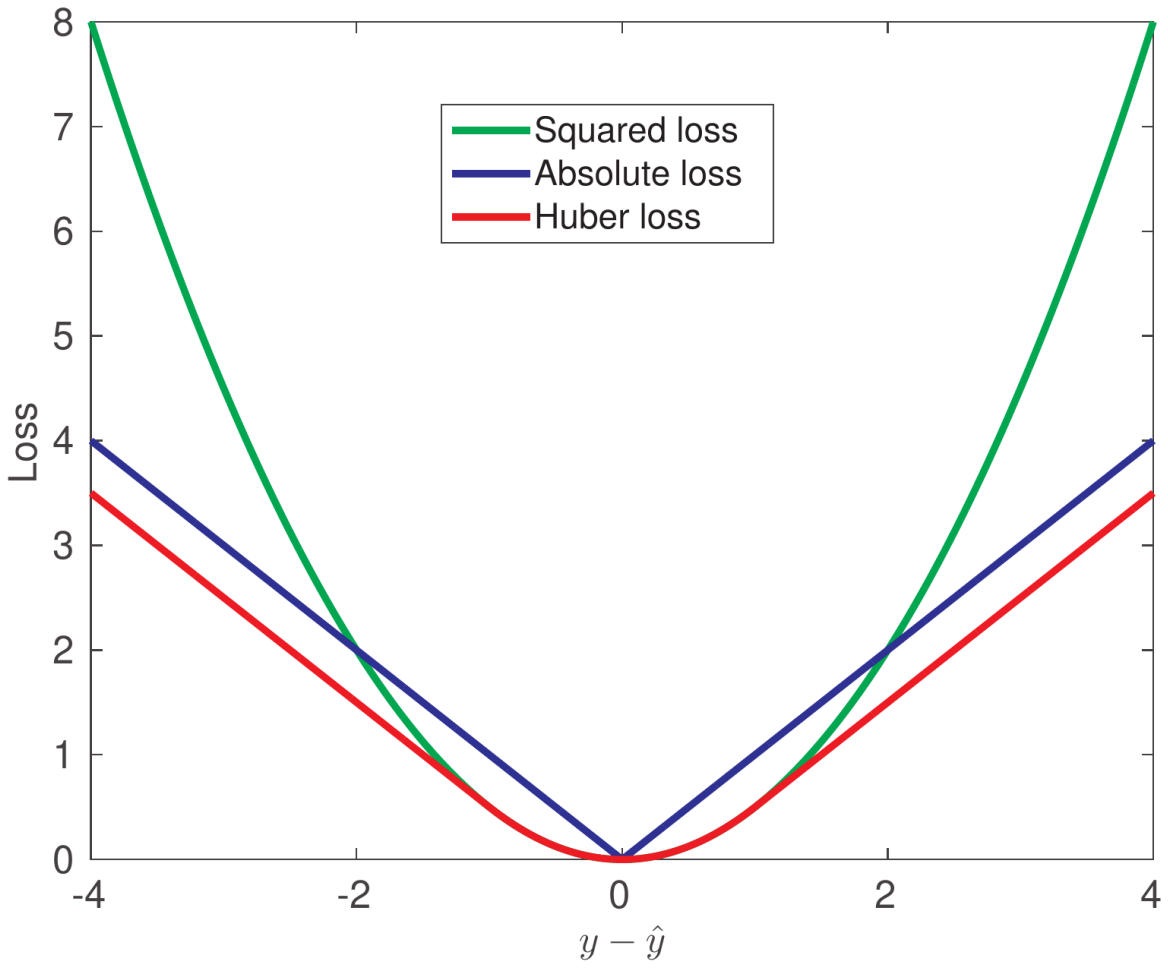
$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$ "square loss"

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

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

"Huber loss"

Loss Functions



– Classification loss functions

For K different classes $\{1, \dots, K\}$

loss function $L(\hat{y}, y) \rightsquigarrow$ matrix

$$L = \begin{pmatrix} L(1,1) & L(1,2) \cdots & L(1,K) \\ L(2,1) & \ddots & \vdots \\ \vdots & & \\ L(K,1) & \cdots & L(K,K) \end{pmatrix}$$

$L(i,i)$ typically zero, other entries > 0

Special case: "Zero-one loss"

$$L(i,j) = \begin{cases} 0 & ; i = j \\ 1 & ; i \neq j \end{cases}$$

Expectation:

$$J(h) = \mathbb{E}_{\underline{x},y}\left[L(h(\underline{x}), y)\right] = \mathbb{E}_{\hat{y},y}\left[L(\hat{y}, y)\right]$$

$$= \sum_{i,j} p(i,j) L(i,j) = \sum_{i \neq j} p(i,j) = P(\hat{y} \neq y)$$

$$= P(h(\underline{x}) \neq y)$$

"misclassification" or "error rate"

For two classes (i.e. $K=2$): let $\hat{y}, y \in \{-1, 1\}$

"true positive rate": $p(\hat{y}=1 | y=1) = \frac{p(\hat{y}=1, y=1)}{p(y=1)}$

("sensitivity", "hit rate", "recall")

"true negative rate": $p(\hat{y}=-1 | y=-1) = \frac{p(\hat{y}=-1, y=-1)}{p(y=-1)}$

("specificity")

"false positive rate": $p(\hat{y}=1 | y=-1) = \ldots$

("type 1 error")

"false negative rate": $p(\hat{y}=-1 | y=1) = \ldots$

("type 2 error")

Loss function that penalises false positive and false negative rates:

$$L = \begin{pmatrix} 0 & \frac{1}{p(y=-1)} \\ \frac{1}{p(y=1)} & 0 \end{pmatrix}$$

Then:

$$J(h) = \sum_{i,j} p(i,j) L(i,j)$$

$$= \frac{p(1,-1)}{p(y=-1)} + \frac{p(-1,1)}{p(y=1)}$$

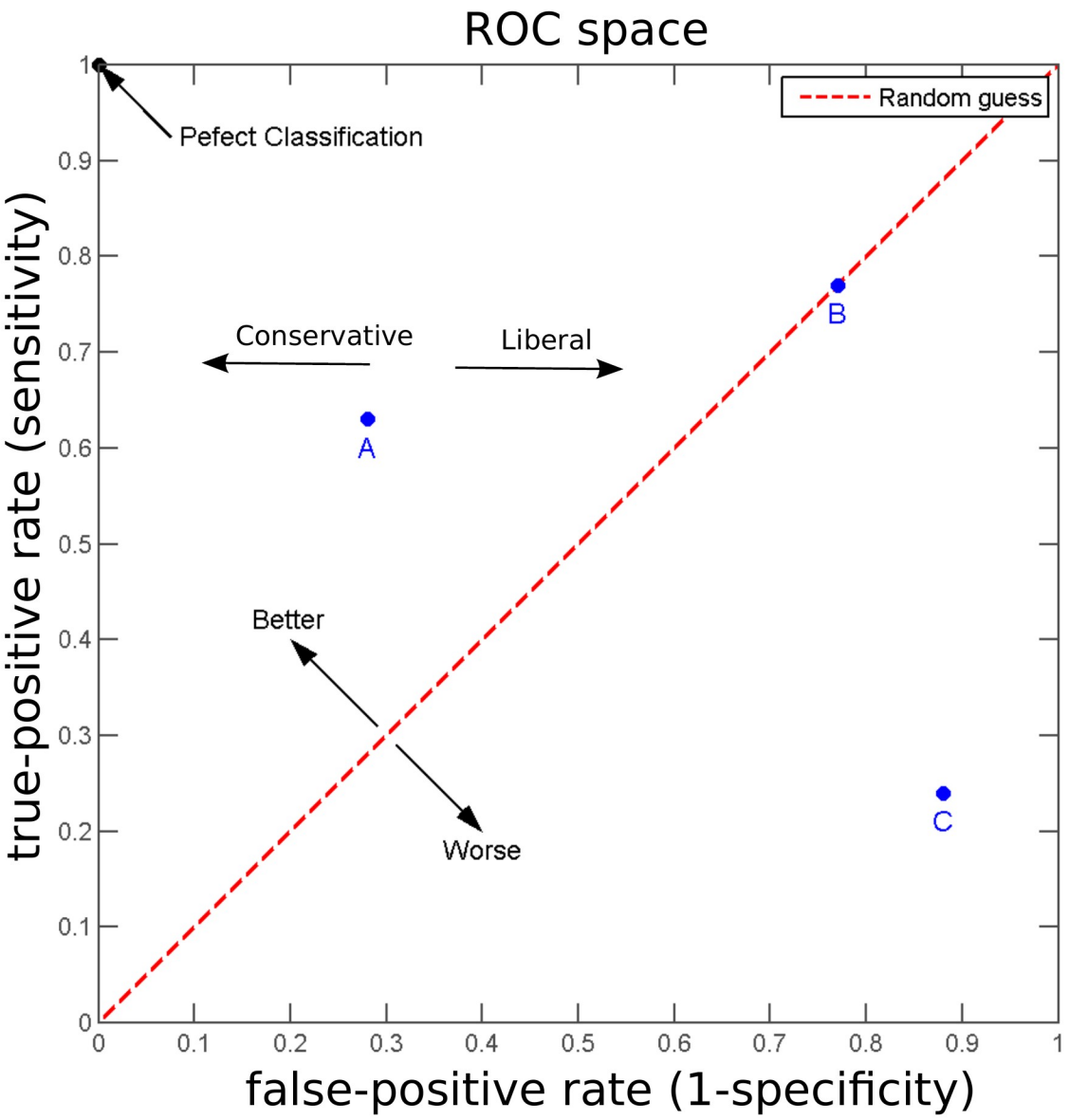
$$= p(\hat{y}=1 | y=-1) + p(\hat{y}=-1 | y=1)$$

Useful for imbalances in $y=1$, $y=-1$

Generally, trade-off between true-positive and false-negative rates

Visualised by "receiver-operating characteristic" (ROC)

Receiver Operating Characteristic



Differentiable loss functions for classification

Here: binary classification only

$\hat{y}(\underline{x}) = \text{sign}(h(\underline{x})) \in \{-1, 1\}$

where h real-valued

$\rightsquigarrow$ correct classification of $\underline{x} \Leftrightarrow y\,h(\underline{x}) > 0$

$y\,h(\underline{x})$ "margin", similar to $y - h(\underline{x})$ in regression

Zero-one loss recovered for $L(h(\underline{x}), y) = \begin{cases} 1 & ; y\,h(\underline{x}) < \\ 0 & ; \text{otherwi} \end{cases}$

Logistic regression:

$p(y=1 \mid \underline{x}; h) = \frac{1}{1+\exp(-h(\underline{x}))}$ $p(y=-1 \mid \underline{x}; h) = \frac{1}{1+\exp(h($

Relation to loss:

maximise log-likelihood $\Leftrightarrow$ minimisation log-loss

Classification Loss Functions that Operate on Margin

$$L(h(\boldsymbol{x}), y) = (h(\boldsymbol{x}) - y)^2 = (1 - yh(\boldsymbol{x}))^2 \quad \text{(square loss)}$$

$$L(h(\boldsymbol{x}), y) = \log(1 + \exp(-yh(\boldsymbol{x}))) \quad \text{(log loss)}$$

$$L(h(\boldsymbol{x}), y) = \exp(-yh(\boldsymbol{x})) \quad \text{(exponential loss)}$$

$$L(h(\boldsymbol{x}), y) = \max(0, 1 - yh(\boldsymbol{x})) \quad \text{(hinge loss)}$$

$$L(h(\boldsymbol{x}), y) = \max(0, 1 - yh(\boldsymbol{x}))^2 \quad \text{(square hinge loss)}$$

$$L(h(\boldsymbol{x}), y) = \begin{cases} -4yh(\boldsymbol{x}) & \text{if } yh(\boldsymbol{x}) < -1 \\ \max(0, 1 - yh(\boldsymbol{x}))^2 & \text{otherwise} \end{cases} \quad \text{(Huberised square hinge loss)}$$