Chapter 6 : Functional Data Fusion Application to Supervised Learning

John Klein

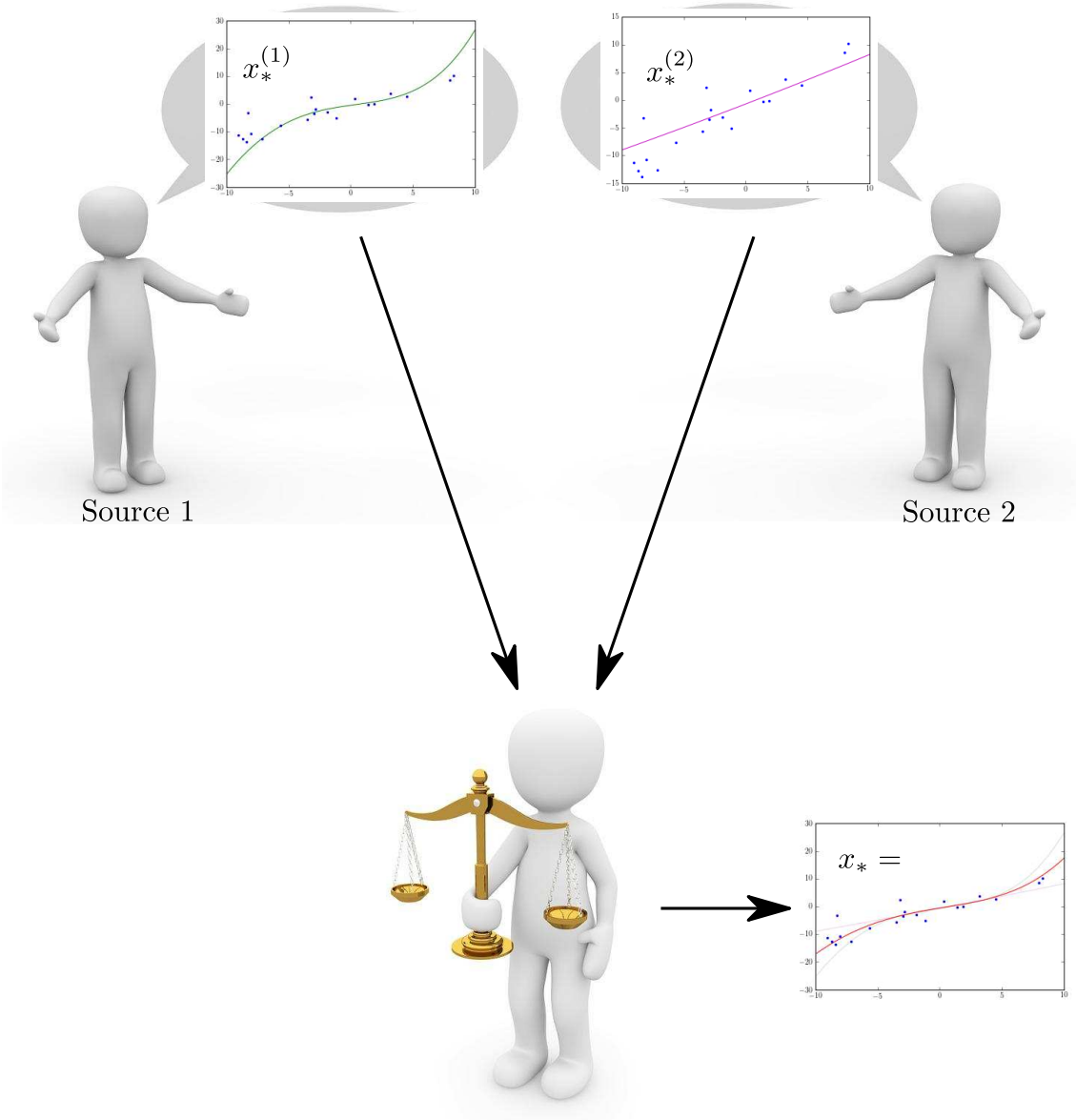
Lille1 Université - CRIStAL UMR CNRS 9189

Chapter organization

- In this chapter, each datum delivered by a source is a **function**.

- We focus on regression or classification functions in a supervised learning paradigm.

Probabilistic data fusion setting :



Functional data fusion :

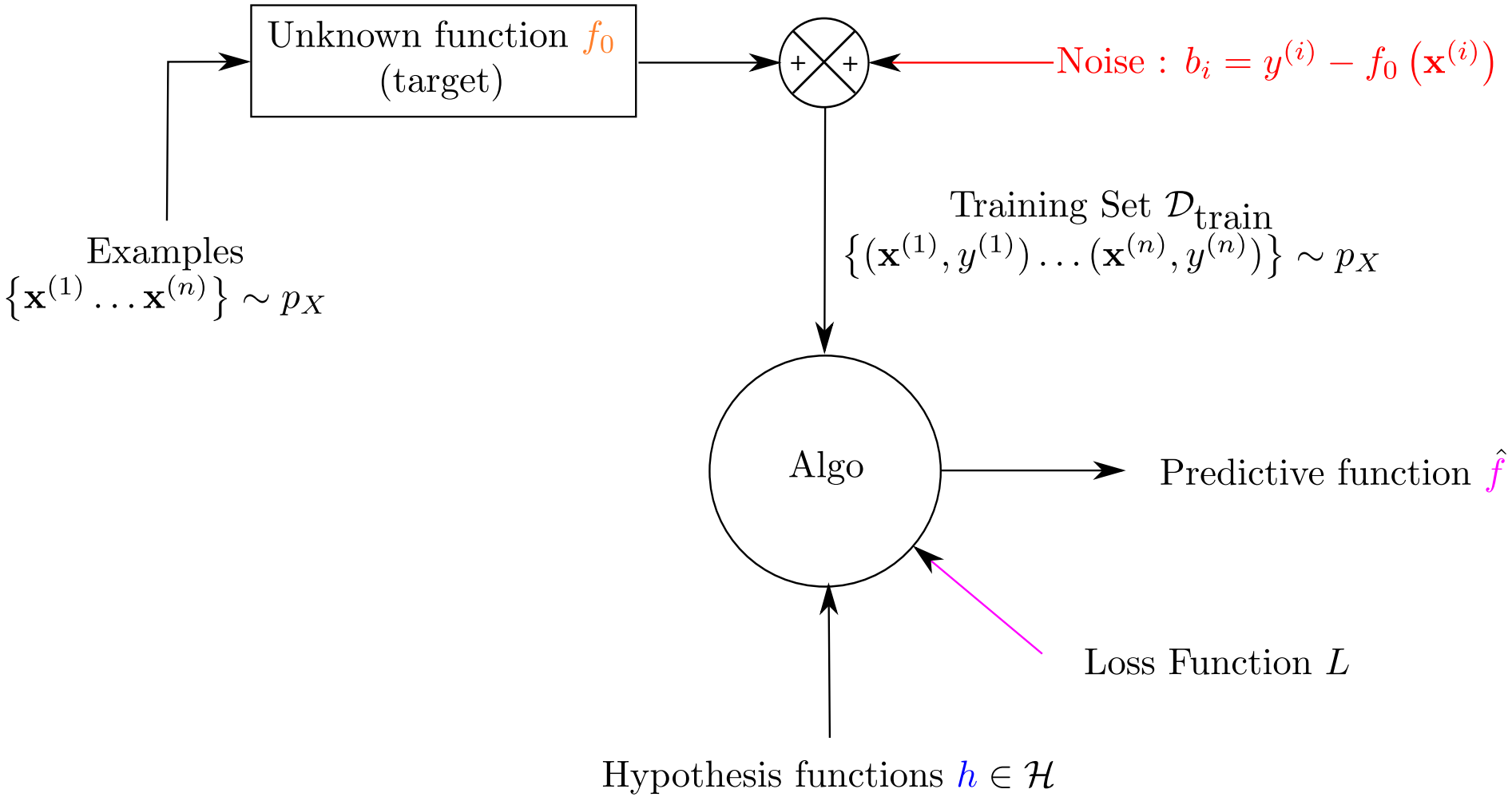
- Let $\mathcal{L}_2$ denote the space of square integrable functions.

- Let us give a formal definition of functional data fusion problems :

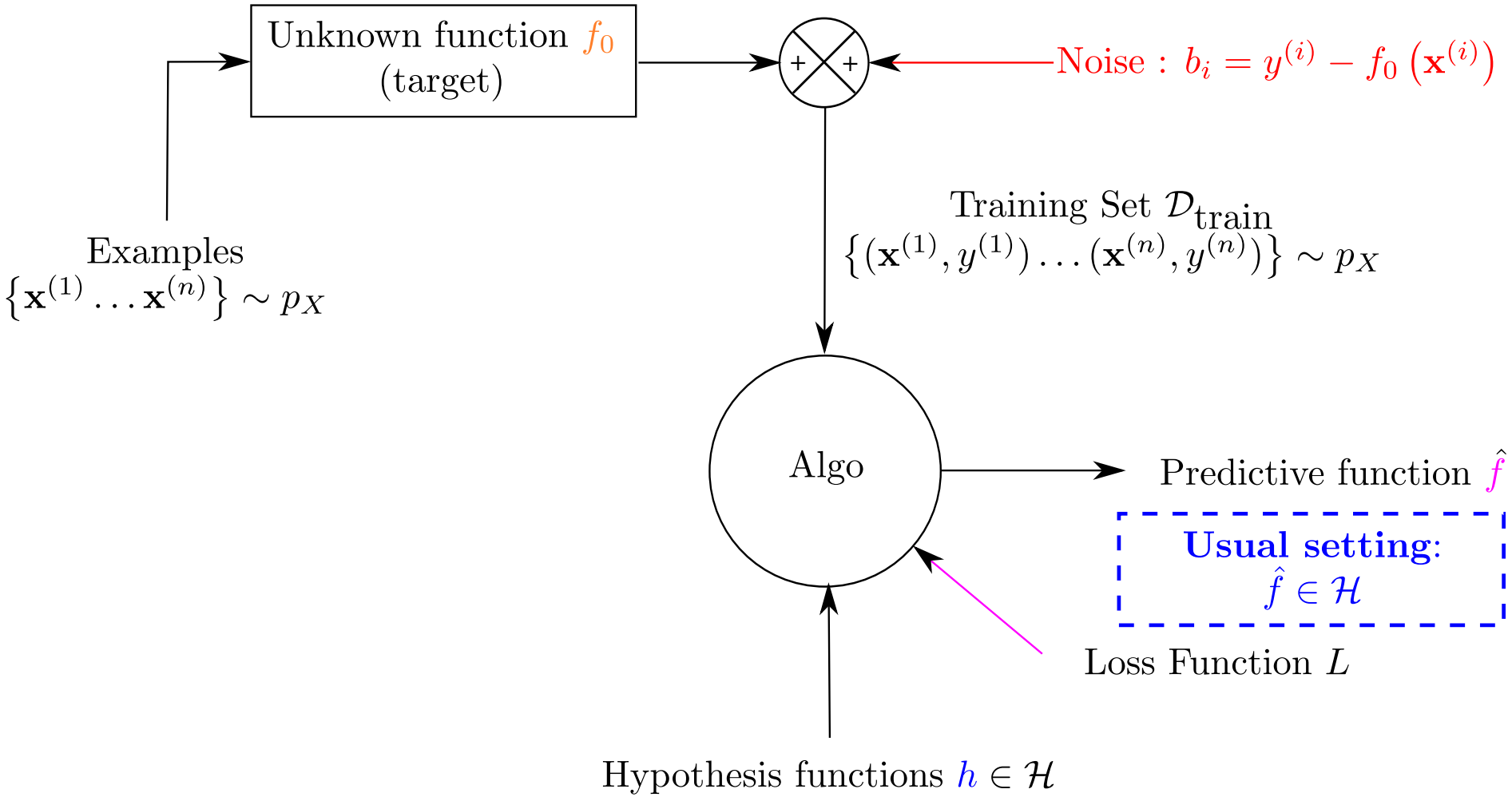
Definition

Functional data fusion is a subclass of data fusion where advocacies live in $\mathbb{X} = \mathcal{L}_2$.

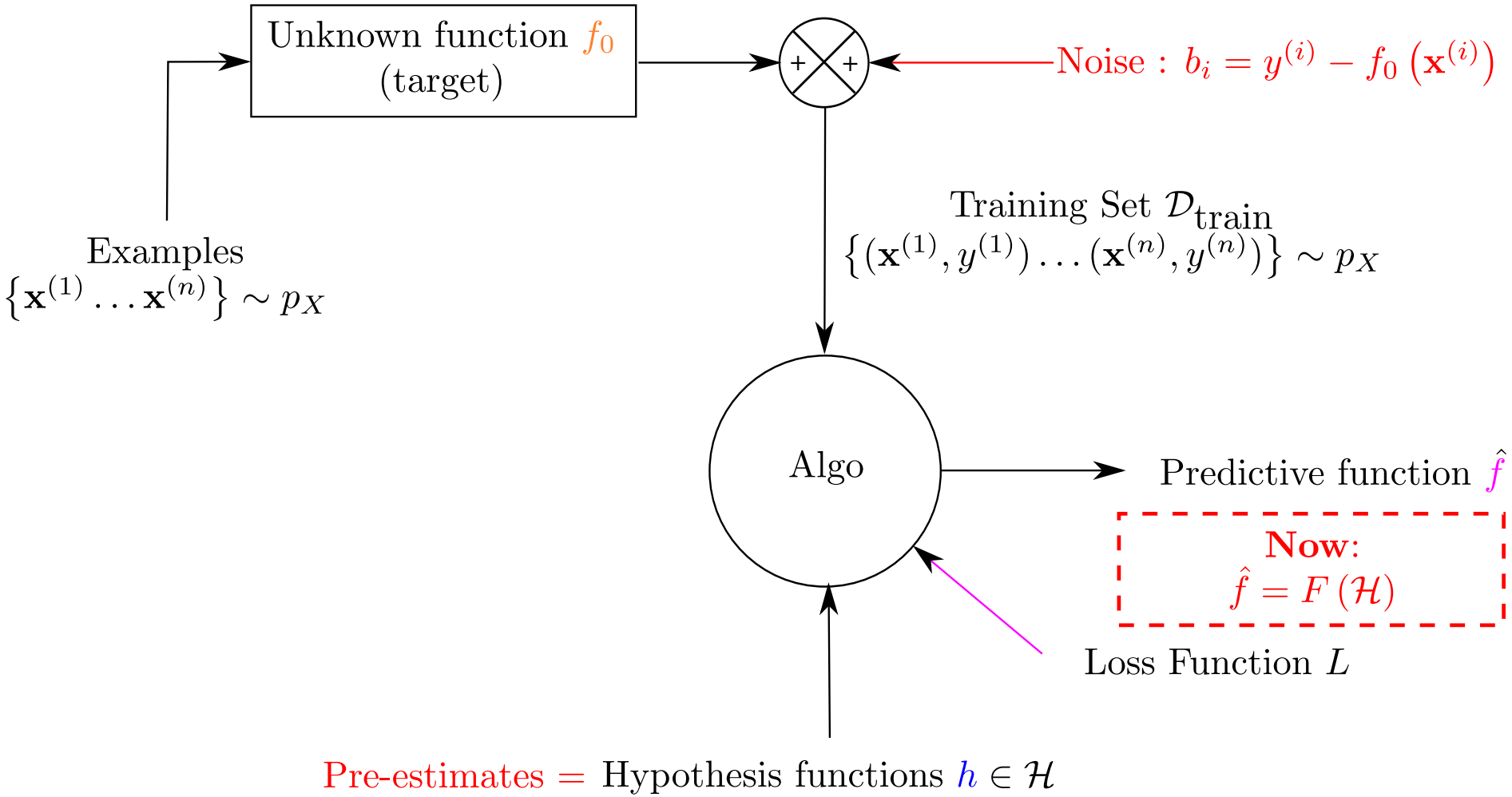
Supervised Learning : the learning diagram



Supervised Learning : the learning diagram



Supervised Learning : the learning diagram



New notations :

Before	Now
x (solution)	f_0 (solution)
x_* (aggregate)	$\hat{f}$ (aggregated estimate)
$x_*^{(i)}$ (advocacy)	$\hat{f}_i$ (pre-estimates)

Supervised Learning : important notions

- Train Error (empirical risk) :

$$Err_{\text{train}} = \sum_{i=1}^{n} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{train}}} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

- Test Error :

$$Err_{\text{test}} = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{test}}} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

- Generalization Error :

$$Err_{\text{gen}} = \mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right), .\right)\right].$$

Supervised Learning : important notions

- Train Error (empirical risk) :

$$Err_{\text{train}} = \sum_{i=1}^{n} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{train}}} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

- Test Error :

$$Err_{\text{test}} = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{test}}} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

- Generalization Error :

$$Err_{\text{gen}} = \mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right), .\right)\right].$$

Supervised Learning : important notions

- Train Error (empirical risk) :

$$Err_{\text{train}} = \sum_{i=1}^{n} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{train}}} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

- Test Error :

$$Err_{\text{test}} = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{test}}} L\left(\hat{f}\left(\mathbf{x}^{(i)}\right), y^{(i)}\right).$$

- Generalization Error :

$$Err_{\text{gen}} = \mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right), .\right)\right].$$

Supervised Learning : important notions

- Err_{train} is a biaised estimator of Err_{gen}.
- Err_{test} is a unbiaised estimator of Err_{gen}.
- If $\hat{f}$ is learnt from $\mathcal{D}_{\text{train}}$, then the genuine generalization error is :

$$Err_{\text{gen}} = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right),.\right)\right]\right].$$

- This definition of Err_{gen} is also known as the risk.
- It features the ability for the algorithm to generalize correctly in this situation while the previous definition features only the ability of the output predictor $\hat{f}$ to generalize.

Supervised Learning : important notions

- Err_{train} is a biaised estimator of Err_{gen}.
- Err_{test} is a unbiaised estimator of Err_{gen}.
- If $\hat{f}$ is learnt from $\mathcal{D}_{\text{train}}$, then the genuine generalization error is :

$$Err_{\text{gen}} = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right),.\right)\right]\right].$$

- This definition of Err_{gen} is also known as the risk.
- It features the ability for the algorithm to generalize correctly in this situation while the previous definition features only the ability of the output predictor $\hat{f}$ to generalize.

Supervised Learning : important notions

- Err_{train} is a biaised estimator of Err_{gen}.
- Err_{test} is a unbiaised estimator of Err_{gen}.
- If $\hat{f}$ is learnt from $\mathcal{D}_{\text{train}}$, then the genuine generalization error is :

$$Err_{\text{gen}} = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right),.\right)\right]\right].$$

- This definition of Err_{gen} is also known as the risk.
- It features the ability for the algorithm to generalize correctly in this situation while the previous definition features only the ability of the output predictor $\hat{f}$ to generalize.

Supervised Learning : important notions

- Err_{train} is a biaised estimator of Err_{gen}.
- Err_{test} is a unbiaised estimator of Err_{gen}.
- If $\hat{f}$ is learnt from $\mathcal{D}_{\text{train}}$, then the genuine generalization error is :

$$Err_{\text{gen}} = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right),.\right)\right]\right].$$

- This definition of Err_{gen} is also known as the risk.
- It features the ability for the algorithm to generalize correctly in this situation while the previous definition features only the ability of the output predictor $\hat{f}$ to generalize.

Supervised Learning : important notions

- Err_{train} is a biaised estimator of Err_{gen}.
- Err_{test} is a unbiaised estimator of Err_{gen}.
- If $\hat{f}$ is learnt from $\mathcal{D}_{\text{train}}$, then the genuine generalization error is :

$$Err_{\text{gen}} = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{(X,Y)}\left[L\left(\hat{f}\left(.\right),.\right)\right]\right].$$

- This definition of Err_{gen} is also known as the risk.
- It features the ability for the algorithm to generalize correctly in this situation while the previous definition features only the ability of the output predictor $\hat{f}$ to generalize.

Chapter organization

Mixture models : definition

- A mixture model is a convex combination of posterior (predictive) distributions $p_{Y|X;\theta_i}(y|\mathbf{x})$:

$$p_{Y|X;\Theta}(y|\mathbf{x}) = \sum_{i=1}^{N_s} \pi_i p_{Y|X;\theta_i}(y|\mathbf{x})$$

- The combined distribution depends on the sources parameters θ_i and the combination parameters π_i :

$$\Theta = \{\theta_1, \ldots, \theta_{N_s}, \pi_1, \ldots, \pi_{N_s}\}.$$

- We have $\sum_{i=1}^{N_s} \pi_i = 1$ and $\pi_i \geq 0, \forall i$.

Mixture models : definition

- A mixture model is a convex combination of posterior (predictive) distributions $p_{Y|X;\theta_i}(y|\mathbf{x})$:

$$p_{Y|X;\Theta}(y|\mathbf{x}) = \sum_{i=1}^{N_s} \pi_i p_{Y|X;\theta_i}(y|\mathbf{x})$$

- The combined distribution depends on the sources parameters θ_i and the combination parameters π_i :

$$\Theta = \{\theta_1, \ldots, \theta_{N_s}, \pi_1, \ldots, \pi_{N_s}\}.$$

- We have $\sum_{i=1}^{N_s} \pi_i = 1$ and $\pi_i \geq 0, \forall i$.

Mixture models : definition

- A mixture model is a convex combination of posterior (predictive) distributions $p_{Y|X;\theta_i}(y|\mathbf{x})$:

$$p_{Y|X;\Theta}(y|\mathbf{x}) = \sum_{i=1}^{N_s} \pi_i p_{Y|X;\theta_i}(y|\mathbf{x})$$

- The combined distribution depends on the sources parameters θ_i and the combination parameters π_i :

$$\Theta = \{\theta_1, \ldots, \theta_{N_s}, \pi_1, \ldots, \pi_{N_s}\}.$$

- We have $\sum_{i=1}^{N_s} \pi_i = 1$ and $\pi_i \geq 0, \forall i$.

Mixture models : definition

- For a classification task, each pre-estimate is given by

$$\hat{f}_i(\mathbf{x}) = \arg\max_y p_{Y|X;\boldsymbol{\theta}_i}(y|\mathbf{x}).$$

- Consequently, the aggregated estimate is obtained by a weighted vote.
- All parameters can be jointly learnt from $\mathcal{D}_{\text{train}}$ using EM.
- Non-probabilistic pre-estimates (SVM, k-NN) cannot be combined this way.

Mixture models : definition

- For a classification task, each pre-estimate is given by

$$\hat{f}_i(\mathbf{x}) = \arg\max_y p_{Y|X;\theta_i}(y|\mathbf{x}).$$

- Consequently, the aggregated estimate is obtained by a weighted vote.
- All parameters can be jointly learnt from $\mathcal{D}_{\text{train}}$ using EM.
- Non-probabilistic pre-estimates (SVM, k-NN) cannot be combined this way.

Mixture models : definition

- For a classification task, each pre-estimate is given by

$$\hat{f}_i(\mathbf{x}) = \arg\max_y p_{Y|X;\theta_i}(y|\mathbf{x}).$$

- Consequently, the aggregated estimate is obtained by a weighted vote.
- All parameters can be jointly learnt from $\mathcal{D}_{\text{train}}$ using EM.
- Non-probabilistic pre-estimates (SVM, k-NN) cannot be combined this way.

Mixture models : definition

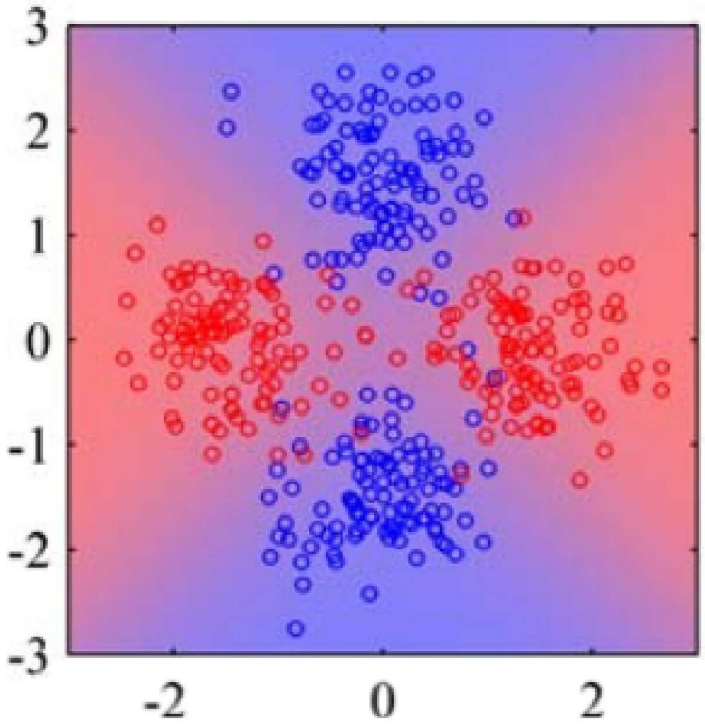
- For a classification task, each pre-estimate is given by

$$\hat{f}_i(\mathbf{x}) = \arg\max_y p_{Y|X;\theta_i}(y|\mathbf{x}).$$

- Consequently, the aggregated estimate is obtained by a weighted vote.
- All parameters can be jointly learnt from $\mathcal{D}_{\text{train}}$ using EM.
- Non-probabilistic pre-estimates (SVM, k-NN) cannot be combined this way.

Mixture models : Example - Mixture of LogRegs

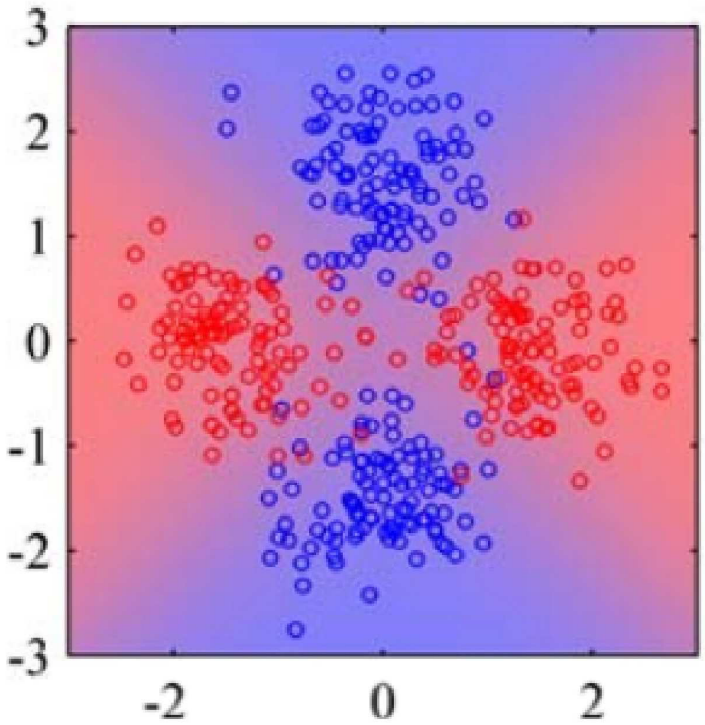
- Impossible to fit a linear model to this dataset :



- Let's try to fit a mixture of 2 logistic regressions.

Mixture models : Example - Mixture of LogRegs

- Impossible to fit a linear model to this dataset :



- Let's try to fit a mixture of 2 logistic regressions.

Mixture models : Example - Mixture of LogRegs

- Outputs y are binary variables $\{0;1\}$ and inputs $\mathbf{x}$ are vectors in $\mathbb{R}^d$.

- The posterior is :

$$p_{Y|X;\Theta}(y|\mathbf{x}) = \pi_1 (1-\hat{y}_1)^{1-y} \hat{y}_1^y + \pi_2 (1-\hat{y}_2)^{1-y} \hat{y}_2^y,$$
$$= \begin{cases} \pi_1 \hat{y}_1 + \pi_2 \hat{y}_2 & \text{if } y = 1 \\ \pi_1 (1-\hat{y}_1) + \pi_2 (1-\hat{y}_2) & \text{if } y = 0 \end{cases},$$

with $\hat{y}_1 = \mathrm{sgm}\left(\theta_1^T \cdot \mathbf{x}_+\right)$ and $\hat{y}_2 = \mathrm{sgm}\left(\theta_2^T \cdot \mathbf{x}_+\right)$ the logitic outputs and

$$\mathbf{x}_+^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_d^{(i)} \\ 1 \end{bmatrix}$$

Mixture models : Example - Mixture of LogRegs

- Outputs y are binary variables $\{0;1\}$ and inputs $\mathbf{x}$ are vectors in $\mathbb{R}^d$.

- The posterior is :

$$\begin{aligned} p_{Y|X;\Theta}(y|\mathbf{x}) &= \pi_1 (1-\hat{y}_1)^{1-y} \hat{y}_1^y + \pi_2 (1-\hat{y}_2)^{1-y} \hat{y}_2^y, \\ &= \begin{cases} \pi_1 \hat{y}_1 + \pi_2 \hat{y}_2 & \text{if } y = 1 \\ \pi_1 (1-\hat{y}_1) + \pi_2 (1-\hat{y}_2) & \text{if } y = 0 \end{cases}, \end{aligned}$$

with $\hat{y}_1 = \mathrm{sgm}\left(\boldsymbol{\theta}_1^T \cdot \mathbf{x}_+\right)$ and $\hat{y}_2 = \mathrm{sgm}\left(\boldsymbol{\theta}_2^T \cdot \mathbf{x}_+\right)$ the logitic outputs and

$$\mathbf{x}_+^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_d^{(i)} \\ 1 \end{bmatrix}$$

Mixture models : Example - Mixture of LogRegs

- The likelihood is

$$
\begin{aligned}
\mathcal{L}(\Theta) &= p(\text{data}|\Theta), \\
&= \prod_{i=1}^{n} p(\text{datum number } i|\Theta), \\
&= \prod_{i=1}^{n} p\left(y^{(i)}|\mathbf{x}^{(i)}, \Theta\right), \\
&= \prod_{i=1}^{n} \pi_1 \left(1-\hat{y}_1^{(i)}\right)^{1-y^{(i)}} \left(\hat{y}_1^{(i)}\right)^{y^{(i)}} + \pi_2 \left(1-\hat{y}_2^{(i)}\right)^{1-y^{(i)}} \left(\hat{y}_2^{(i)}\right)^{y^{(i)}}.
\end{aligned}
$$

Mixture models : Example - Mixture of LogRegs

- Let us introduce latent variables $z^{(i)}$ standing for the fact that example $\mathbf{x}^{(i)}$ was generated by mixture component n°1.

- The complete data[1] likelihood is then :

$$\begin{aligned}\mathcal{L}_{\text{comp}}(\Theta) &= \prod_{i=1}^{n} p\left(y^{(i)}, z^{(i)} | \mathbf{x}^{(i)}, \Theta\right), \\ &= \prod_{i=1}^{n} \prod_{k=1}^{2} \left(\pi_k \left(1-\hat{y}_k^{(i)}\right)^{1-y^{(i)}} \left(\hat{y}_k^{(i)}\right)^{y^{(i)}}\right)^{\mathbb{I}_{k-1}\left(1-z^{(i)}\right)}.\end{aligned}$$

- A fake multiplication appears because now each point is concerned with only one component of the mixture !

1. hidden and observed data

Mixture models : Example - Mixture of LogRegs

- Let us introduce latent variables $z^{(i)}$ standing for the fact that example $\mathbf{x}^{(i)}$ was generated by mixture component n°1.

- The complete data [1] likelihood is then :

$$\begin{aligned}\mathcal{L}_{\text{comp}}(\boldsymbol{\Theta}) &= \prod_{i=1}^{n} p\left(y^{(i)}, z^{(i)} | \mathbf{x}^{(i)}, \boldsymbol{\Theta}\right), \\ &= \prod_{i=1}^{n} \prod_{k=1}^{2} \left(\pi_k \left(1-\hat{y}_k^{(i)}\right)^{1-y^{(i)}} \left(\hat{y}_k^{(i)}\right)^{y^{(i)}}\right)^{\mathbb{I}_{k-1}\left(1-z^{(i)}\right)}.\end{aligned}$$

- A fake multiplication appears because now each point is concerned with only one component of the mixture !

1. hidden and observed data

Mixture models : Example - Mixture of LogRegs

- Let us introduce latent variables $z^{(i)}$ standing for the fact that example $\mathbf{x}^{(i)}$ was generated by mixture component n°1.

- The complete data [1] likelihood is then :

$$\begin{aligned}\mathcal{L}_{\text{comp}}(\boldsymbol{\Theta}) &= \prod_{i=1}^{n} p\left(y^{(i)}, z^{(i)} | \mathbf{x}^{(i)}, \boldsymbol{\Theta}\right), \\ &= \prod_{i=1}^{n} \prod_{k=1}^{2} \left(\pi_k \left(1-\hat{y}_k^{(i)}\right)^{1-y^{(i)}} \left(\hat{y}_k^{(i)}\right)^{y^{(i)}}\right)^{\mathbb{I}_{k-1}\left(1-z^{(i)}\right)}.\end{aligned}$$

- A fake multiplication appears because now each point is concerned with only one component of the mixture !

1. hidden and observed data

Mixture models : Example - Mixture of LogRegs

- E step : one can show that

$$\mathbb{E}_{z^{(1)},..,z^{(n)}}\left[\log \mathcal{L}_{\text{comp}}\left(\Theta\right)\right] = \sum_{i=1}^{n}\sum_{k=1}^{2}\gamma_k^{(i)}\left[\log\left(\pi_k\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}_k^{(i)}\right) + y^{(i)}\log\left(\hat{y}_k^{(i)}\right)\right],$$

$$\gamma_k^{(i)} = \frac{\pi_k\left(1-\hat{y}_k^{(i)}\right)^{1-y^{(i)}}\left(\hat{y}_k^{(i)}\right)^{y^{(i)}}}{\sum_{k'}\pi_{k'}\left(1-\hat{y}_{k'}^{(i)}\right)^{1-y^{(i)}}\left(\hat{y}_{k'}^{(i)}\right)^{y^{(i)}}}.$$

- M step : parameters θ_j need to be estimated using a gradient ascent (Newton's method) while mixing weights are given by :

$$\pi_k = \frac{1}{n}\sum_{i=1}^{n}\gamma_k^{(i)}.$$

Mixture models : Example - Mixture of LogRegs

- E step : one can show that

$$\mathbb{E}_{z^{(1)},..,z^{(n)}}\left[\log \mathcal{L}_{\text{comp}}\left(\boldsymbol{\Theta}\right)\right] = \sum_{i=1}^{n}\sum_{k=1}^{2}\gamma_k^{(i)}\left[\log\left(\pi_k\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}_k^{(i)}\right) + y^{(i)}\log\left(\hat{y}_k^{(i)}\right)\right],$$

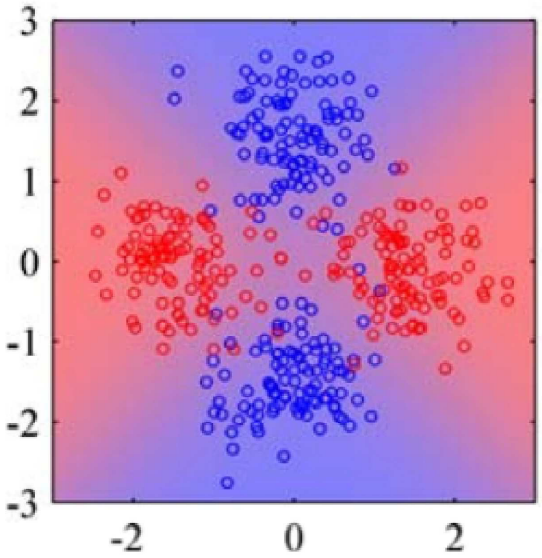
$$\gamma_k^{(i)} = \frac{\pi_k\left(1-\hat{y}_k^{(i)}\right)^{1-y^{(i)}}\left(\hat{y}_k^{(i)}\right)^{y^{(i)}}}{\sum_{k'}\pi_{k'}\left(1-\hat{y}_{k'}^{(i)}\right)^{1-y^{(i)}}\left(\hat{y}_{k'}^{(i)}\right)^{y^{(i)}}}.$$

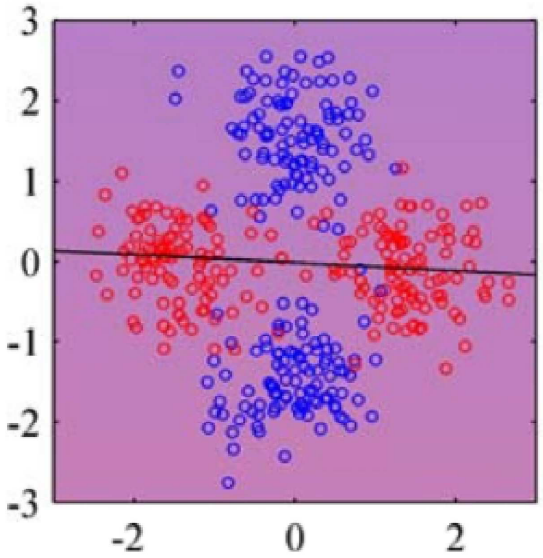
- M step : parameters $\boldsymbol{\theta}_i$ need to be estimated using a gradient ascent (Newton's method) while mixing weights are given by :

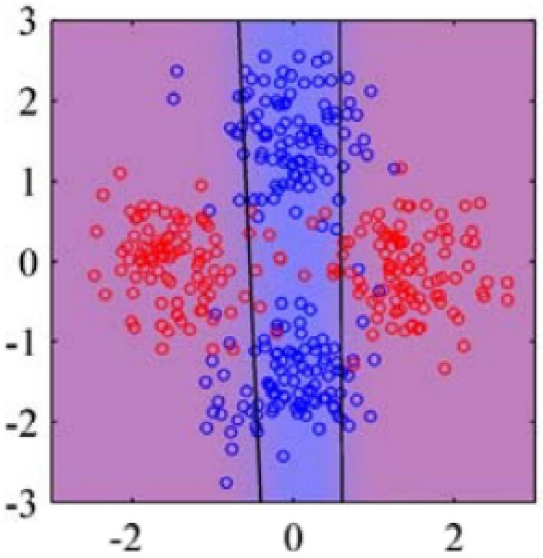
$$\pi_k = \frac{1}{n}\sum_{i=1}^{n}\gamma_k^{(i)}.$$

Mixture models : Example - Mixture of LogRegs

- The fit result is



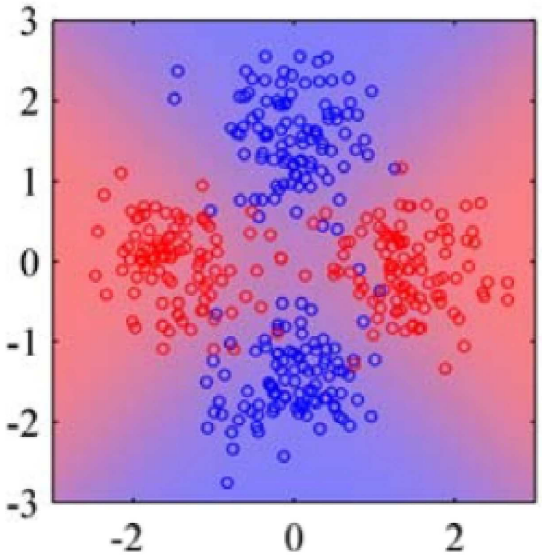


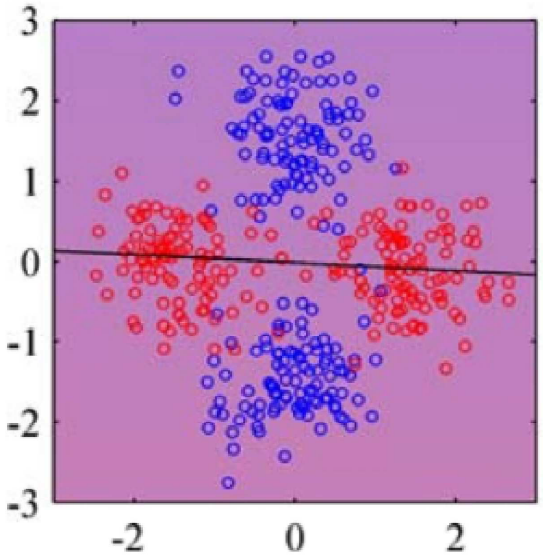


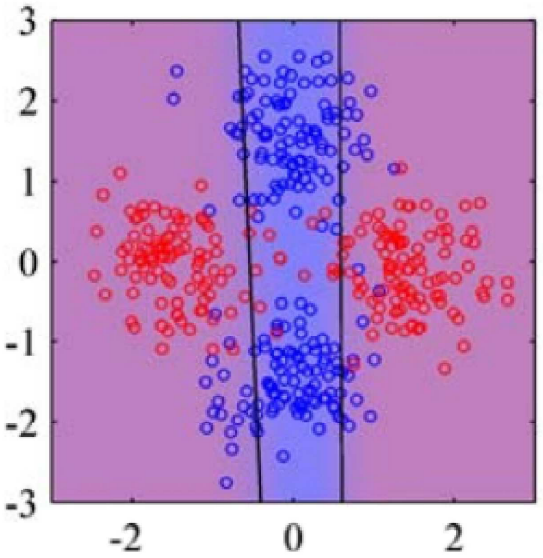
- See [Bishop 14.5.2] for more details.

Mixture models : Example - Mixture of LogRegs

- The fit result is







- See [Bishop 14.5.2] for more details.

Mixture of Experts : definition

- The fact that pre-estimates are functions of $\mathbf{x}$ is not exploited in the previous model.

- Mixtures of experts generalize this model by making mixing weights input-dependent :

$$\pi_k(\mathbf{x}) = \operatorname{smax}\left(\mathbf{V}^T \cdot \mathbf{x}\right).$$

- In this context, mixing weights are called gating functions and each pre-estimate is called an expert.

Mixture of Experts : definition

- The fact that pre-estimates are functions of $\mathbf{x}$ is not exploited in the previous model.

- Mixtures of experts generalize this model by making mixing weights input-dependent :

$$\pi_k(\mathbf{x}) = \operatorname{smax}\left(\mathbf{V}^T \cdot \mathbf{x}\right).$$

- In this context, mixing weights are called gating functions and each pre-estimate is called an expert.

Mixture of Experts : definition

- The fact that pre-estimates are functions of $\mathbf{x}$ is not exploited in the previous model.

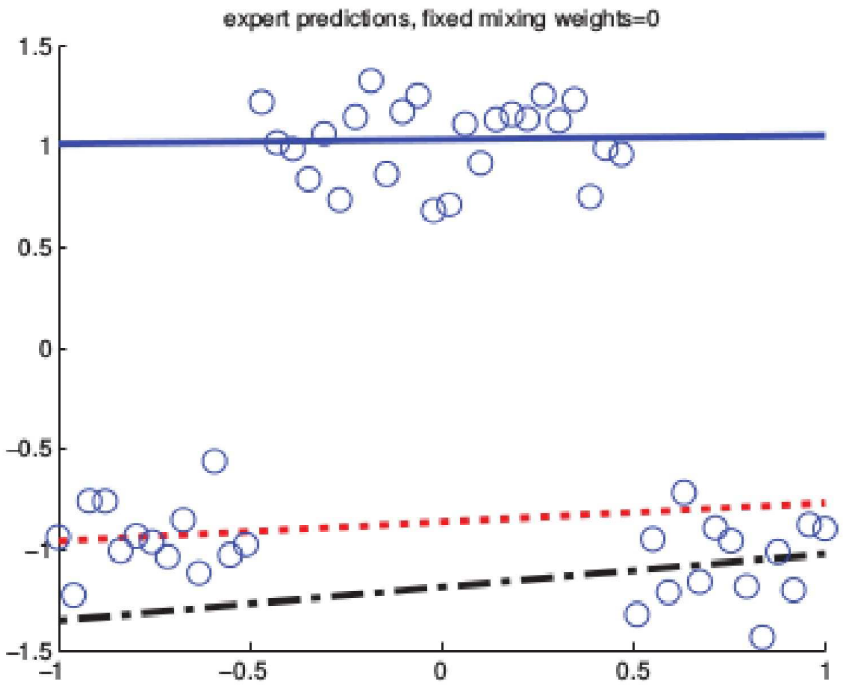
- Mixtures of experts generalize this model by making mixing weights input-dependent :

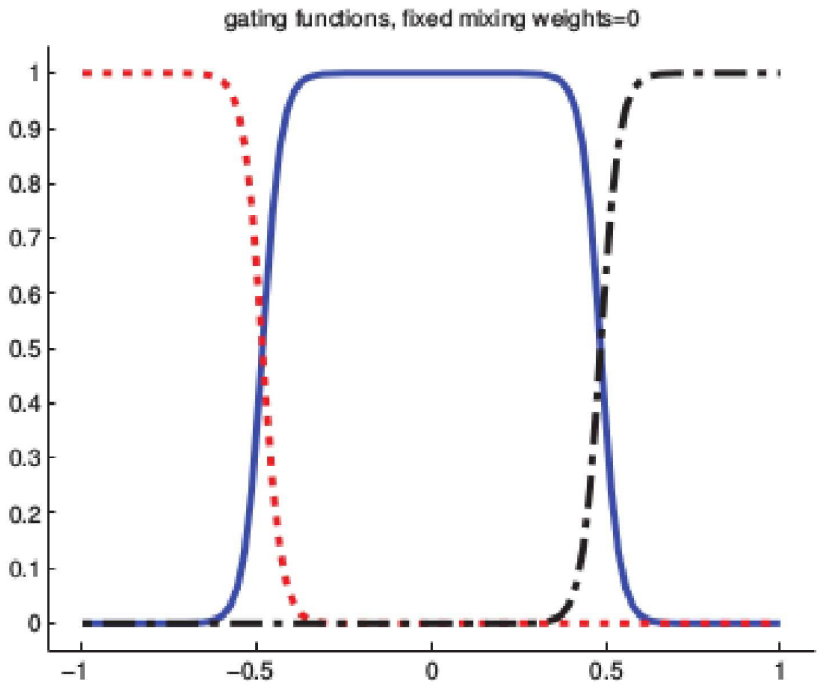
$$\pi_k(\mathbf{x}) = \operatorname{smax}\left(\mathbf{V}^T \cdot \mathbf{x}\right).$$

- In this context, mixing weights are called gating functions and each pre-estimate is called an expert.

Mixture of Experts : Example - Mixture of Linear Regressors

- Obviously, a linear regression is a good model for subsets of the following data :

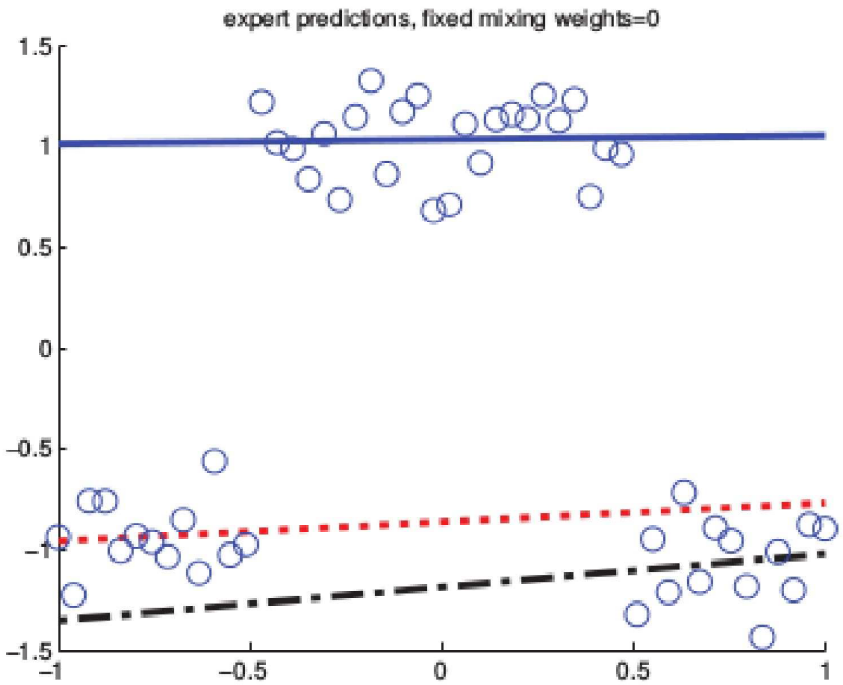


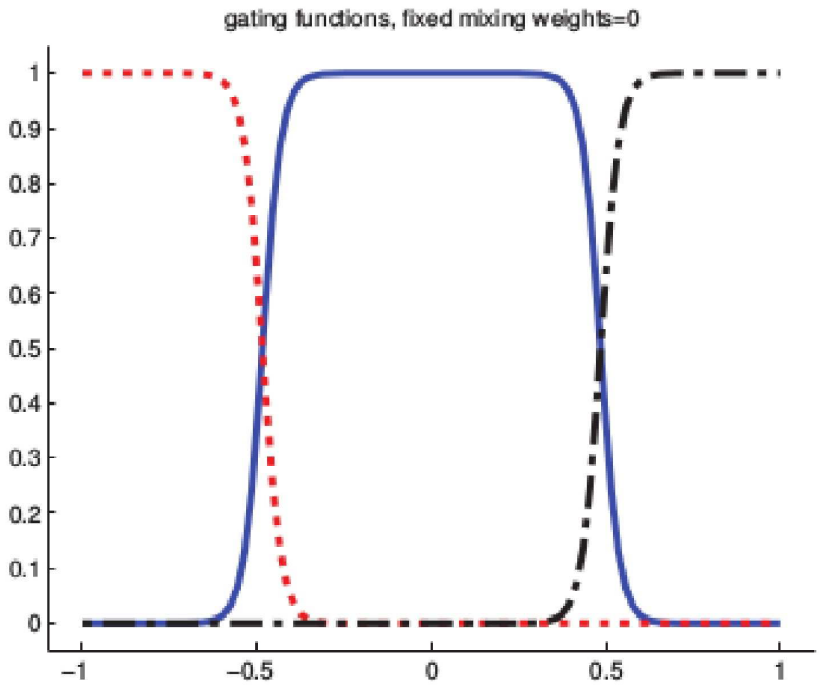


- The right figure shows trained gating functions for each regressor.

Mixture of Experts : Example - Mixture of Linear Regressors

- Obviously, a linear regression is a good model for subsets of the following data :





- The right figure shows trained gating functions for each regressor.

Mixture of Experts : Example - Mixture of Linear Regressors

- E step : one can show that

$$\mathbb{E}_{z^{(1)},..,z^{(n)}}\left[\log \mathcal{L}_{\text{comp}}\left(\Theta\right)\right] = \sum_{i=1}^{n}\sum_{k=1}^{2}\gamma_k^{(i)}\left[\log\left(\pi_k^{(i)}\right) - \frac{\left(y^{(i)} - \theta_k^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma_k^2}\right],$$

$$\gamma_k^{(i)} = \frac{\pi_k^{(i)} \times \frac{1}{\sqrt{2\pi}\sigma_k} e^{\frac{\left(y^{(i)} - \theta_k^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma_k^2}}}{\sum_{k'} \pi_{k'}^{(i)} \times \frac{1}{\sqrt{2\pi}\sigma_{k'}} e^{\frac{\left(y^{(i)} - \theta_{k'}^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma_{k'}^2}}},$$

$$\pi_k^{(i)} = \text{smax}\left(\mathbf{V}^T \cdot \mathbf{x}^{(i)}\right).$$

- M step : there is a closed-form MLE solution for parameters θ_k and σ_k. $\mathbf{V}$ is estimated by gradient ascent (Newton's method).

Mixture of Experts : Example - Mixture of Linear Regressors

- E step : one can show that

$$\mathbb{E}_{z^{(1)},..,z^{(n)}}\left[\log \mathcal{L}_{\text{comp}}\left(\Theta\right)\right] = \sum_{i=1}^{n}\sum_{k=1}^{2}\gamma_k^{(i)}\left[\log\left(\pi_k^{(i)}\right) - \frac{\left(y^{(i)} - \theta_k^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma_k^2}\right],$$

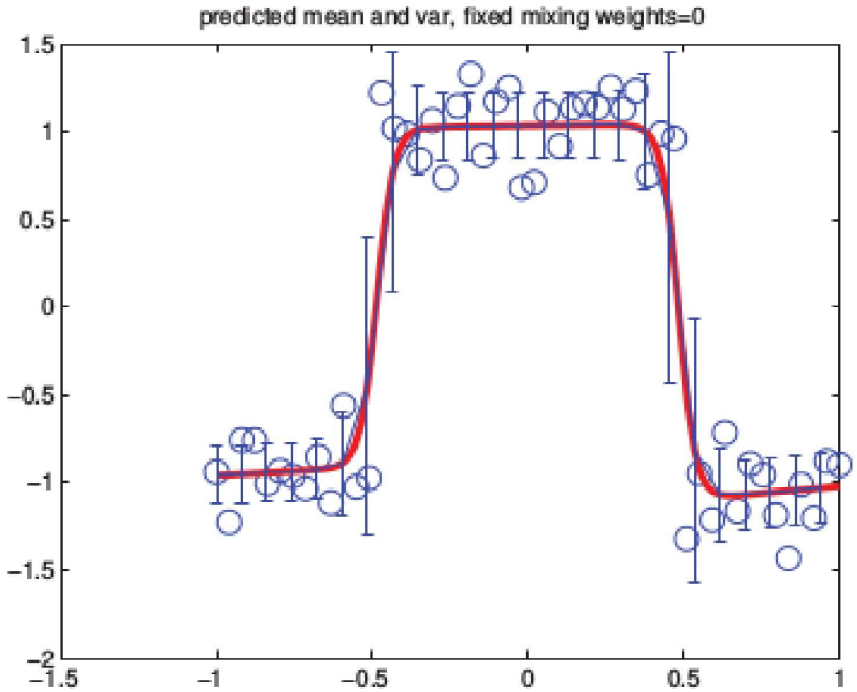
$$\gamma_k^{(i)} = \frac{\pi_k^{(i)} \times \frac{1}{\sqrt{2\pi}\sigma_k} e^{\frac{\left(y^{(i)} - \theta_k^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma_k^2}}}{\sum_{k'} \pi_{k'}^{(i)} \times \frac{1}{\sqrt{2\pi}\sigma_{k'}} e^{\frac{\left(y^{(i)} - \theta_{k'}^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma_{k'}^2}}},$$

$$\pi_k^{(i)} = \text{smax}\left(\mathbf{V}^T \cdot \mathbf{x}^{(i)}\right).$$

- M step : there is a closed-form MLE solution for parameters θ_k and σ_k. $\mathbf{V}$ is estimated by gradient ascent (Newton's method).

Mixture of Experts : Example - Mixture of Linear Regressors

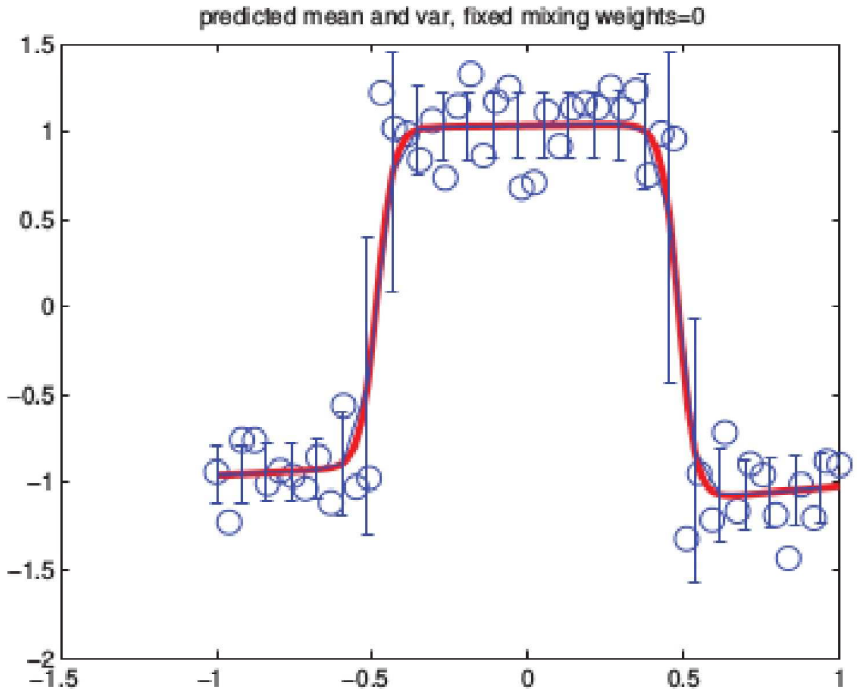
- The fit result is



- See [Murphy 2012 - 11.4.3] for more details.

Mixture of Experts : Example - Mixture of Linear Regressors

- The fit result is



- See [Murphy 2012 - 11.4.3] for more details.

Chapter organization

Ensemble methods : definition

- Ensemble methods are homogeneous classifier combination methods.

- This means that pre-estimates are trained using the same algorithm under different circumstances : different hyperparameters, different datasets, etc.

Ensemble methods : definition

- Ensemble methods are homogeneous classifier combination methods.

- This means that pre-estimates are trained using the same algorithm under different circumstances : different hyperparameters, different datasets, etc.

Ensemble methods : bagging

1. Generate a dataset $\mathcal{D}_{\text{boot}}$ from $\mathcal{D}_{\text{train}}$ (unif. samping with replacement).
2. Train your algorithm on $\mathcal{D}_{\text{boot}}$.
3. After m such trainings, combine using majority voting (classifiers) or average (regressors).

Ensemble methods : bagging

1. Generate a dataset $\mathcal{D}_{\text{boot}}$ from $\mathcal{D}_{\text{train}}$ (unif. samping with replacement).
2. Train your algorithm on $\mathcal{D}_{\text{boot}}$.
3. After m such trainings, combine using majority voting (classifiers) or average (regressors).

Ensemble methods : bagging

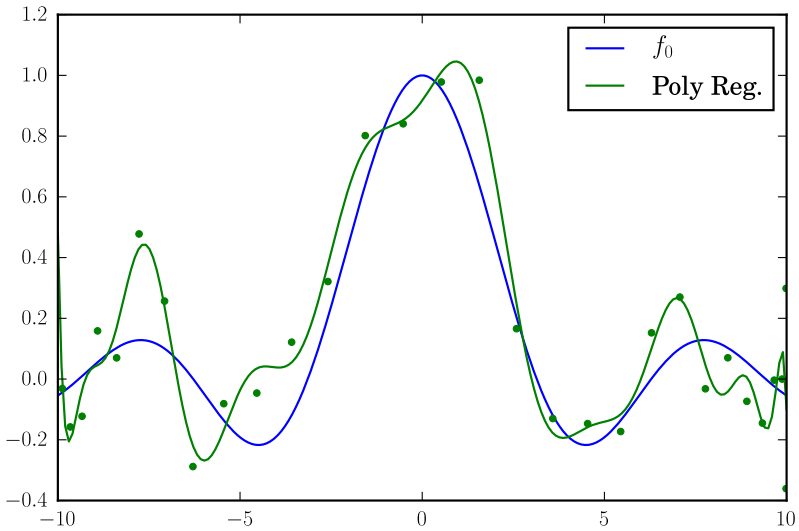
1. Generate a dataset $\mathcal{D}_{\text{boot}}$ from $\mathcal{D}_{\text{train}}$ (unif. samping with replacement).

2. Train your algorithm on $\mathcal{D}_{\text{boot}}$.

3. After m such trainings, combine using majority voting (classifiers) or average (regressors).

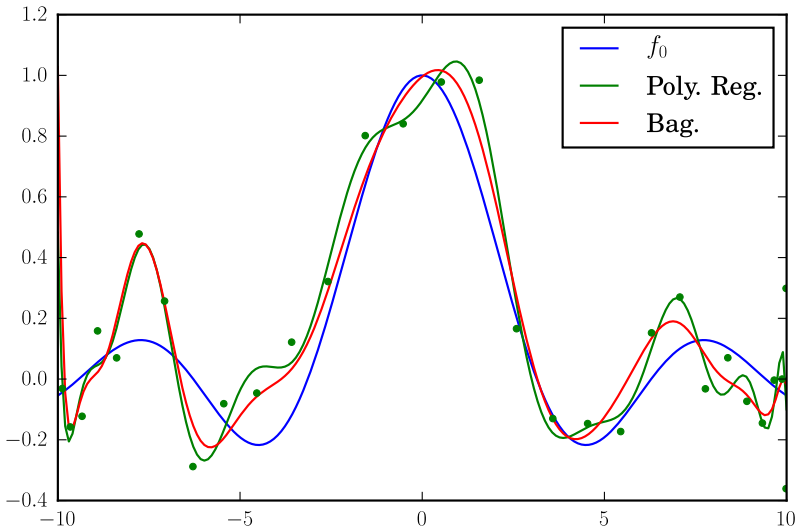
Ensemble methods : bagging - Regression example

- Suppose one fits a 20 degree polynomial on this noisy data :



Ensemble methods : bagging - Regression example

- Bagging will help a bit to mitigate overfitting



Ensemble methods : bagging - Why should it work ?

- Let ϵ_k denote the (signed) error function between the true function f_0 and one of my pre-estimate f_k that was trained on a bootstap sample :

$$\epsilon_k(\mathbf{x}) = f_k(\mathbf{x}) - f_0(\mathbf{x}).$$

- For regression problems, the usual loss function is the **quadratic** one.

- The generalization error for the predictor $f_k(\mathbf{x})$ is thus :

$$Err_{\text{gen}}^{(k)} = \mathbb{E}_X\left[(f_k(\mathbf{x}) - f_0(\mathbf{x}))^2\right] = \mathbb{E}_X\left[\epsilon_k(\mathbf{x})^2\right]$$

Ensemble methods : bagging - Why should it work ?

- Let ϵ_k denote the (signed) error function between the true function f_0 and one of my pre-estimate f_k that was trained on a bootstap sample :

$$\epsilon_k(\mathbf{x}) = f_k(\mathbf{x}) - f_0(\mathbf{x}).$$

- For regression problems, the usual loss function is the **quadratic** one.

- The generalization error for the predictor $f_k(\mathbf{x})$ is thus :

$$Err_{\text{gen}}^{(k)} = \mathbb{E}_X\left[(f_k(\mathbf{x}) - f_0(\mathbf{x}))^2\right] = \mathbb{E}_X\left[\epsilon_k(\mathbf{x})^2\right]$$

Ensemble methods : bagging - Why should it work ?

- Let ϵ_k denote the (signed) error function between the true function f_0 and one of my pre-estimate f_k that was trained on a bootstap sample :

$$\epsilon_k(\mathbf{x}) = f_k(\mathbf{x}) - f_0(\mathbf{x}).$$

- For regression problems, the usual loss function is the **quadratic** one.

- The generalization error for the predictor $f_k(\mathbf{x})$ is thus :

$$Err_{\text{gen}}^{(k)} = \mathbb{E}_X\left[(f_k(\mathbf{x}) - f_0(\mathbf{x}))^2\right] = \mathbb{E}_X\left[\epsilon_k(\mathbf{x})^2\right]$$

Ensemble methods : bagging - Why should it work ?

- Now, the generalization error for the bagging ensemble writes :

$$Err_{\text{gen}}^{\text{ens}} = \mathbb{E}_X \left[\left(\frac{1}{m} \sum_{k=1}^{m} f_k(\mathbf{x}) - f_0(\mathbf{x}) \right)^2 \right] = \mathbb{E}_X \left[\left(\frac{1}{m} \sum_{k=1}^{m} \epsilon_k(\mathbf{x}) \right)^2 \right]$$

- If $\mathbb{E}_X[\epsilon_k] = 0 \ (\forall k)$ and $\mathbb{E}_X[\epsilon_k \epsilon_{k'}] = 0 \ (\forall k \neq k')$, then

$$Err_{\text{gen}}^{\text{ens}} = \frac{1}{m} \times Err_{\text{gen}}^{\text{ave}} \quad \text{with} \quad Err_{\text{gen}}^{\text{ave}} = \frac{1}{m} \sum_{k=1}^{m} \mathbb{E}_X[\epsilon_k^2] = \frac{1}{m} \sum_{k=1}^{m} Err_{\text{gen}}^{(k)}$$

- $\rightarrow$ reduced error !

Ensemble methods : bagging - Why should it work ?

- Now, the generalization error for the bagging ensemble writes :

$$Err_{\text{gen}}^{\text{ens}} = \mathbb{E}_X \left[\left(\frac{1}{m} \sum_{k=1}^{m} f_k(\mathbf{x}) - f_0(\mathbf{x}) \right)^2 \right] = \mathbb{E}_X \left[\left(\frac{1}{m} \sum_{k=1}^{m} \epsilon_k(\mathbf{x}) \right)^2 \right]$$

- If $\mathbb{E}_X[\epsilon_k] = 0 \ (\forall k)$ and $\mathbb{E}_X[\epsilon_k \epsilon_{k'}] = 0 \ (\forall k \neq k')$, then

$$Err_{\text{gen}}^{\text{ens}} = \frac{1}{m} \times Err_{\text{gen}}^{\text{ave}} \quad \text{with} \quad Err_{\text{gen}}^{\text{ave}} = \frac{1}{m} \sum_{k=1}^{m} \mathbb{E}_X[\epsilon_k^2] = \frac{1}{m} \sum_{k=1}^{m} Err_{\text{gen}}^{(k)}$$

- $\rightarrow$ reduced error !

Ensemble methods : bagging - Why should it work ?

- Now, the generalization error for the bagging ensemble writes :

$$Err_{\text{gen}}^{\text{ens}} = \mathbb{E}_X \left[\left(\frac{1}{m} \sum_{k=1}^{m} f_k(\mathbf{x}) - f_0(\mathbf{x}) \right)^2 \right] = \mathbb{E}_X \left[\left(\frac{1}{m} \sum_{k=1}^{m} \epsilon_k(\mathbf{x}) \right)^2 \right]$$

- If $\mathbb{E}_X[\epsilon_k] = 0 \ (\forall k)$ and $\mathbb{E}_X[\epsilon_k \epsilon_{k'}] = 0 \ (\forall k \neq k')$, then

$$Err_{\text{gen}}^{\text{ens}} = \frac{1}{m} \times Err_{\text{gen}}^{\text{ave}} \quad \text{with} \quad Err_{\text{gen}}^{\text{ave}} = \frac{1}{m} \sum_{k=1}^{m} \mathbb{E}_X[\epsilon_k^2] = \frac{1}{m} \sum_{k=1}^{m} Err_{\text{gen}}^{(k)}$$

- $\rightarrow$ reduced error !

Ensemble methods : boosting

- Use bagging in case of overfitting ($\mathcal{H}$ is big $\rightarrow$ high variance in trained estimates).

- Use boosting in case of underfitting ($\mathcal{H}$ is small $\rightarrow$ low variance in trained estimates but high bias).

Ensemble methods : boosting

- Use bagging in case of overfitting ($\mathcal{H}$ is big $\rightarrow$ high variance in trained estimates).

- Use boosting in case of underfitting ($\mathcal{H}$ is small $\rightarrow$ low variance in trained estimates but high bias).

Ensemble methods : boosting - procedure

Sequentially train weak classifiers as :

1. Train classifier f_k on the weighted training set $\mathcal{D}_{\text{train}} \times \left\{ w_k^{(1)}, \ldots, w_k^{(n)} \right\}$.
2. Evaluate the classification uncertainty on each data point and update weights accordingly.
3. Update classifier mixing coefficients α_k

Finally, return (in the binary classification case) :

$$\hat{f} = \text{sign}\left(\sum_{k=1}^{m} \alpha_k f_k \right).$$

Ensemble methods : boosting - procedure

Sequentially train weak classifiers as :

1. Train classifier f_k on the weighted training set $\mathcal{D}_{\text{train}} \times \left\{ w_k^{(1)}, \ldots, w_k^{(n)} \right\}$.

2. Evaluate the classification uncertainty on each data point and update weights accordingly.

3. Update classifier mixing coefficients α_k

Finally, return (in the binary classification case) :

$$\hat{f} = \text{sign}\left(\sum_{k=1}^{m} \alpha_k f_k \right).$$

Ensemble methods : boosting - procedure

Sequentially train weak classifiers as :

1. Train classifier f_k on the weighted training set $\mathcal{D}_{\text{train}} \times \left\{ w_k^{(1)}, \ldots, w_k^{(n)} \right\}$.

2. Evaluate the classification uncertainty on each data point and update weights accordingly.

3. Update classifier mixing coefficients α_k

Finally, return (in the binary classification case) :

$$\hat{f} = \text{sign}\left(\sum_{k=1}^{m} \alpha_k f_k \right).$$

Ensemble methods : boosting - procedure

Sequentially train weak classifiers as :

1. Train classifier f_k on the weighted training set $\mathcal{D}_{\text{train}} \times \left\{ w_k^{(1)}, \ldots, w_k^{(n)} \right\}$.

2. Evaluate the classification uncertainty on each data point and update weights accordingly.

3. Update classifier mixing coefficients α_k

Finally, return (in the binary classification case) :

$$\hat{f} = \text{sign}\left(\sum_{k=1}^{m} \alpha_k f_k \right).$$

Ensemble methods : boosting - Why should it work ?

- In classification, the standard loss function is the **0-1 loss**.
- Boosting aims at minimizing a weighted **0-1 loss** over the training set :

$$J_k = \sum_{i=1}^{n} w_k^{(i)} \left(1 - \mathbb{I}_{y^{(i)}} \left(f_k \left(\mathbf{x}^{(i)}\right)\right)\right).$$

- The boosted ensemble, however, minimizes the exponential loss :

$$J_{\text{ens}} = \sum_{i=1}^{n} \exp\left(-y^{(i)} \frac{1}{2} \sum_{k=1}^{m} \alpha_k f_k \left(\mathbf{x}^{(i)}\right)\right).$$

- The first loss is derived from the ensemble one when the minimization focusses on f_k only.

Ensemble methods : boosting - Why should it work ?

- In classification, the standard loss function is the **0-1 loss**.
- Boosting aims at minimizing a weighted **0-1 loss** over the training set :

$$J_k = \sum_{i=1}^{n} w_k^{(i)} \left(1 - \mathbb{I}_{y^{(i)}} \left(f_k \left(\mathbf{x}^{(i)}\right)\right)\right).$$

- The boosted ensemble, however, minimizes the exponential loss :

$$J_{\text{ens}} = \sum_{i=1}^{n} \exp\left(-y^{(i)} \frac{1}{2} \sum_{k=1}^{m} \alpha_k f_k \left(\mathbf{x}^{(i)}\right)\right).$$

- The first loss is derived from the ensemble one when the minimization focusses on f_k only.

Ensemble methods : boosting - Why should it work ?

- In classification, the standard loss function is the **0-1 loss**.

- Boosting aims at minimizing a weighted **0-1 loss** over the training set :

$$J_k = \sum_{i=1}^{n} w_k^{(i)} \left(1 - \mathbb{I}_{y^{(i)}} \left(f_k \left(\mathbf{x}^{(i)}\right)\right)\right).$$

- The boosted ensemble, however, minimizes the exponential loss :

$$J_{\text{ens}} = \sum_{i=1}^{n} \exp\left(-y^{(i)} \frac{1}{2} \sum_{k=1}^{m} \alpha_k f_k \left(\mathbf{x}^{(i)}\right)\right).$$

- The first loss is derived from the ensemble one when the minimization focusses on f_k only.

Ensemble methods : boosting - Why should it work ?

- In classification, the standard loss function is the **0-1 loss**.

- Boosting aims at minimizing a weighted **0-1 loss** over the training set :

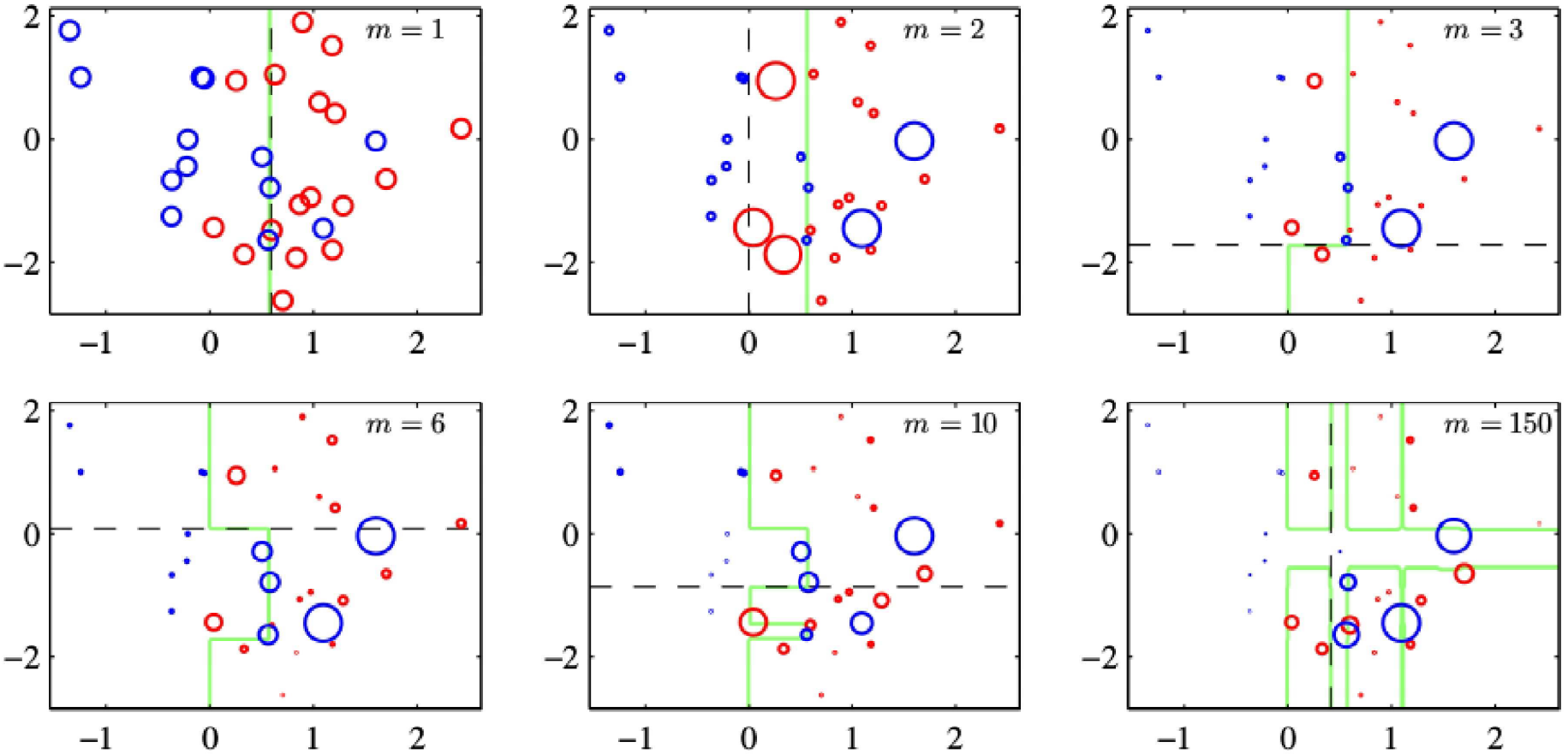
$$J_k = \sum_{i=1}^{n} w_k^{(i)} \left(1 - \mathbb{I}_{y^{(i)}} \left(f_k \left(\mathbf{x}^{(i)}\right)\right)\right).$$

- The boosted ensemble, however, minimizes the exponential loss :

$$J_{\text{ens}} = \sum_{i=1}^{n} \exp\left(-y^{(i)} \frac{1}{2} \sum_{k=1}^{m} \alpha_k f_k \left(\mathbf{x}^{(i)}\right)\right).$$

- The first loss is derived from the ensemble one when the minimization focusses on f_k only.

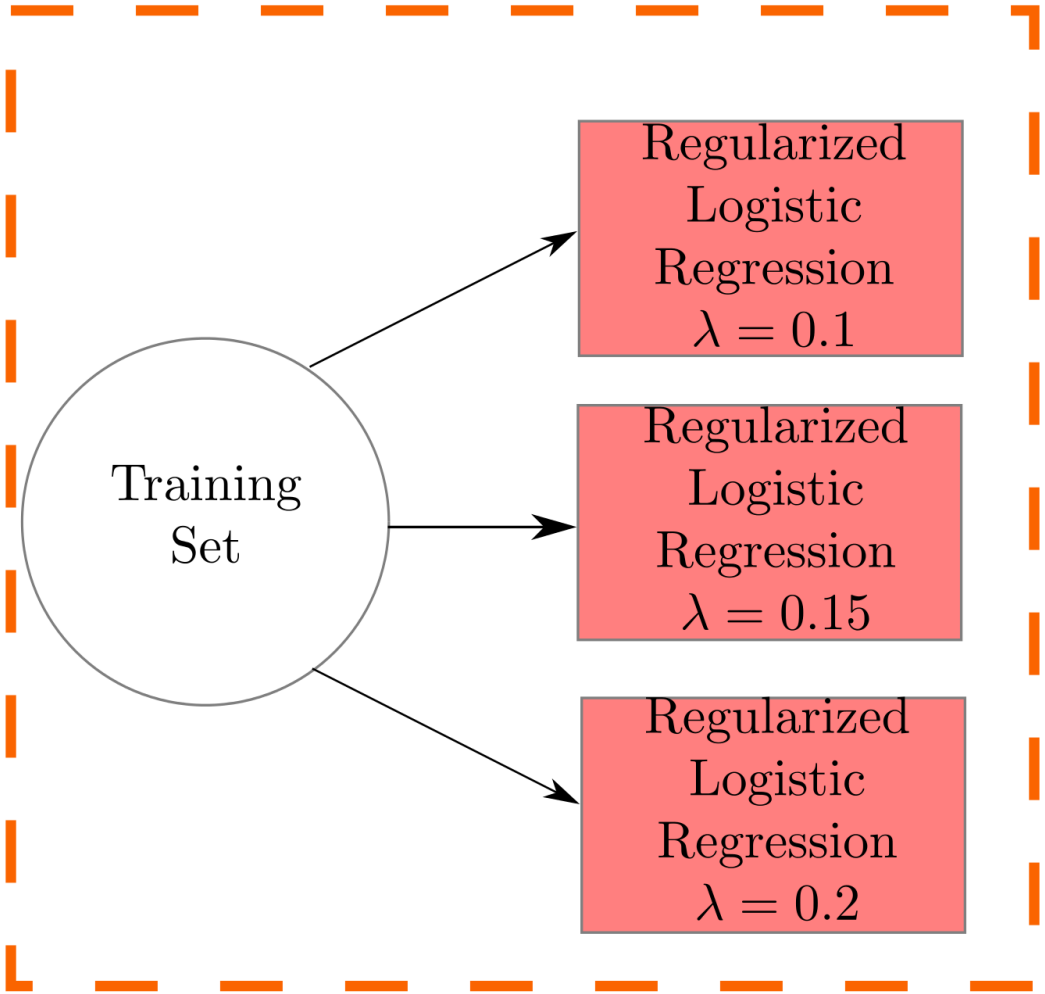
Ensemble methods : boosting - Illustration



[Bishop 2006]

Ensemble methods : stacking - Procedure

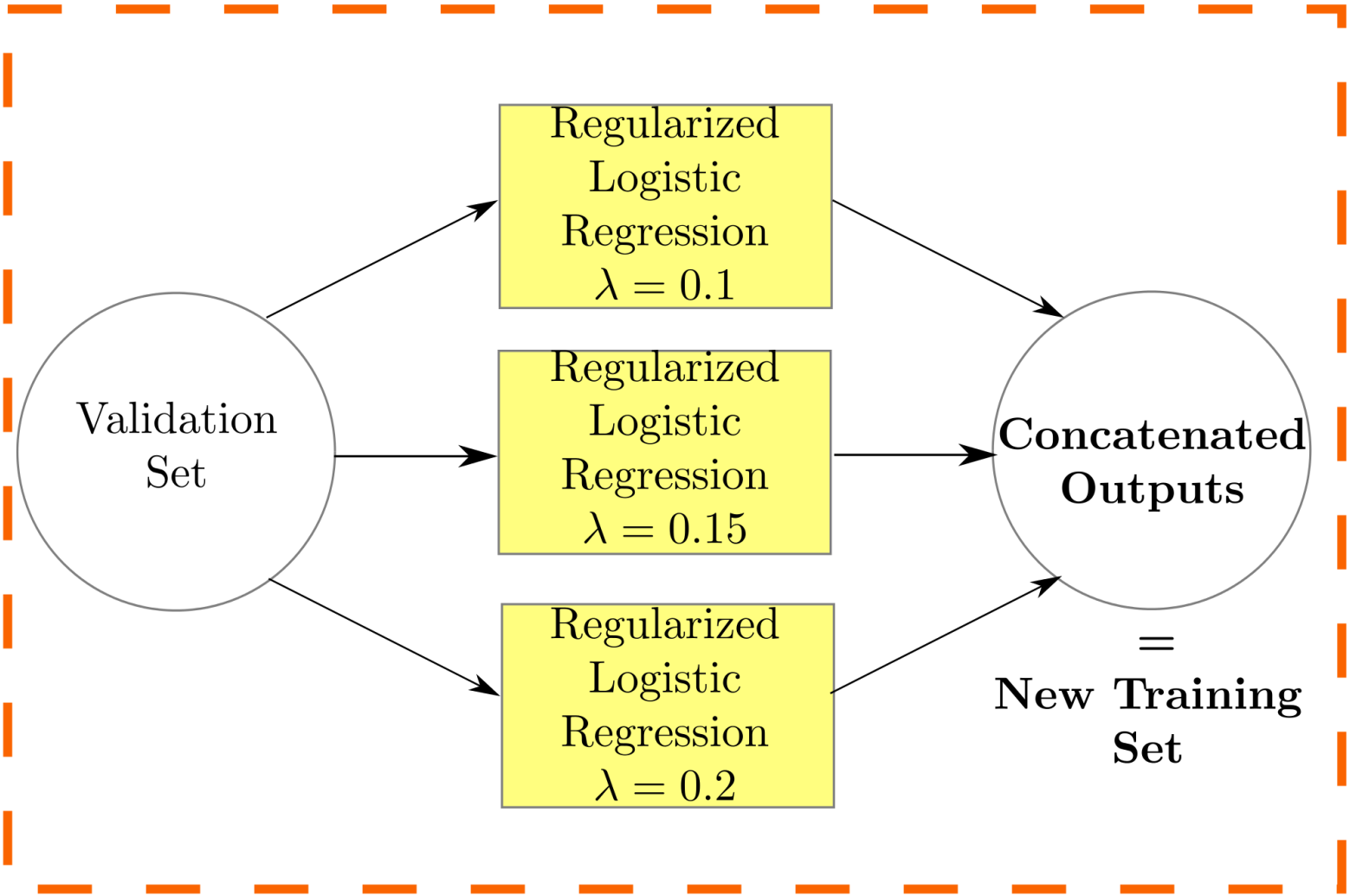
Train each member of the ensemble.



Step 1

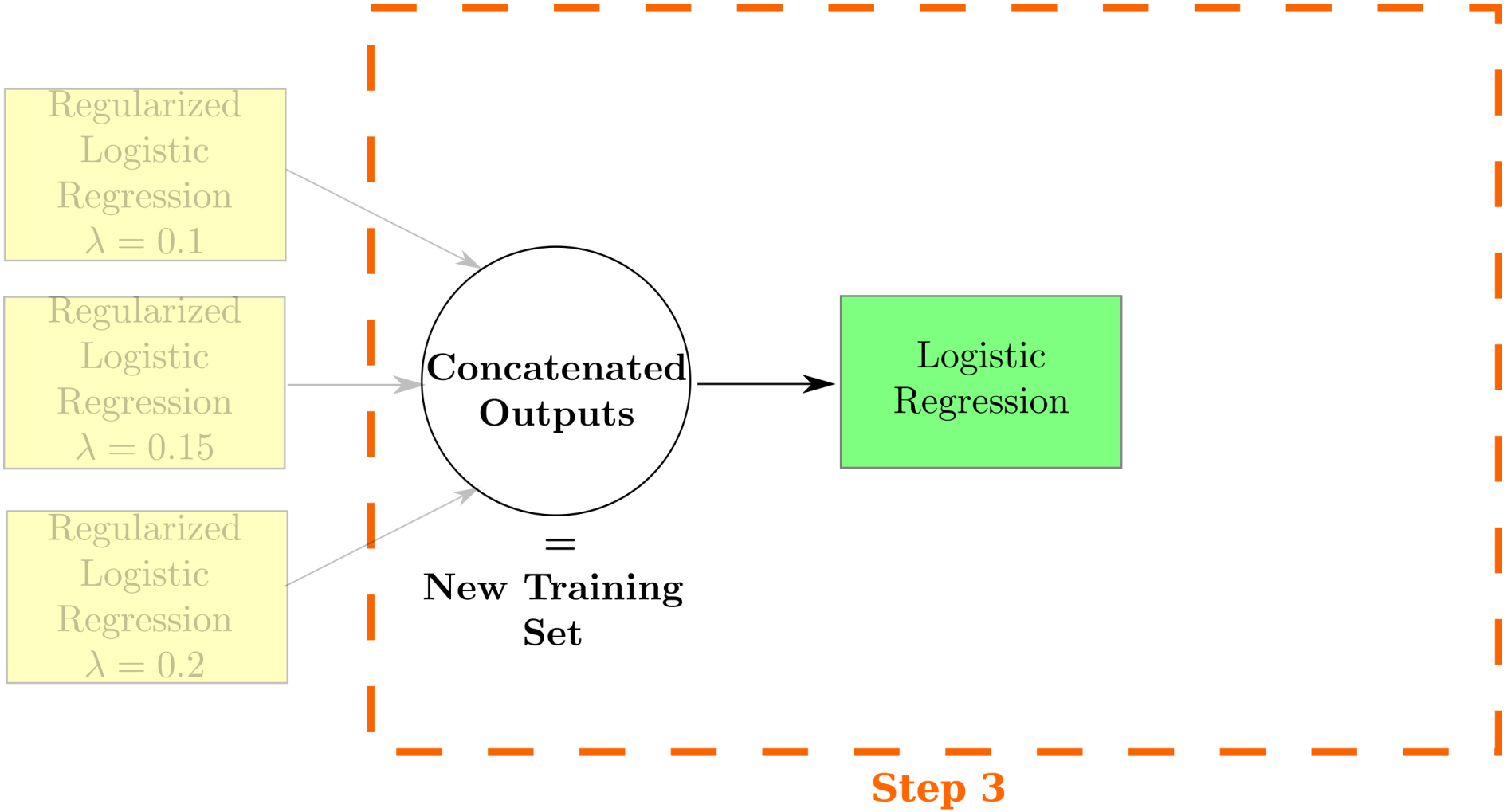
Ensemble methods : stacking - Procedure

Generate a new training set for the fusion op.



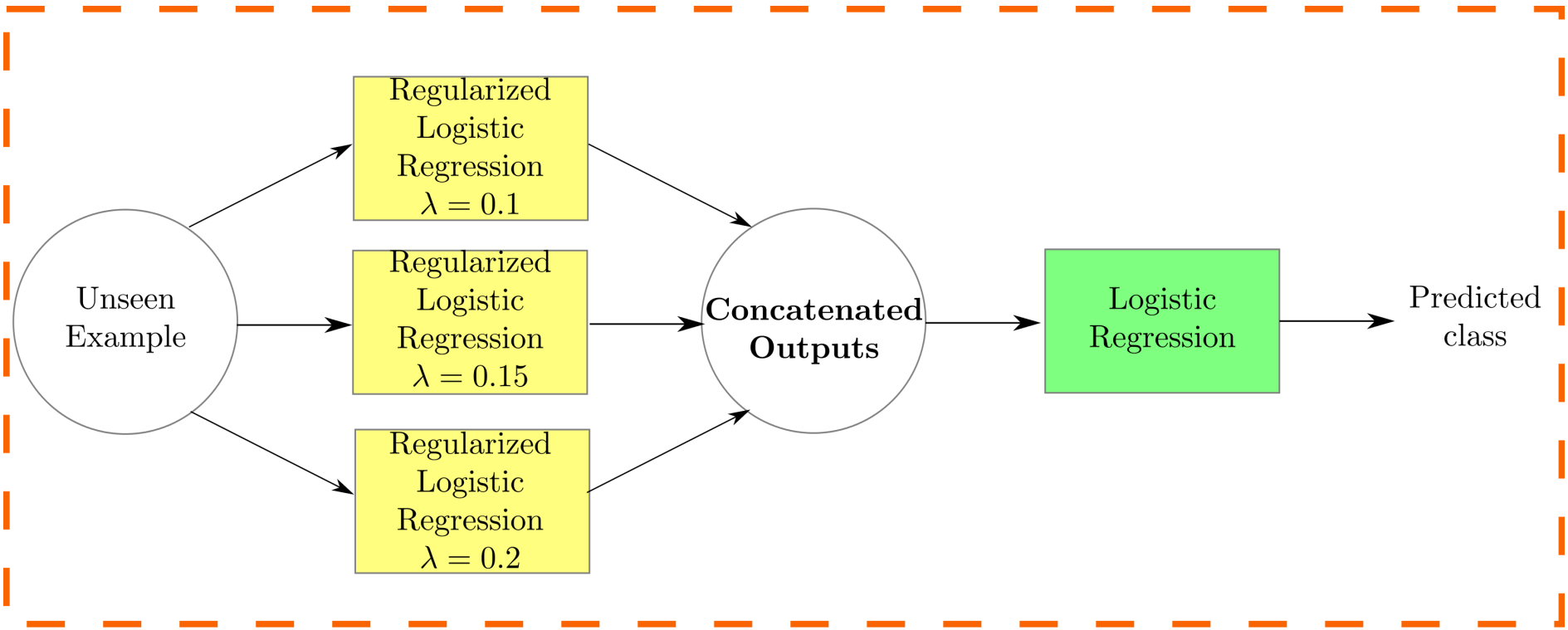
Ensemble methods : stacking - Procedure

Train the fusion op.



Ensemble methods : stacking - Procedure

Use it all.



Step 4

Ensemble methods : stacking - Comments

- Stacking can be used for heterogeneous classifiers too.
- Unlike mixture models, stacking need private data for training the fusion op.
- Solution : repeat the procedure in a cross validation (CV) fashion.
- Supposing one performs leave-one-out CV (**LOOCV**) and let $f_k^{(-i)}$ denote the k^{th} training on the whole training set **except** data point $\mathbf{x}^{(i)}$, then in step 3, one needs to minimize

$$J(\mathbf{w}) = \sum_{i=1}^{n} L\left(y^{(i)}, \operatorname{sgm}\left(\sum_{k=1}^{m} w_k f_k^{(-i)}\right)\right).$$

Ensemble methods : stacking - Comments

- Stacking can be used for heterogeneous classifiers too.
- Unlike mixture models, stacking need private data for training the fusion op.
- Solution : repeat the procedure in a cross validation (CV) fashion.
- Supposing one performs leave-one-out CV (**LOOCV**) and let $f_k^{(-i)}$ denote the k^{th} training on the whole training set **except** data point $\mathbf{x}^{(i)}$, then in step 3, one needs to minimize

$$J(\mathbf{w}) = \sum_{i=1}^{n} L\left(y^{(i)}, \text{sgm}\left(\sum_{k=1}^{m} w_k f_k^{(-i)}\right)\right).$$

Ensemble methods : stacking - Comments

- Stacking can be used for heterogeneous classifiers too.
- Unlike mixture models, stacking need private data for training the fusion op.
- Solution : repeat the procedure in a cross validation (CV) fashion.
- Supposing one performs leave-one-out CV (**LOOCV**) and let $f_k^{(-i)}$ denote the k^{th} training on the whole training set **except** data point $\mathbf{x}^{(i)}$, then in step 3, one needs to minimize

$$J(\mathbf{w}) = \sum_{i=1}^{n} L\left(y^{(i)}, \operatorname{sgm}\left(\sum_{k=1}^{m} w_k f_k^{(-i)}\right)\right).$$

Ensemble methods : stacking - Comments

- Stacking can be used for heterogeneous classifiers too.
- Unlike mixture models, stacking need private data for training the fusion op.
- Solution : repeat the procedure in a cross validation (CV) fashion.
- Supposing one performs leave-one-out CV (**LOOCV**) and let $f_k^{(-i)}$ denote the k^{th} training on the whole training set **except** data point $\mathbf{x}^{(i)}$, then in step 3, one needs to minimize

$$J(\mathbf{w}) = \sum_{i=1}^{n} L\left(y^{(i)}, \operatorname{sgm}\left(\sum_{k=1}^{m} w_k f_k^{(-i)}\right)\right).$$

Ensemble methods : Error-correcting output codes

- Suppose examples **x** need to be sorted into $\ell > 2$ classes.
- One can assign a code encoded as a bit-vector **b** for each class.
- For example :

Class	b_1	b_2	b_3	b_4	b_5	b_6
a	1	1	0	0	0	0
b	1	0	1	1	0	0
c	0	0	0	0	1	1

- This encription uses more bits than necessary.
- Hamming distance : nbr. of unequal bits

Ensemble methods : Error-correcting output codes

- Suppose examples **x** need to be sorted into $\ell > 2$ classes.
- One can assign a code encoded as a bit-vector **b** for each class.
- For example :

Class	b_1	b_2	b_3	b_4	b_5	b_6
a	1	1	0	0	0	0
b	1	0	1	1	0	0
c	0	0	0	0	1	1

- This encription uses more bits than necessary.
- Hamming distance : nbr. of unequal bits

Ensemble methods : Error-correcting output codes

- Suppose examples **x** need to be sorted into $\ell > 2$ classes.
- One can assign a code encoded as a bit-vector **b** for each class.
- For example :

Class	b_1	b_2	b_3	b_4	b_5	b_6
a	1	1	0	0	0	0
b	1	0	1	1	0	0
c	0	0	0	0	1	1

- This encription uses more bits than necessary.
- Hamming distance : nbr. of unequal bits

Ensemble methods : Error-correcting output codes

- Suppose examples **x** need to be sorted into $\ell > 2$ classes.
- One can assign a code encoded as a bit-vector **b** for each class.
- For example :

Class	b_1	b_2	b_3	b_4	b_5	b_6
a	1	1	0	0	0	0
b	1	0	1	1	0	0
c	0	0	0	0	1	1

- This encription uses more bits than necessary.
- Hamming distance : nbr. of unequal bits

Ensemble methods : Error-correcting output codes

- Suppose examples **x** need to be sorted into $\ell > 2$ classes.
- One can assign a code encoded as a bit-vector **b** for each class.
- For example :

Class	b_1	b_2	b_3	b_4	b_5	b_6
a	1	1	0	0	0	0
b	1	0	1	1	0	0
c	0	0	0	0	1	1

- This encription uses more bits than necessary.
- Hamming distance : nbr. of unequal bits

Ensemble methods : Error-correcting output codes

- The minimal Hamming distance between each pair of codes is 3.
- Suppose we train $m = 6$ different classifiers allowing to estimate probabilities $p_{b_i}(\mathbf{x})$ (ex : LogReg).
- A class is decided for example $\mathbf{x}$ according to the rule :

$$\arg\min_{y} \sum_{i=1}^{m} |b_i(y) - \hat{p}_{b_i}(\mathbf{x})|$$

- This method is more robust to bit-flips, i.e. classification errors.

Ensemble methods : Error-correcting output codes

- The minimal Hamming distance between each pair of codes is 3.
- Suppose we train $m = 6$ different classifiers allowing to estimate probabilities $p_{b_i}(\mathbf{x})$ (ex : LogReg).
- A class is decided for example $\mathbf{x}$ according to the rule :

$$\arg\min_{y} \sum_{i=1}^{m} |b_i(y) - \hat{p}_{b_i}(\mathbf{x})|$$

- This method is more robust to bit-flips, i.e. classification errors.

Ensemble methods : Error-correcting output codes

- The minimal Hamming distance between each pair of codes is 3.
- Suppose we train $m = 6$ different classifiers allowing to estimate probabilities $p_{b_i}(\mathbf{x})$ (ex : LogReg).
- A class is decided for example $\mathbf{x}$ according to the rule :

$$\arg\min_{y} \sum_{i=1}^{m} |b_i(y) - \hat{p}_{b_i}(\mathbf{x})|$$

- This method is more robust to bit-flips, i.e. classification errors.

Ensemble methods : Error-correcting output codes

- The minimal Hamming distance between each pair of codes is 3.
- Suppose we train $m = 6$ different classifiers allowing to estimate probabilities $p_{b_i}(\mathbf{x})$ (ex : LogReg).
- A class is decided for example $\mathbf{x}$ according to the rule :

$$\arg\min_{y} \sum_{i=1}^{m} |b_i(y) - \hat{p}_{b_i}(\mathbf{x})|$$

- This method is more robust to bit-flips, i.e. classification errors.

Chapter organization

Bayesian Learning :

- A catch sentence for this chapter could be :

  « Why use only one classifier when I can use many ? »

- With Bayesian learning, this would become :

  « Why use only one classifier when I can use infinitely many ? »

- Let us see under which circumstances such a result can be achieved.

Bayesian Learning :

- A catch sentence for this chapter could be :

  « Why use only one classifier when I can use many ? »

- With Bayesian learning, this would become :

  « Why use only one classifier when I can use infinitely many ? »

- Let us see under which circumstances such a result can be achieved.

Bayesian Learning :

- A catch sentence for this chapter could be :

  « Why use only one classifier when I can use many ? »

- With Bayesian learning, this would become :

  « Why use only one classifier when I can use infinitely many ? »

- Let us see under which circumstances such a result can be achieved.

Bayesian Learning : - starting point

- Most of learning algorithms translate into an optimization problem of the following kind :

$$\arg\min_{\boldsymbol{\theta}} \quad \mathrm{DataFit}\left(\boldsymbol{\theta}\right) \quad + \quad \mathrm{Regularizer}\left(\boldsymbol{\theta}\right).$$

- In this setting, each $f \in \mathcal{H}$ is in bijective correspondance with a given $\theta \in \Theta$.

- Almost all such algorithms have an equivalent probabilistic formulation :

$$\arg\max_{\theta} \quad \mathrm{Likelihood}\left(\theta\right) \quad \times \quad \mathrm{Prior}\left(\theta\right).$$

Bayesian Learning : - starting point

- Most of learning algorithms translate into an optimization problem of the following kind :

$$\arg\min_{\boldsymbol{\theta}} \quad \mathrm{DataFit}\left(\boldsymbol{\theta}\right) \quad + \quad \mathrm{Regularizer}\left(\boldsymbol{\theta}\right).$$

- In this setting, each $f \in \mathcal{H}$ is in bijective correspondance with a given $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

- Almost all such algorithms have an equivalent probabilistic formulation :

$$\arg\max_{\theta} \quad \mathrm{Likelihood}\left(\theta\right) \quad \times \quad \mathrm{Prior}\left(\theta\right).$$

Bayesian Learning : - starting point

- Most of learning algorithms translate into an optimization problem of the following kind :

$$\arg\min_{\boldsymbol{\theta}} \quad \mathrm{DataFit}\left(\boldsymbol{\theta}\right) \quad + \quad \mathrm{Regularizer}\left(\boldsymbol{\theta}\right).$$

- In this setting, each $f \in \mathcal{H}$ is in bijective correspondance with a given $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

- Almost all such algorithms have an equivalent probabilistic formulation :

$$\arg\max_{\boldsymbol{\theta}} \quad \mathrm{Likelihood}\left(\boldsymbol{\theta}\right) \quad \times \quad \mathrm{Prior}\left(\boldsymbol{\theta}\right).$$

Bayesian Learning : - Linear regression example

- Suppose we are trying to predict the selling price y of a house.
- For each house, we collected data like surface, previous buying price, GPS coordinates, etc.
- These features are concatenated into a vector $\mathbf{x}$;
- We need to learn the function f_0 mapping vectors $\mathbf{x}$ to y.
- We believe a linear combination of the features shoud be a relevant model :

$$y = \theta^T \cdot \mathbf{x}.$$

Bayesian Learning : - Linear regression example

- Suppose we are trying to predict the selling price y of a house.
- For each house, we collected data like surface, previous buying price, GPS coordinates, etc.
- These features are concatenated into a vector $\mathbf{x}$;
- We need to learn the function f_0 mapping vectors $\mathbf{x}$ to y.
- We believe a linear combination of the features shoud be a relevant model :

$$y = \theta^T \cdot \mathbf{x}.$$

Bayesian Learning : - Linear regression example

- Suppose we are trying to predict the selling price y of a house.
- For each house, we collected data like surface, previous buying price, GPS coordinates, etc.
- These features are concatenated into a vector $\mathbf{x}$;
- We need to learn the function f_0 mapping vectors $\mathbf{x}$ to y.
- We believe a linear combination of the features shoud be a relevant model :

$$y = \theta^T \cdot \mathbf{x}.$$

Bayesian Learning : - Linear regression example

- Suppose we are trying to predict the selling price y of a house.
- For each house, we collected data like surface, previous buying price, GPS coordinates, etc.
- These features are concatenated into a vector $\mathbf{x}$;
- We need to learn the function f_0 mapping vectors $\mathbf{x}$ to y.
- We believe a linear combination of the features shoud be a relevant model :

$$y = \theta^T \cdot \mathbf{x}.$$

Bayesian Learning : - Linear regression example

- Suppose we are trying to predict the selling price y of a house.
- For each house, we collected data like surface, previous buying price, GPS coordinates, etc.
- These features are concatenated into a vector $\mathbf{x}$;
- We need to learn the function f_0 mapping vectors $\mathbf{x}$ to y.
- We believe a linear combination of the features shoud be a relevant model :

$$y = \theta^T \cdot \mathbf{x}.$$

Bayesian Learning : - Linear regression example

- Yet we also believe that this linear combination is just an approximation of f_0 and therefore we go for a probabilistic formulation :

$$Y \sim \mathcal{N}\left(\boldsymbol{\theta}^T \cdot \mathbf{x}, \sigma\right).$$

- Now the likelihood is given by :

$$\text{Likelihood}\left(\theta\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y^{(i)} - \theta^T \cdot \mathrm{x}^{(i)}\right)^2}{2\sigma^2}}$$

- For simplicity, we assume the noise variance σ^2 is known.

Bayesian Learning : - Linear regression example

- Yet we also believe that this linear combination is just an approximation of f_0 and therefore we go for a probabilistic formulation :

$$Y \sim \mathcal{N}\left(\boldsymbol{\theta}^T \cdot \mathbf{x}, \sigma\right).$$

- Now the likelihood is given by :

$$\text{Likelihood}\left(\boldsymbol{\theta}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y^{(i)} - \boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma^2}}$$

- For simplicity, we assume the noise variance σ^2 is known.

Bayesian Learning : - Linear regression example

- Yet we also believe that this linear combination is just an approximation of f_0 and therefore we go for a probabilistic formulation :

$$Y \sim \mathcal{N}\left(\boldsymbol{\theta}^T \cdot \mathbf{x}, \sigma\right).$$

- Now the likelihood is given by :

$$\text{Likelihood}\left(\boldsymbol{\theta}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y^{(i)} - \boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma^2}}$$

- For simplicity, we assume the noise variance σ^2 is known.

Bayesian Learning : - Linear regression example

- We already have some knowledge on what values of $\boldsymbol{\theta}$ are more likely before seeing any datum :

$$\text{Prior}\left(\boldsymbol{\theta}\right) = \frac{1}{\left(2\pi\right)^{\frac{d}{2}} \det\left(\mathbf{V}_0\right)^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_0\right)^T \cdot \mathbf{V}_0^{-1}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_0\right)}$$

- After seeing data $\mathcal{D}$, our knowledge is given by the following posterior distribution

$$p\left(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\theta}_0, \mathbf{V}_0\right) \quad \propto \quad \text{Likelihood}\left(\boldsymbol{\theta}\right) \quad \times \quad \text{Prior}\left(\boldsymbol{\theta}\right).$$

Bayesian Learning : - Linear regression example

- We already have some knowledge on what values of $\boldsymbol{\theta}$ are more likely before seeing any datum :

$$\text{Prior}\left(\boldsymbol{\theta}\right) = \frac{1}{\left(2\pi\right)^{\frac{d}{2}} \det\left(\mathbf{V}_0\right)^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_0\right)^T \cdot \mathbf{V}_0^{-1}\left(\boldsymbol{\theta}-\boldsymbol{\theta}_0\right)}$$

- After seeing data $\mathcal{D}$, our knowledge is given by the following posterior distribution

$$p\left(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\theta}_0, \mathbf{V}_0\right) \quad \propto \quad \text{Likelihood}\left(\boldsymbol{\theta}\right) \quad \times \quad \text{Prior}\left(\boldsymbol{\theta}\right).$$

Bayesian Learning : - Linear regression example

- If the prior parameters are such that $\boldsymbol{\theta}_0 = \mathbf{0}$ and $\mathbf{V}_0 = \tau^2 \mathbf{I}$, applying $-\log$ leads to the following cost function (up to an additive constant)

$$J(\boldsymbol{\theta}) = \underbrace{\sum_{i=1}^{n} \frac{\left(y^{(i)} - \boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)}\right)^2}{2\sigma^2}}_{\text{Least Squares}} + \underbrace{\frac{1}{\tau^2} \|\boldsymbol{\theta}\|_2}_{\text{Ridge Reg.}}$$

Bayesian Learning : - Linear regression example

- Going back to probabilities, one can show that the posterior $p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\theta}_0, \mathbf{V}_0)$ is also Gaussian, in which case our prior is conjugate[2].

$$p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\theta}_0, \mathbf{V}_0) \sim \mathcal{N}(\boldsymbol{\theta}_n, \mathbf{V}_n), \quad (1)$$

$$\boldsymbol{\theta}_n = \mathbf{V}_n \left(\mathbf{V}_0^{-1} \cdot \boldsymbol{\theta}_0 + \frac{1}{\sigma^2} \mathbf{X}^T \cdot \mathbf{y} \right), \quad (2)$$

$$\mathbf{V}_n = \left(\mathbf{V}_0^{-1} + \frac{1}{\sigma^2} \mathbf{X}^T \cdot \mathbf{X} \right)^{-1}, \quad (3)$$

with

$$\mathbf{X} = \begin{pmatrix} — \left(\mathbf{x}^{(1)}\right)^T — \\ \vdots \\ — \left(\mathbf{x}^{(n)}\right)^T — \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix}.$$

2. Under conjugacy, learning boils down to updating the prior parameters and the updates are easy to compute.

Bayesian Learning : - Linear regression example

- No fusion for now .. just Bayesian statistics !

- As learners, what we really need is the posterior predictive $p(y|\mathbf{x}, \mathcal{D})$.

- The expectation of this distribution is our proxy for f_0 and allows to make a prediction for the selling price of a house whose features are the entries of the unseen example $\mathbf{x}$.

Bayesian Learning : - Linear regression example

- No fusion for now .. just Bayesian statistics !
- As learners, what we really need is the posterior predictive $p(y|\mathbf{x}, \mathcal{D})$.
- The expectation of this distribution is our proxy for f_0 and allows to make a prediction for the selling price of a house whose features are the entries of the unseen example $\mathbf{x}$.

Bayesian Learning : - Linear regression example

- No fusion for now .. just Bayesian statistics !
- As learners, what we really need is the posterior predictive $p(y|\mathbf{x}, \mathcal{D})$.
- The expectation of this distribution is our proxy for f_0 and allows to make a prediction for the selling price of a house whose features are the entries of the unseen example $\mathbf{x}$.

Bayesian Learning : - Linear regression example

- Observe that the predictive distribution is free of un-observed parameter conditioning... because we marginalized them out :

$$p\left(y|\mathbf{x},\mathcal{D}\right) = \int_{\Theta} p\left(y|\mathbf{x},\mathcal{D},\theta\right) p\left(\theta|\mathbf{x},\mathcal{D}\right) d\theta \tag{4}$$

- The above calculus is the weighted combination of an infinity of regressors !

- The weights depend on the ability of each regressor to fit well the data.

Bayesian Learning : - Linear regression example

- Observe that the predictive distribution is free of un-observed parameter conditioning... because we marginalized them out :

$$p\left(y|\mathbf{x},\mathcal{D}\right) = \int_{\Theta} p\left(y|\mathbf{x},\mathcal{D},\theta\right) p\left(\theta|\mathbf{x},\mathcal{D}\right) d\theta \tag{4}$$

- The above calculus is the weighted combination of an infinity of regressors !

- The weights depend on the ability of each regressor to fit well the data.

Bayesian Learning : - Linear regression example

- Observe that the predictive distribution is free of un-observed parameter conditioning... because we marginalized them out :

$$p\left(y|\mathbf{x},\mathcal{D}\right) = \int_{\Theta} p\left(y|\mathbf{x},\mathcal{D},\theta\right) p\left(\theta|\mathbf{x},\mathcal{D}\right) d\theta \tag{4}$$

- The above calculus is the weighted combination of an infinity of regressors !

- The weights depend on the ability of each regressor to fit well the data.

Bayesian Learning : - Linear regression example

- In our linear regression case, we have

$$p(y|\mathbf{x},\mathcal{D}) = \int_{\Theta} p(y|\mathbf{x},\mathcal{D},\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathbf{x},\mathcal{D})\, d\boldsymbol{\theta}, \quad (5)$$

$$= \int_{\Theta} p(y|\mathbf{x},\mathcal{D},\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}, \quad (6)$$

$$= \int_{\Theta} G\left(y; \boldsymbol{\theta}^T \cdot \mathbf{x}, \sigma^2\right) G(\boldsymbol{\theta}; \boldsymbol{\theta}_n, \mathbf{V}_n)\, d\boldsymbol{\theta}, \quad (7)$$

with G the Gaussian density function.

- Finally, one can show that

$$y|\mathbf{x},\mathcal{D} \sim \mathcal{N}\left(\boldsymbol{\theta}_n^T \cdot \mathbf{x}, \sigma_n\right), \quad (8)$$

$$\sigma_n^2 = \sigma^2 + \mathbf{x}^T \cdot \mathbf{V}_n \cdot \mathbf{x}. \quad (9)$$

Bayesian Learning : - Linear regression example

- In our linear regression case, we have

$$p(y|\mathbf{x},\mathcal{D}) = \int_{\Theta} p(y|\mathbf{x},\mathcal{D},\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathbf{x},\mathcal{D})\, d\boldsymbol{\theta}, \quad (5)$$

$$= \int_{\Theta} p(y|\mathbf{x},\mathcal{D},\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}, \quad (6)$$

$$= \int_{\Theta} G\left(y;\boldsymbol{\theta}^T\cdot\mathbf{x},\sigma^2\right) G(\boldsymbol{\theta};\boldsymbol{\theta}_n,\mathbf{V}_n)\, d\boldsymbol{\theta}, \quad (7)$$

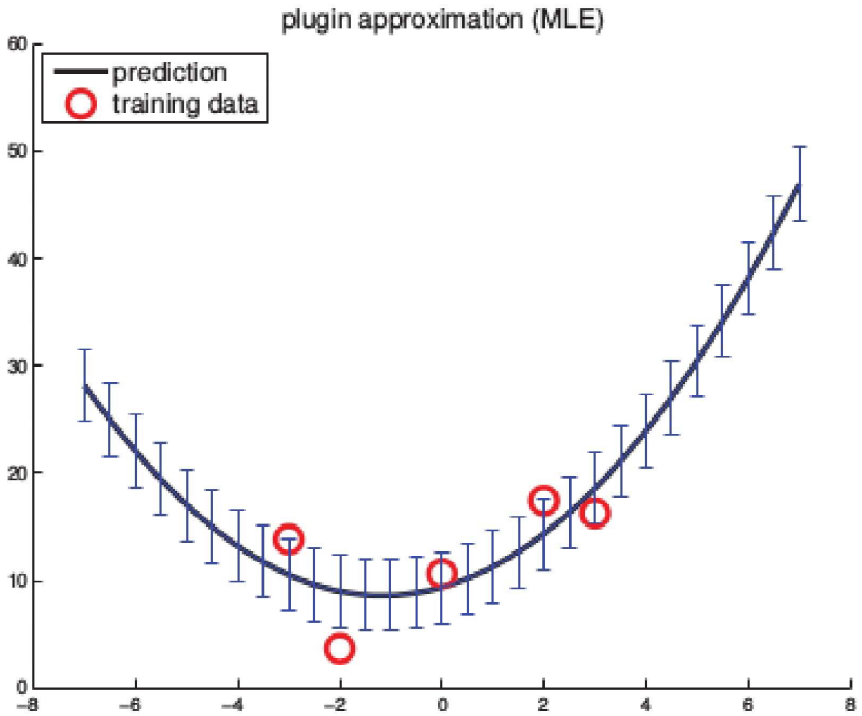
with G the Gaussian density function.

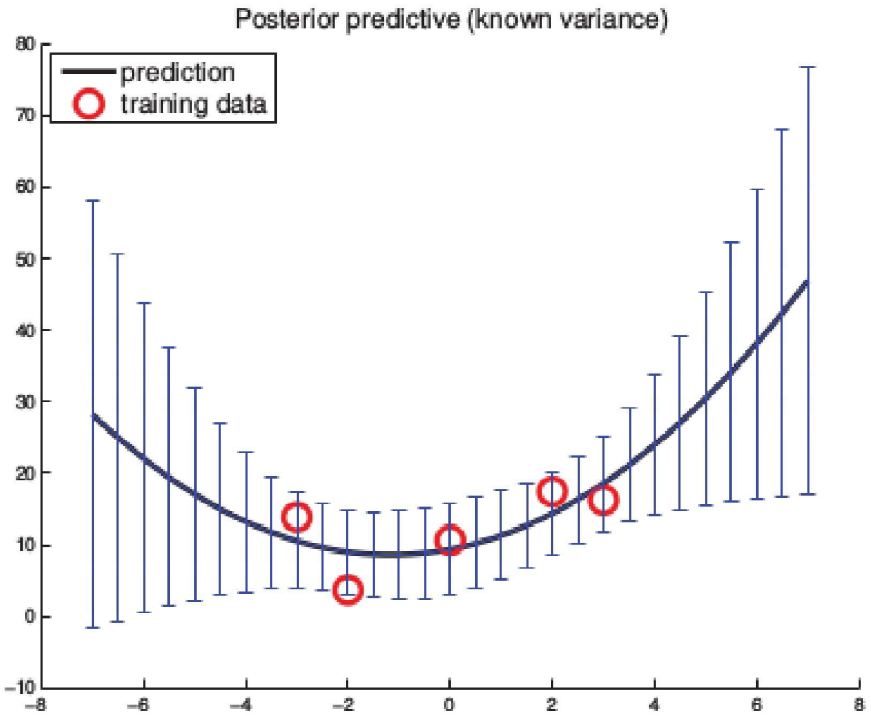
- Finally, one can show that

$$y|\mathbf{x},\mathcal{D} \sim \mathcal{N}\left(\boldsymbol{\theta}_n^T\cdot\mathbf{x},\sigma_n\right), \quad (8)$$

$$\sigma_n^2 = \sigma^2 + \mathbf{x}^T\cdot\mathbf{V}_n\cdot\mathbf{x}. \quad (9)$$

Bayesian Learning : - Illustration (polynomial reg.)





[Murphy 2012 - 7.6]

Bayesian Learning : - Comments

- The posterior predictive is not always known in closed form $\longrightarrow$ use Monte-Carlo to approximate the marginalization.
- Have we really gotten rid of all the parameters ?
- No, we are still conditioning w.r.t. $\boldsymbol{\theta}_0 = \mathbf{0}$ and $\mathbf{V}_0$.
- They can be marginalized out too by introducing a distribution for them called a hyperprior. This setting is known as hierarchical Bayes.

Bayesian Learning : - Comments

- The posterior predictive is not always known in closed form $\longrightarrow$ use Monte-Carlo to approximate the marginalization.
- Have we really gotten rid of all the parameters ?
- No, we are still conditioning w.r.t. $\boldsymbol{\theta}_0 = \mathbf{0}$ and $\mathbf{V}_0$.
- They can be marginalized out too by introducing a distribution for them called a hyperprior. This setting is known as hierarchical Bayes.

Bayesian Learning : - Comments

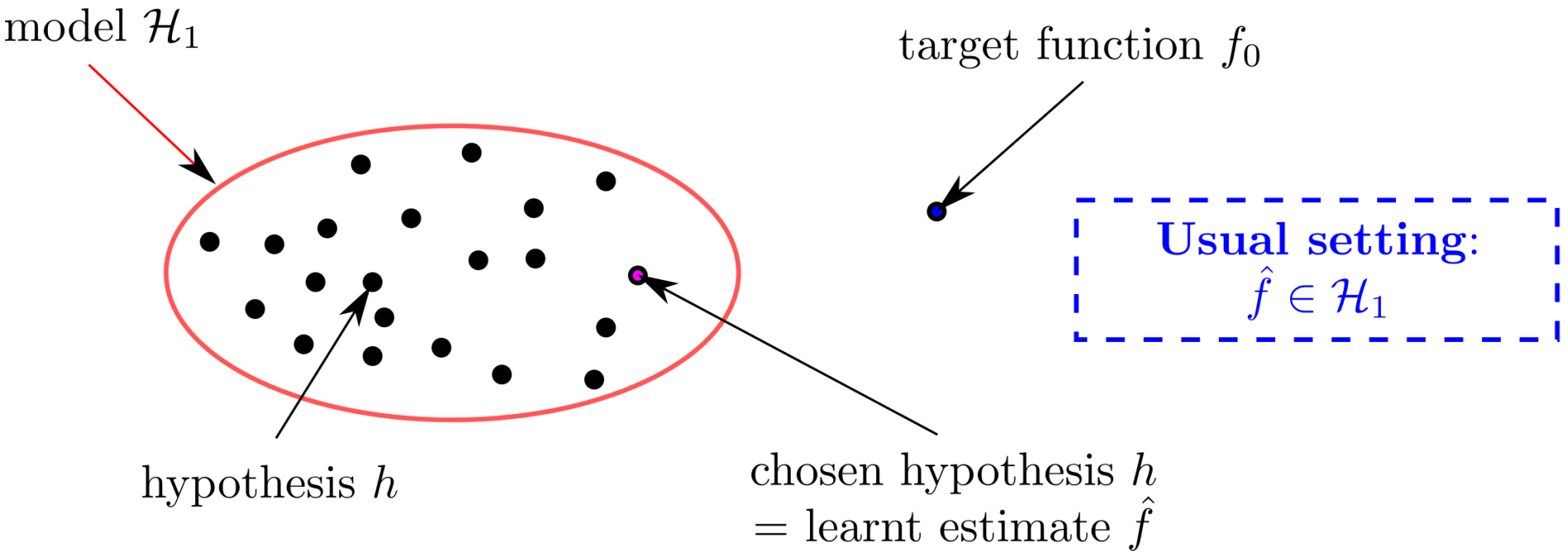
- The posterior predictive is not always known in closed form $\longrightarrow$ use Monte-Carlo to approximate the marginalization.
- Have we really gotten rid of all the parameters ?
- No, we are still conditioning w.r.t. $\boldsymbol{\theta}_0 = \mathbf{0}$ and $\mathbf{V}_0$.
- They can be marginalized out too by introducing a distribution for them called a hyperprior. This setting is known as hierarchical Bayes.

Bayesian Learning : - Comments

- The posterior predictive is not always known in closed form $\longrightarrow$ use Monte-Carlo to approximate the marginalization.
- Have we really gotten rid of all the parameters ?
- No, we are still conditioning w.r.t. $\boldsymbol{\theta}_0 = \mathbf{0}$ and $\mathbf{V}_0$.
- They can be marginalized out too by introducing a distribution for them called a hyperprior. This setting is known as hierarchical Bayes.

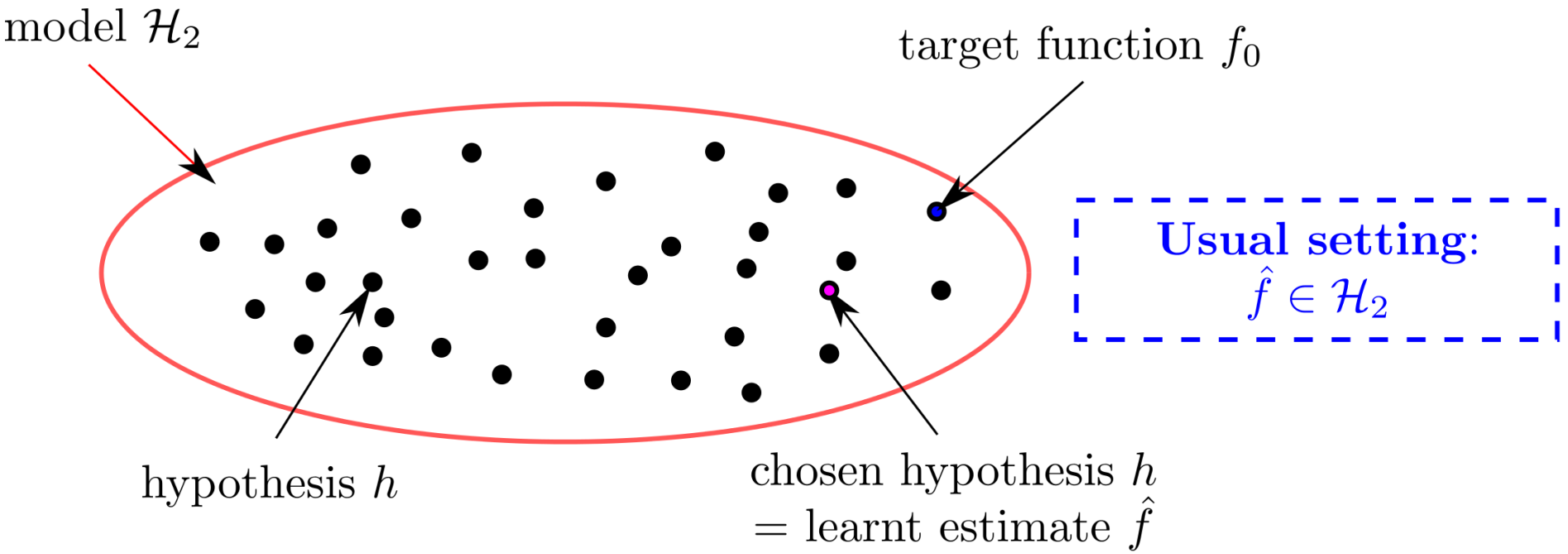
Bayesian Model Averaging : let's start with model selection

Example : Polynomial regression with small degree $q = 1$



Bayesian Model Averaging : let's start with model selection

Example : Polynomial regression with higher degree $q = 2$



Bayesian Model Averaging :

Example : Polynomial regression with degree q

- In model selection, the candidate value for q is sought using, for example, CV.
  In general, it could be obtained as

$$q^* = \arg\max_{q \in \mathbb{N}^*} p\left(q|\mathcal{D}\right).$$

- In model averaging, several values for q are considered as relevant candidates. We are now elaborating the predictive posterior as

$$\begin{aligned} p\left(y|\mathbf{x},\mathcal{D}\right) &= \sum_{q \in \mathbb{N}^*} p\left(y|\mathbf{x},\mathcal{D},q\right) p\left(q|\mathbf{x},\mathcal{D}\right), \\ &= \sum_{q \in \mathbb{N}^*} p\left(y|\mathbf{x},\mathcal{D},q\right) p\left(q|\mathcal{D}\right). \end{aligned}$$

Bayesian Model Averaging :

Example : Polynomial regression with degree q

- In model selection, the candidate value for q is sought using, for example, CV.
  In general, it could be obtained as

$$q^* = \arg\max_{q \in \mathbb{N}^*} p\left(q|\mathcal{D}\right).$$

- In model averaging, several values for q are considered as relevant candidates. We are now elaborating the predictive posterior as

$$\begin{aligned} p\left(y|\mathbf{x},\mathcal{D}\right) &= \sum_{q \in \mathbb{N}^*} p\left(y|\mathbf{x},\mathcal{D},q\right) p\left(q|\mathbf{x},\mathcal{D}\right), \\ &= \sum_{q \in \mathbb{N}^*} p\left(y|\mathbf{x},\mathcal{D},q\right) p\left(q|\mathcal{D}\right). \end{aligned}$$

Bayesian Model Averaging : comments

- BMA only works for probabistic models allowing to determine both $p(q|\mathcal{D})$ and $p(y|\mathbf{x}, \mathcal{D}, q)$.

- Its philosophy is close to hierarchical Bayes in the sense that each hyperprior parameter choice can be regarded as a given model.

Bayesian Model Averaging : comments

- BMA only works for probabistic models allowing to determine both $p(q|\mathcal{D})$ and $p(y|\mathbf{x}, \mathcal{D}, q)$.

- Its philosophy is close to hierarchical Bayes in the sense that each hyperprior parameter choice can be regarded as a given model.

Chapter organization

Statistical aggregation theory :

- The satistical theory of aggregation formalizes the intuitions behind the fusion of prediction functions in a supervised learning context.

- Its main contribution are generalization performance waranties.

Statistical aggregation theory :

- The satistical theory of aggregation formalizes the intuitions behind the fusion of prediction functions in a supervised learning context.

- Its main contribution are generalization performance waranties.

Statistical aggregation theory : problem statement

- For simplicity, suppose the hypothesis set [3] $\mathcal{H}$ is finite : $|\mathcal{H}| = m < \infty$.
- Let $\mathcal{H}_\Theta$ denote the following space :

$$\mathcal{H}_\Theta = \left\{ f_\theta = \sum_{j=1}^{m} \theta_j f_j : \theta \in \Theta \right\}.$$

- $\mathcal{H}_\Theta$ is the functional vector space spanned by the members of $\mathcal{H}$.

3. often called dictionnary in this framework.

Statistical aggregation theory : problem statement

- For simplicity, suppose the hypothesis set [3] $\mathcal{H}$ is finite : $|\mathcal{H}| = m < \infty$.

- Let $\mathcal{H}_\Theta$ denote the following space :

$$\mathcal{H}_\Theta = \left\{ f_{\boldsymbol{\theta}} = \sum_{j=1}^{m} \theta_j f_j : \boldsymbol{\theta} \in \Theta \right\}.$$

- $\mathcal{H}_\Theta$ is the functional vector space spanned by the members of $\mathcal{H}$.

3. often called dictionnary in this framework.

Statistical aggregation theory : problem statement

- For simplicity, suppose the hypothesis set [3] $\mathcal{H}$ is finite : $|\mathcal{H}| = m < \infty$.

- Let $\mathcal{H}_\Theta$ denote the following space :

$$\mathcal{H}_\Theta = \left\{ f_{\boldsymbol{\theta}} = \sum_{j=1}^{m} \theta_j f_j : \boldsymbol{\theta} \in \Theta \right\}.$$

- $\mathcal{H}_\Theta$ is the functional vector space spanned by the members of $\mathcal{H}$.

3. often called dictionnary in this framework.

Statistical aggregation theory : problem statement

Depending the $\Theta \subseteq \mathbb{R}^m$, several sub-classes of problems are obtained :

- **Model selection** : Θ is the set of canonical base vectors, or one-hot vectors.

$$\theta^T = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}$$

- **Convex aggregation** : Θ is the canonical simplex of $\mathbb{R}^m$,

$$\sum_{j=1}^{m} \theta_j = 1 \text{ and } \theta_j \geq 0.$$

- **Linear aggregation** : $\Theta = \mathbb{R}^m$.

- **Sparse linear aggregation** : $\Theta = \{\theta \in \mathbb{R}^m \text{ s.t. } \|\theta\|_0 < r\}$.

Statistical aggregation theory : problem statement

Depending the $\Theta \subseteq \mathbb{R}^m$, several sub-classes of problems are obtained :

- **Model selection** : Θ is the set of canonical base vectors, or one-hot vectors.

$$\theta^T = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}$$

- **Convex aggregation** : Θ is the canonical simplex of $\mathbb{R}^m$,

$$\sum_{j=1}^{m} \theta_j = 1 \text{ and } \theta_j \geq 0.$$

- **Linear aggregation** : $\Theta = \mathbb{R}^m$.

- **Sparse linear aggregation** : $\Theta = \{\theta \in \mathbb{R}^m \text{ s.t. } \|\theta\|_0 < r\}$.

Statistical aggregation theory : problem statement

Depending the $\Theta \subseteq \mathbb{R}^m$, several sub-classes of problems are obtained :

- **Model selection** : Θ is the set of canonical base vectors, or one-hot vectors.

$$\theta^T = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}$$

- **Convex aggregation** : Θ is the canonical simplex of $\mathbb{R}^m$,

$$\sum_{j=1}^{m} \theta_j = 1 \text{ and } \theta_j \geq 0.$$

- **Linear aggregation** : $\Theta = \mathbb{R}^m$.

- **Sparse linear aggregation** : $\Theta = \{\theta \in \mathbb{R}^m \text{ s.t. } \|\theta\|_0 < r\}$.

Statistical aggregation theory : problem statement

Depending the $\Theta \subseteq \mathbb{R}^m$, several sub-classes of problems are obtained :

- **Model selection** : Θ is the set of canonical base vectors, or one-hot vectors.

$$\theta^T = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}$$

- **Convex aggregation** : Θ is the canonical simplex of $\mathbb{R}^m$,

$$\sum_{j=1}^{m} \theta_j = 1 \text{ and } \theta_j \geq 0.$$

- **Linear aggregation** : $\Theta = \mathbb{R}^m$.

- **Sparse linear aggregation** : $\Theta = \{\theta \in \mathbb{R}^m \text{ s.t. } \|\theta\|_0 < r\}$.

Statistical aggregation theory : problem statement

Let $R(f_\theta)$ denote the generalization error (risk) of the pre-estimator f_θ :

$$R(f_\theta) = \mathbb{E}_{X,Y}\left[L\left(f_\theta(\mathbf{x}), y\right)\right].$$

Definition

$\theta^\star$ is an **oracle** for the aggregation problem if

$$R(f_{\theta^\star}) = \inf_{\theta \in \Theta} R(f_\theta).$$

Statistical aggregation theory : problem statement

What are we looking for ?

- An aggregated estimate $\hat{\theta}$ close to the oracle $\theta^{\star}$.
- An aggregated estimate $\hat{\theta}$ learnt from private data $\mathcal{D}_{\text{val}}$.

Close in what sense ?

- In a the following probabilistic sense :

$$\mathbb{E}_{\mathcal{D}_{\text{val}}}\left[R\left(f_{\hat{\theta}}\right)\right] \leq C \inf_{\theta \in \Theta} \left\{R\left(f_{\theta}\right) + \Delta_{n,m}\left(\theta\right)\right\},$$

with constant $C \geq 1$ and rate $\Delta_{n,m} \longrightarrow 0$ as $n \longrightarrow \infty$.

Assymptotical warranty : when n is big, the risk is not worse (in average) than C times the oracle risk.

Statistical aggregation theory : problem statement

What are we looking for ?

- An aggregated estimate $\hat{\theta}$ close to the oracle $\theta^{\star}$.
- An aggregated estimate $\hat{\theta}$ learnt from private data $\mathcal{D}_{\text{val}}$.

Close in what sense ?

- In a the following probabilistic sense :

$$\mathbb{E}_{\mathcal{D}_{\text{val}}}\left[R\left(f_{\hat{\theta}}\right)\right] \leq C \inf_{\theta \in \Theta} \left\{R\left(f_{\theta}\right) + \Delta_{n,m}\left(\theta\right)\right\},$$

with constant $C \geq 1$ and rate $\Delta_{n,m} \longrightarrow 0$ as $n \longrightarrow \infty$.

Assymptotical warranty : when n is big, the risk is not worse (in average) than C times the oracle risk.

Statistical aggregation theory : problem statement

What are we looking for ?

- An aggregated estimate $\hat{\theta}$ close to the oracle $\theta^{\star}$.
- An aggregated estimate $\hat{\theta}$ learnt from private data $\mathcal{D}_{\text{val}}$.

Close in what sense ?

- In a the following probabilistic sense :

$$\mathbb{E}_{\mathcal{D}_{\text{val}}}\left[R\left(f_{\hat{\theta}}\right)\right] \leq C \inf_{\theta \in \Theta} \left\{R\left(f_{\theta}\right) + \Delta_{n,m}\left(\theta\right)\right\},$$

with constant $C \geq 1$ and rate $\Delta_{n,m} \longrightarrow 0$ as $n \longrightarrow \infty$.

Assymptotical warranty : when n is big, the risk is not worse (in average) than C times the oracle risk.

Statistical aggregation theory : problem statement

What are we looking for ?

- An aggregated estimate $\hat{\theta}$ close to the oracle $\theta^{\star}$.
- An aggregated estimate $\hat{\theta}$ learnt from private data $\mathcal{D}_{\text{val}}$.

Close in what sense ?

- In a the following probabilistic sense :

$$\mathbb{E}_{\mathcal{D}_{\text{val}}}\left[R\left(f_{\hat{\theta}}\right)\right] \leq C \inf_{\theta \in \Theta} \left\{R\left(f_{\theta}\right) + \Delta_{n,m}\left(\theta\right)\right\},$$

with constant $C \geq 1$ and rate $\Delta_{n,m} \longrightarrow 0$ as $n \longrightarrow \infty$.

Assymptotical warranty : when n is big, the risk is not worse (in average) than C times the oracle risk.

Statistical aggregation theory : problem analysis

- There is a bias-variance tradeoff binding approximation quality ($\Delta_{n,m}$) and the size of the parameter space Θ.

- In the $C = 1$ case, oracle inequalities are said to be sharp.

Statistical aggregation theory : problem analysis

- There is a bias-variance tradeoff binding approximation quality ($\Delta_{n,m}$) and the size of the parameter space Θ.

- In the $C = 1$ case, oracle inequalities are said to be sharp.

Statistical aggregation theory : problem analysis

- New question : for a given Θ, what is the best rate $\Delta_{n,m}$?

Definition

Let $\mathcal{F}$ denote a functional space.
Optimal rates or minimax rates are a sequence $\Phi_{n,m}$ such that

(i) For any pre-estimates $f_1, \ldots, f_m$ and any target f_0, there is an aggregate $f_{\tilde{\theta}}$ s.t.

$$R\left(f_{\tilde{\theta}}\right) - R\left(f_0\right) \leq c_1 \Phi_{n,m}$$

(ii) For any estimator $f_{\tilde{\theta}}$ trained on $\mathcal{D}_{\text{val}}$,

$$\sup_{f_0 \in \mathcal{F}} \left\{ R\left(f_{\tilde{\theta}}\right) - R\left(f_0\right) \right\} \geq c_2 \Phi_{n,m}.$$

c_1 and c_2 are positive constants. Note that we may have $f_{\tilde{\theta}} \notin \mathcal{H}_{\Theta}$.

Statistical aggregation theory : problem analysis

- New question : for a given Θ, what is the best rate $\Delta_{n,m}$?

Definition

Let $\mathcal{F}$ denote a functional space.
Optimal rates or minimax rates are a sequence $\Phi_{n,m}$ such that

(i) For any pre-estimates f_1, .., f_m and any target f_0, there is an aggregate $f_{\hat{\theta}}$ s.t.

$$R\left(f_{\hat{\theta}}\right) - R\left(f_0\right) \leq c_1 \Phi_{n,m}$$

(ii) For any estimator $f_{\tilde{\theta}}$ trained on $\mathcal{D}_{\text{val}}$,

$$\sup_{f_0 \in \mathcal{F}} \left\{ R\left(f_{\tilde{\theta}}\right) - R\left(f_0\right) \right\} \geq c_2 \Phi_{n,m}.$$

c_1 and c_2 are positive constants. Note that we may have $f_{\tilde{\theta}} \notin \mathcal{H}_{\Theta}$.

Statistical aggregation theory : problem analysis

- New question : for a given Θ, what is the best rate $\Delta_{n,m}$?

Definition

Let $\mathcal{F}$ denote a functional space.
Optimal rates or minimax rates are a sequence $\Phi_{n,m}$ such that

(i) For any pre-estimates f_1, .., f_m and any target f_0, there is an aggregate $f_{\hat{\theta}}$ s.t.

$$R\left(f_{\hat{\theta}}\right) - R\left(f_0\right) \leq c_1 \Phi_{n,m}$$

(ii) For any estimator $f_{\tilde{\theta}}$ trained on $\mathcal{D}_{\text{val}}$,

$$\sup_{f_0 \in \mathcal{F}} \left\{ R\left(f_{\tilde{\theta}}\right) - R\left(f_0\right) \right\} \geq c_2 \Phi_{n,m}.$$

c_1 and c_2 are positive constants. Note that we may have $f_{\tilde{\theta}} \notin \mathcal{H}_{\Theta}$.

Statistical aggregation theory : problem analysis

- New question : for a given Θ, what is the best rate $\Delta_{n,m}$?

Definition

Let $\mathcal{F}$ denote a functional space.
Optimal rates or minimax rates are a sequence $\Phi_{n,m}$ such that

(i) For any pre-estimates f_1, .., f_m and any target f_0, there is an aggregate $f_{\hat{\theta}}$ s.t.

$$R\left(f_{\hat{\theta}}\right) - R\left(f_0\right) \leq c_1 \Phi_{n,m}$$

(ii) For any estimator $f_{\tilde{\theta}}$ trained on $\mathcal{D}_{\text{val}}$,

$$\sup_{f_0 \in \mathcal{F}} \left\{ R\left(f_{\tilde{\theta}}\right) - R\left(f_0\right) \right\} \geq c_2 \Phi_{n,m}.$$

c_1 and c_2 are positive constants. Note that we may have $f_{\tilde{\theta}} \notin \mathcal{H}_\Theta$.

Statistical aggregation theory : problem analysis

Example : for linear aggregation of regressors (under additive Gaussian noise), we have

$$\Phi_{n,m} = \min\left\{1; \frac{m}{n}\right\}.$$

- The dimensionality of Θ inscreases w.r.t. m.
- The more data, the better the generalization.

Statistical aggregation theory : problem analysis

Example : for linear aggregation of regressors (under additive Gaussian noise), we have

$$\Phi_{n,m} = \min\left\{1; \frac{m}{n}\right\}.$$

- The dimensionality of Θ inscreases w.r.t. m.
- The more data, the better the generalization.

Statistical aggregation theory : some solutions

Penalized approaches :

- Since the distribution $p_{X,Y}$ is unknown, explicit risk computation is impossible.

- A biaised proxy for it is the empirical risk

$$R_{\text{emp}}(f_\theta) = \sum_{i=1}^{n} L\left(y^{(i)}, f_\theta\left(\mathbf{x}^{(i)}\right)\right),$$

$$= \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{val}}} L\left(y^{(i)}, \theta_1 f_1\left(\mathbf{x}^{(i)}\right) + \ldots + \theta_m f_m\left(\mathbf{x}^{(i)}\right)\right).$$

- Keep in mind that for each f_k, the training error is

$$Err_{\text{train}} = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{train}}} L\left(y^{(i)}, f_k\left(\mathbf{x}^{(i)}\right)\right).$$

Statistical aggregation theory : some solutions

Penalized approaches :

- Since the distribution $p_{X,Y}$ is unknown, explicit risk computation is impossible.

- A biaised proxy for it is the empirical risk

$$\begin{aligned} R_{\text{emp}}(f_\theta) &= \sum_{i=1}^{n} L\left(y^{(i)}, f_\theta\left(\mathbf{x}^{(i)}\right)\right), \\ &= \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{val}}} L\left(y^{(i)}, \theta_1 f_1\left(\mathbf{x}^{(i)}\right) + \ldots + \theta_m f_m\left(\mathbf{x}^{(i)}\right)\right). \end{aligned}$$

- Keep in mind that for each f_k, the training error is

$$Err_{\text{train}} = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{train}}} L\left(y^{(i)}, f_k\left(\mathbf{x}^{(i)}\right)\right).$$

Statistical aggregation theory : some solutions

Penalized approaches :

- Since the distribution $p_{X,Y}$ is unknown, explicit risk computation is impossible.

- A biaised proxy for it is the empirical risk

$$\begin{aligned} R_{\text{emp}}(f_\theta) &= \sum_{i=1}^{n} L\left(y^{(i)}, f_\theta\left(\mathbf{x}^{(i)}\right)\right), \\ &= \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{val}}} L\left(y^{(i)}, \theta_1 f_1\left(\mathbf{x}^{(i)}\right) + \ldots + \theta_m f_m\left(\mathbf{x}^{(i)}\right)\right). \end{aligned}$$

- Keep in mind that for each f_k, the training error is

$$Err_{\text{train}} = \sum_{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}_{\text{train}}} L\left(y^{(i)}, f_k\left(\mathbf{x}^{(i)}\right)\right).$$

Statistical aggregation theory : some solutions

Penalized approaches :

- A possibility to learn $\hat{\theta}$ is to minimize the empirical risk

$$\hat{\theta} = \arg\min_{\theta} R_{\text{emp}}(f_\theta).$$

- This is doomed to overfit, and therefore we solve

$$\hat{\theta} = \arg\min_{\theta} R_{\text{emp}}(f_\theta) + \text{Regularizer}(\theta).$$

- This penalized problem mimics the oracle bound, yet the regularizer does not depend on n !

- By carefully choosing $\text{Regularizer}(\theta)$ (with a $\|\theta\|_0$ term in it), then minimax rates are achieved for linear, convex and model selection aggregation problems.

Statistical aggregation theory : some solutions

Penalized approaches :

- A possibility to learn $\hat{\theta}$ is to minimize the empirical risk

$$\hat{\theta} = \arg\min_{\theta} R_{\text{emp}}\left(f_{\theta}\right).$$

- This is doomed to overfit, and therefore we solve

$$\hat{\theta} = \arg\min_{\theta} R_{\text{emp}}\left(f_{\theta}\right) + \text{Regularizer}\left(\theta\right).$$

- This penalized problem mimics the oracle bound, yet the regularizer does not depend on n !

- By carefully choosing $\text{Regularizer}\left(\theta\right)$ (with a $\|\theta\|_0$ term in it), then minimax rates are achieved for linear, convex and model selection aggregation problems.

Statistical aggregation theory : some solutions

Penalized approaches :

- A possibility to learn $\hat{\boldsymbol{\theta}}$ is to minimize the empirical risk

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} R_{\text{emp}}\left(f_{\boldsymbol{\theta}}\right).$$

- This is doomed to overfit, and therefore we solve

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} R_{\text{emp}}\left(f_{\boldsymbol{\theta}}\right) + \text{Regularizer}\left(\boldsymbol{\theta}\right).$$

- This penalized problem mimics the oracle bound, yet the regularizer does not depend on n !

- By carefully choosing $\text{Regularizer}\left(\boldsymbol{\theta}\right)$ (with a $\|\boldsymbol{\theta}\|_0$ term in it), then minimax rates are achieved for linear, convex and model selection aggregation problems.

Statistical aggregation theory : some solutions

Penalized approaches :

- A possibility to learn $\hat{\boldsymbol{\theta}}$ is to minimize the empirical risk

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} R_{\text{emp}}\left(f_{\boldsymbol{\theta}}\right).$$

- This is doomed to overfit, and therefore we solve

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} R_{\text{emp}}\left(f_{\boldsymbol{\theta}}\right) + \text{Regularizer}\left(\boldsymbol{\theta}\right).$$

- This penalized problem mimics the oracle bound, yet the regularizer does not depend on n !

- By carefully choosing $\text{Regularizer}\left(\boldsymbol{\theta}\right)$ (with a $\|\boldsymbol{\theta}\|_0$ term in it), then minimax rates are achieved for linear, convex and model selection aggregation problems.

Statistical aggregation theory : some solutions

Exponential weights :

- In the convex aggregation problem, weights $\left[\theta_1 \dots \theta_m\right]$ can be regarded as a probability distribution.

- By choosing a regularizer involving the Kullback-Leibler divergence between θ and a prior π and by replacing the empirical risk with the surrogate

$$\sum_{j=1}^{m} \theta_k R_{\text{emp}}\left(f_k\right),$$

we obtain

$$\hat{\theta} = \text{smax}\left(n \begin{bmatrix} \pi_1 R_{\text{emp}}\left(f_1\right) \\ \vdots \\ \pi_m R_{\text{emp}}\left(f_m\right) \end{bmatrix}\right).$$

Statistical aggregation theory : some solutions

Exponential weights :

- In the convex aggregation problem, weights $\left[\theta_1 \dots \theta_m\right]$ can be regarded as a probability distribution.

- By choosing a regularizer involving the Kullback-Leibler divergence between $\boldsymbol{\theta}$ and a prior $\boldsymbol{\pi}$ and by replacing the empirical risk with the surrogate

$$\sum_{j=1}^{m} \theta_k R_{\text{emp}}\left(f_k\right),$$

we obtain

$$\hat{\boldsymbol{\theta}} = \operatorname{smax}\left(n \begin{bmatrix} \pi_1 R_{\text{emp}}\left(f_1\right) \\ \vdots \\ \pi_m R_{\text{emp}}\left(f_m\right) \end{bmatrix}\right).$$

Statistical aggregation theory : our local expert

- Benjamin Guedj (INRIA / Painlevé - MODAL) homepage

- Internship position.

Statistical aggregation theory : our local expert

- Benjamin Guedj (INRIA / Painlevé - MODAL) homepage

- Internship position.