

Last lecture: Classification

- regression, density estimation, kNN, logistic regression

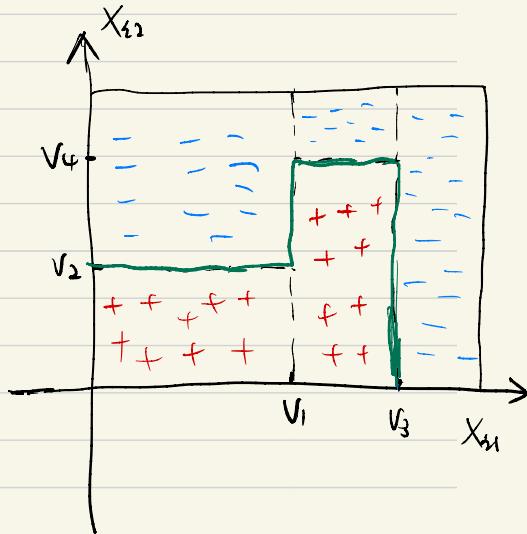
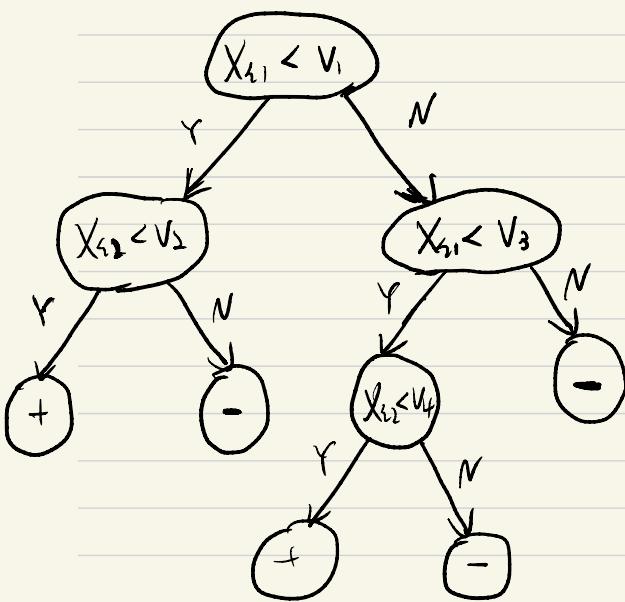
Decision Tree

$$\{(X_i, Y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0, 1\}$$

↑
features / predictors

$$X_i = (X_{i1}, X_{i2})$$

$$\underline{d=2}$$



$$X^* = (X_1^*, X_2^*)$$

$$X_1^* < v_1, X_2^* \geq v_2$$

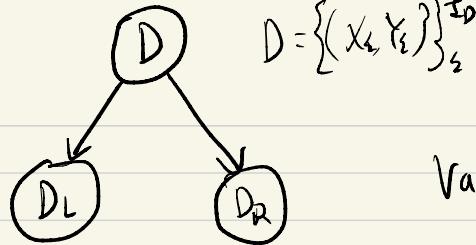
$$\hat{Y}^* \leftarrow -$$

① How to determine the split?

↳ regression tree: use "purity", find a split that

maximizes

$$|D| \cdot \text{Var}(D) - [|D_L| \cdot \text{Var}(D_L) + |D_R| \cdot \text{Var}(D_R)]$$



$$D = \{(X_i, Y_i)\}_{i=1}^{|D|}$$

$|D| = \# \text{ of samples}$

$$\text{Var}(D) = \frac{1}{|D|} \sum_{i \in D} (Y_i - \bar{Y})^2$$

↑ Variance of Y_i in D

↪ $Y_i \in \{0, 1, \dots, K-1\}$ classification

use { Gini-index / Gini-impurity
information gain, i.e.

$$\text{maximize } \text{IG}(Y|X) = H(Y) - H(Y|X)$$

• $H(Y)$ is the entropy of Y

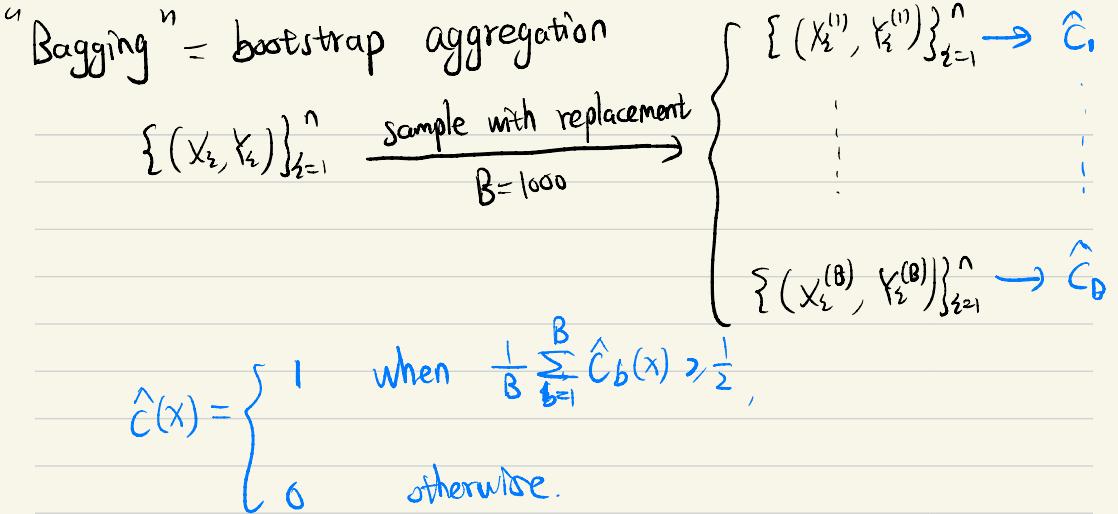
$$= - \sum_{j=0}^{K-1} p(Y_j) \cdot \log p(Y_j)$$

• $H(Y|X)$ is the conditional entropy of Y | X

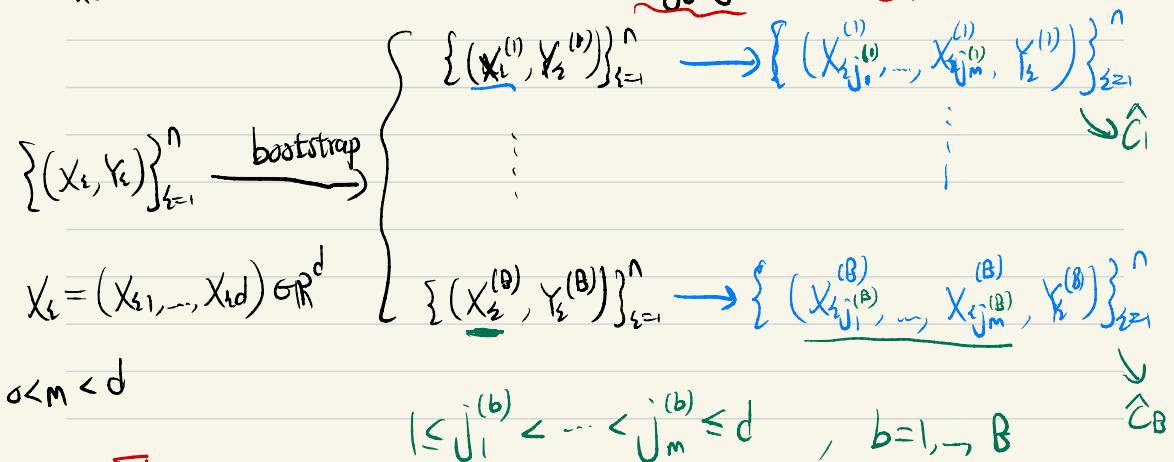
↪ greedy searching

② When to stop?

- When the leaf is "pure", $\text{Var}(D_L) < \underline{\epsilon}$ ← threshold
- When $|D| \leq \underline{\delta_N} = 20$
- Stop at a fixed depth $D = 100$



Random forest = Decision Tree + Bagging + Random Features



$$m = \sqrt{d}$$

$$\underbrace{(X_{i1}^{(1)}, X_{i2}^{(1)}, \dots, X_{id}^{(1)}, Y_i^{(1)})}_{d} \downarrow \quad m < d$$

$$X_{ij_1}^{(1)}, \dots, X_{ij_m}^{(1)}$$

$$\hat{C}(x) = \begin{cases} 1 & \text{when } \frac{1}{B} \sum_{b=1}^B \hat{C}_b(x) \geq \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

kernel-based random forest

Caveat: It is motivated by random forest regression but is NOT a random forest.

- $A_j(x)$ be the cell/leaf that x falls into in the j -th regression tree.

\Rightarrow The b -th regression tree is

$$\hat{m}_b(x) = \frac{\sum_{i=1}^n Y_i \cdot \mathbb{1}_{\{X_i \in A_b(x)\}}}{\sum_{j=1}^n \mathbb{1}_{\{X_j \in A_b(x)\}}}$$

\Rightarrow Random forest regression is

$$\begin{aligned}\tilde{m}(x) &= \frac{1}{B} \sum_{b=1}^B \hat{m}_b(x) \\ &= \frac{1}{B} \sum_{b=1}^B \frac{\sum_{i=1}^n Y_i \cdot \mathbb{1}_{\{X_i \in A_b(x)\}}}{\sum_{j=1}^n \mathbb{1}_{\{X_j \in A_b(x)\}}}\end{aligned}$$

$$= \sum_{i=1}^n \sum_{b=1}^B \frac{Y_i \cdot \mathbb{1}_{\{X_i \in A_b(x)\}}}{B \cdot \sum_{j=1}^n \mathbb{1}_{\{X_j \in A_b(x)\}}}$$

Obs: In some extreme cases, $A_b(x)$ is too small to contain any X_i .
(high prob.)

$$B \cdot \underbrace{\sum_{j=1}^n \mathbb{1}_{\{X_j \in A_b(x)\}}}_{I(x)} = 0 \quad (\text{not good})$$

$$B \cdot I(x) = \sum_{b=1}^B I(x) \Rightarrow \sum_{b=1}^B I_b(x) = \sum_{b=1}^B \sum_{j=1}^n \mathbb{1}_{\{x_j \in A_b(x)\}}$$

$$W_1, \dots, W_B, \quad B \cdot W_b \approx \sum_{b=1}^B W_b$$

$$\hat{M}_{K\text{eRF}}(x) = \sum_{i=1}^n \frac{\sum_{b=1}^B \mathbb{1}_{\{x_i \in A_b(x)\}}}{\sum_{b=1}^B \sum_{j=1}^n \mathbb{1}_{\{x_j \in A_b(x)\}}}$$

$$\triangleq \sum_{i=1}^n \frac{Y_i \cdot K(x, x_i)}{\sum_{j=1}^n K(x, x_j)}$$

$$K(x, u) = \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{\{u \in A_b(x)\}}$$

↑ Nadaraya-Watson kernel regression
(linear smoother)

Imbalanced classification

$$\text{. Imbalanced dataset} \quad \left\{ (x_i, y_i) \right\}_{i=1}^n \quad \left\{ y_i = 1 \right\} \quad \left\{ y_i = 0 \right\}$$

• $c: X \rightarrow Y = \{0, 1\}$ classifier

• $L: Y \times Y \rightarrow \mathbb{R}_{\geq 0}$ loss function

• $R(c) = \mathbb{E}[L(c(x), Y)]$ risk

• Bayes classifier $c^* = \operatorname{argmin}_c R(c) = \operatorname{argmin}_c \mathbb{E}[L(c(x), Y)]$

$$= \operatorname{argmin}_c \int [\mathbb{E}[L(c(x), Y)] | x=x] dP(x)$$

$$\text{0-1 loss} \rightarrow = \begin{cases} 1 & \text{when } P(Y=1|x) \geq P(Y=0|x) \\ 0 & \text{otherwise.} \end{cases}$$

		Actual label	
		$\hat{Y}=0$	$\hat{Y}=1$
predicted label	$\hat{Y}=0$	n_{00}	n_{01}
	$\hat{Y}=1$	n_{10}	n_{11}

$$\text{Misclassification error} = \frac{n_{01} + n_{10}}{n_{00} + n_{01} + n_{10} + n_{11}}$$

n_{00} = True negative , n_{11} = True positive , n_{10} = false positive

n_{01} = false negative

- False positive rate (FPR) = $\frac{FP}{N} = \frac{n_{10}}{n_{10} + n_{00}}$ ↗ Type I error rate
- False negative rate (FNR) = $\frac{FN}{P} = \frac{n_{01}}{n_{01} + n_{11}}$ ↗ Type II error rate
- Recall = $\frac{TP}{P} = 1 - FNR$ (focus on the " $\hat{Y}=1$ " class)
- False discovery rate = $\frac{FP}{PP}$ (focus on the " $\hat{Y}=1$ " class)
 (FDR)
 $= 1 - \underline{\text{precision}}$
- $F1\text{-score} = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = \frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$

Q: Why do we need all these additional metrics?

$$N=1000, |\{Y_i=1\}|=10, |\{Y_i=0\}|=990$$

$C_{\text{naive}}(x) = 0 \implies \text{misclassification error of } C_{\text{naive}} = \frac{10}{1000} = 1\%$

recall of $C_{\text{naive}} = \frac{0}{10} = 0$ $\frac{0}{0} = 1$

precision of $C_{\text{naive}} = 1 - \text{FDR} = 1 - \frac{0}{0} = 0$

Example (finance) $Y = \begin{cases} 1 & \text{fraudulent transaction,} \\ 0 & \text{benign transaction.} \end{cases}$

- $C_{\text{naive}}(x) = 0$ is problematic
- Missing a true fraud can cause some severe financial losses.

Q: Would Bayes classifier $C^*(x)$ work here?

$$C^*(x) = \begin{cases} 1 & \text{when } P(Y=1|x) \geq P(Y=0|x), \\ 0 & \text{when } P(Y=1|x) < P(Y=0|x). \end{cases}$$
$$= \begin{cases} 1 & \text{when } P(x|Y=1) \cdot P(Y=1) \geq P(x|Y=0) \cdot P(Y=0), \\ 0 & \text{when } P(x|Y=1) \cdot P(Y=1) < P(x|Y=0) \cdot P(Y=0). \end{cases}$$

The estimated classifier will be

$$\hat{C}(x) = \begin{cases} 1 & \text{when } \hat{P}(x|Y=1) \cdot \hat{P}(Y=1) \geq \hat{P}(x|Y=0) \cdot \hat{P}(Y=0), \\ 0 & \text{when } \hat{P}(x|Y=1) \cdot \hat{P}(Y=1) < \hat{P}(x|Y=0) \cdot \hat{P}(Y=0) \end{cases}$$

≈ 0

with high prob.

$$\hat{P}(Y=0) = \frac{990}{1000} = 0.99$$

Standard 0-1 loss and its associated Bayes classifier are not suitable for imbalanced dataset!

Solution: Use a weighted 0-1 loss:

$$L((c(x), Y)) = w_1 \cdot \mathbb{1}_{\{c(x)=0, Y=1\}} + w_2 \cdot \mathbb{1}_{\{c(x)=1, Y=0\}}, \quad w_1 > w_2$$

e.g. $w_1 = 100$

$w_2 = 1$

$$\mathbb{E}[L_w(c(x), Y) | X=x] = w_1 \cdot \mathbb{1}_{\{c(x)=0\}} \cdot P(Y=1 | X=x)$$

$$+ w_2 \cdot \mathbb{1}_{\{c(x)=1\}} \cdot P(Y=0 | X=x)$$

$$= \begin{cases} w_1 \cdot P(Y=1 | X=x) & \text{if } c(x)=0, \\ w_2 \cdot P(Y=0 | X=x) & \text{if } c(x)=1. \end{cases}$$

Bayes classifier

$$c^*(x) = \begin{cases} 0 & \text{when } w_1 \cdot P(Y=1 | X=x) \leq w_2 \cdot P(Y=0 | X=x), \\ 1 & \text{when } w_1 \cdot P(Y=1 | X=x) > w_2 \cdot P(Y=0 | X=x). \end{cases}$$

$$= \begin{cases} 0 & \text{when } w_1 \cdot P(Y=1) \cdot P(X|Y=1) \leq w_2 \cdot P(X|Y=0) \cdot P(Y=0), \\ 1 & \text{when } w_1 \cdot P(Y=1) \cdot P(X|Y=1) > w_2 \cdot P(Y=0) \cdot P(X|Y=0). \end{cases}$$

$w_1 > w_2$

Optimization perspective of classification

$$c^* = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \mathbb{E} [L(c(x), Y)]$$

\mathcal{F} is a class of candidate classifiers

\Rightarrow Empirical risk minimization

$$\hat{c}_n = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{s=1}^n L(c(x_s), Y_s) \right] = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \hat{R}_n(c)$$

- Unbiasedness: $\mathbb{E} [\hat{R}_n(c)] = \mathbb{E} \left[\frac{1}{n} \sum_{s=1}^n L(c(x_s), Y_s) \right] = R(c)$

- generalization: $c^* = \underset{c \in \mathcal{F}}{\operatorname{argmin}} R(c)$, $\hat{c}_n = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \hat{R}_n(c)$

Ideally, we want the excess risk $R(\hat{c}_n) - R(c^*)$ to be small.

$$R(\hat{c}_n) - R(c^*) = R(\hat{c}_n) - \hat{R}_n(\hat{c}_n) + \underbrace{\hat{R}_n(\hat{c}_n) - \hat{R}_n(c^*)}_{\leq 0} + \hat{R}_n(c^*) - R(c^*)$$

≤ 0 because \hat{c}_n is a minimizer of \hat{R}_n

$$\leq |R(\hat{c}_n) - \hat{R}_n(\hat{c}_n)| + 0 + |\hat{R}_n(c^*) - R(c^*)|$$

$$\leq 2 \sup_{c \in \mathcal{F}} |R(c) - \hat{R}_n(c)|$$

In order for $\hat{c}_n = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \hat{R}_n(c)$ to generalize well, we need

$$\sup_{c \in \mathcal{F}} |R(c) - \hat{R}_n(c)| = o_p(1)$$

- Case 1: if $|\mathcal{F}|$ is finite, then

$$\sup_{c \in \mathcal{F}} |\hat{R}_n(c) - R(c)| = O_p\left(\sqrt{\frac{\log |\mathcal{F}|}{n}}\right) \text{ by Hoeffding's inequality.}$$

- Case 2: if $|\mathcal{F}| = \infty$ but \mathcal{F} has a finite VC-dimension V ,

then

$$\sup_{c \in \mathcal{F}} |\hat{R}_n(c) - R(c)| = O_p\left(\sqrt{\frac{V \cdot \log n}{n}}\right)$$

e.g. for linear regression / classifier, VC-dimension = $O(\text{dimension})$

for neural networks, a fully connected NN with width W and depth H has

$$\text{VC-dimension} = \Theta(W \cdot H)$$

Peter Bartlett et al. 2017

"Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks" COLT 2017

VC-dimension bounds are not useful when $W \cdot H \gtrsim n$!!

- Rademacher complexity

For a given classifier c , $\hat{R}_n(c) = E_{\mathcal{F}} \sup_{c \in \mathcal{F}} \sum_{i=1}^n f_i \cdot c(x_i)$

where $\xi_i = \begin{cases} 1 & \text{with prob. } \frac{1}{2}, \\ -1 & \text{with prob. } \frac{1}{2}. \end{cases}$

ξ_1, \dots, ξ_n are IID.

The population-level Rademacher complexity is defined as

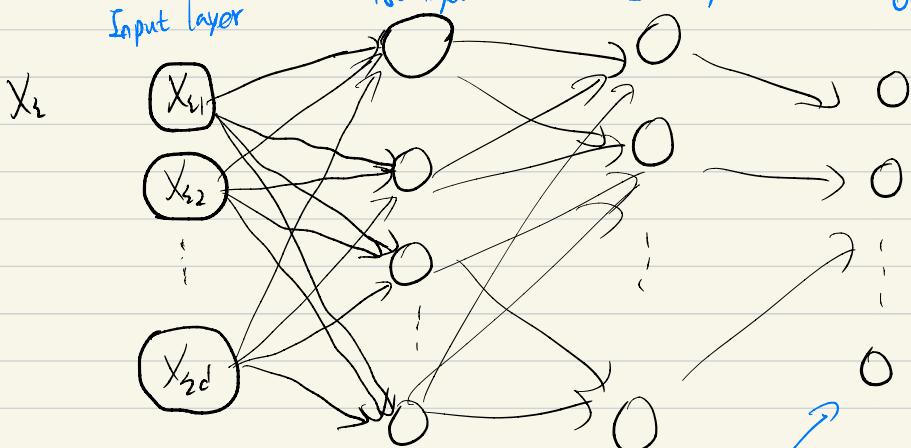
$$RC(c) = \underbrace{\mathbb{E}\left[\hat{RC}_n(c) \right]}_{\text{over } X_1, \dots, X_n}$$

$$\sup_{c \in \mathcal{G}} |\hat{R}_n(c) - R(c)| = O_p\left(\frac{RC(c)}{n} + \sqrt{\frac{\log \log n}{n}}\right).$$

Neural network classification

$$\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathcal{Y} \quad \mathcal{Y} = \{0, 1, \dots, k-1\}$$

Input layer 1st layer 2nd layer output layer



K output neurons in total

$$x_i \in \mathbb{R}^d, \quad w_L \in \mathbb{R}^{d \times d_{L-1}}, \quad L=1, \dots, H \quad \text{and} \quad d_0 = d$$

$$b_L \in \mathbb{R}^{d_{L-1}}$$

$$d_H = k$$

$$f(x) = \sigma_L(w_L z_L + b_L), \quad z_L = \sigma_{L-1}(w_{L-1} z_{L-1} + b_{L-1}), \quad L=1, \dots, H$$

with $z_1 = x$

σ_L 's are some activation function

$$\text{ReLU} \quad \sigma(u) = \begin{cases} u & \text{when } u \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{sigmoid} \quad \sigma(u) = \frac{1}{1+e^{-u}}$$

For K -classes classification problem, the last layer will use the softmax function

$$\sigma_L(u) = \left(\frac{e^{u_1}}{\sum_{j=1}^K e^{u_j}}, \dots, \frac{e^{u_K}}{\sum_{j=1}^K e^{u_j}} \right) = (f(x)_1, \dots, f(x)_K)$$

\Rightarrow The output of this NN is $f(x) \in [0, 1]^K$, $\sum_{j=1}^K f(x)_j = 1$.

How do we train NN in practice?

$$Y_i \in \{0, 1, \dots, K-1\} \quad \xrightarrow{\text{one-hot encoding}} \quad \tilde{Y}_i = (0, \dots, 1, \dots, 0)$$

$$\tilde{Y}_{i,j}=1 \text{ if } Y_i=j-1$$

Define the cross-entropy loss

$$L_c(f(x), \tilde{Y}) = - \sum_{j=1}^K \tilde{Y}_j \cdot \log [f(x)_j] \quad \sum_{j=1}^K f(x)_j = 1$$
$$f(x)_j \geq 0$$

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n L_c(f(x_i), Y_i)$$

$$= - \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^K \tilde{Y}_{ij} \cdot \log [f(x)_j]$$

- special case (binary classification): $Y_i \in \{0, 1\}$

$$L_c(f(x), Y) = -Y \cdot \log f(x) - (1-Y) \log [1-f(x)]$$

↳ This is indeed the log-likelihood of a binomial distribution.

$$Y_i | X_i \sim \text{Bernoulli}(p_i) \quad \text{with } p_i = f(x_i)$$

$$\hat{R}_n(f) = - \frac{1}{n} \sum_{i=1}^n \left(Y_i \log f(x_i) + (1-Y_i) \log [1-f(x_i)] \right)$$

penalized method:

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \left[\hat{R}_n(f) + P_\lambda(f) \right]$$

e.g. (L1-penalized logistic regression)

$$-\frac{1}{n} \sum_{i=1}^n \left[Y_i (\beta_0 + \beta^\top X_i) - \log (1 + e^{\beta_0 + \beta^\top X_i}) \right] + \lambda \sum_{j=0}^d |\beta_j|$$