Data Science In Practice - Model selection

EPF - P2023
Yassine Idrissi - yassine.idrissi@teads.com
Ismail Chafai - ismail.chafai@teads.com

ÉCOLE D'INGÉNIEUR-E-S

Summary

1. Evaluating a Hypothesis: Metrics
2. Model Selection and Cross Validation
3. Bias vs Variance
4. Regularization
5. Hyperparameter tuning

Evaluating a Hypothesis:

Metrics

ÉCOLE D'INGÉNIEUR-E-S

Evaluating a Hypothesis: Metrics

- Small recap: in a Machine Learning problem

Evaluating a Hypothesis: Metrics

- Small recap: in a Machine Learning problem
  - We define a **hypothesis function**

Evaluating a Hypothesis: Metrics

- Small recap: in a Machine Learning problem
  - We define a **hypothesis function**
  - We define a **loss function**

Evaluating a Hypothesis: Metrics

- Small recap: in a Machine Learning problem
  - We define a **hypothesis function**
  - We define a **loss function**
  - We choose an **optimization algorithm** to find the best set of **parameters** (that minimize our loss)

Evaluating a Hypothesis: Metrics

- Small recap: in a Machine Learning problem
  - We define a **hypothesis function**
  - We define a **loss function**
    - We choose an **optimization algorithm** to find the best set of **parameters** (that minimize our loss)
- But how can we **evaluate/iterate** on our choices?

Evaluating a Hypothesis: Metrics

- Small recap: in a Machine Learning problem
  - We define a **hypothesis function**
  - We define a **loss function**
    - We choose an **optimization algorithm** to find the best set of **parameters** (that minimize our loss)
- But how can we **evaluate/iterate** on our choices?
- How can we tune the **hyperparameters**?

Evaluating a Hypothesis: Regression Metrics

- For **regression**, we can define the following **metrics**:

Evaluating a Hypothesis: Regression Metrics

- For **regression**, we can define the following **metrics**:
  - MSE: Mean Squared Error $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

Evaluating a Hypothesis: Regression Metrics

- For **regression**, we can define the following **metrics**:
  - MSE: Mean Squared Error $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$
  - RMSE: Root Mean Squared Error $\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$

Evaluating a Hypothesis: Regression Metrics

- For **regression**, we can define the following **metrics**:
  - MSE: Mean Squared Error $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$
  - RMSE: Root Mean Squared Error $\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$
  - MAE: Mean Absolute Error $\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$

Evaluating a Hypothesis: Regression Metrics

- For **regression**, we can define the following **metrics**:
  - MSE: Mean Squared Error $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$
  - RMSE: Root Mean Squared Error $\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$
  - MAE: Mean Absolute Error $\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$
  - R2: Coefficient of Determination $\mathrm{R2} = 1 - \frac{MSE(Model)}{MSE(Baseline)}$

Evaluating a Hypothesis: Classification Metrics

- For **classification**, we use very often the **confusion matrix**

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Evaluating a Hypothesis: Classification Metrics

- For **classification**, we use very often the **confusion matrix**
  - **True Positive**: Observation = True && Prediction = True
  - **False Positive**: Observation = False && Prediction = True
  - **True Negative**: Observation = False && Prediction = False
  - **False Negative**: Observation = True && Prediction = False

Evaluating a Hypothesis: Classification Metrics

- For **classification**, we use very often the **confusion matrix**
  - **True Positive**: Observation = True && Prediction = True
  - **False Positive**: Observation = False && Prediction = True
  - **True Negative**: Observation = False && Prediction = False
  - **False Negative**: Observation = True && Prediction = False
- In general, it makes more sense to talk about **ratios**:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad , \mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \quad , \mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}$$

Evaluating a Hypothesis: Classification Metrics

- We can also define:

Evaluating a Hypothesis: Classification Metrics

- We can also define:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

Evaluating a Hypothesis: Classification Metrics

- We can also define:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Number of correct positives}}{\text{Total number of predicted positives}}$$

Evaluating a Hypothesis: Classification Metrics

- We can also define:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Number of correct positives}}{\text{Total number of predicted positives}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Number of correct positives}}{\text{Total number of actual positives}}$$

Evaluating a Hypothesis: Classification Metrics

- We can also define:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Number of correct positives}}{\text{Total number of predicted positives}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Number of correct positives}}{\text{Total number of actual positives}}$$

$$\text{F}_1 = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} = \text{Harmonic mean of Precision and Recall}$$

Evaluating a Hypothesis: Classification Metrics

- We can also define:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Number of correct positives}}{\text{Total number of predicted positives}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Number of correct positives}}{\text{Total number of actual positives}}$$

$$\text{F}_1 = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} = \text{Harmonic mean of Precision and Recall}$$

- We should **carefully choose** our metrics depending on the data

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 1: We are constructing a cancer prediction model:

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 1: We are constructing a cancer prediction model:
  - We want to correctly classify all cancerous patients

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 1: We are constructing a cancer prediction model:
  - We want to correctly classify all cancerous patients
  - Which type of errors is most costly?

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 1: We are constructing a cancer prediction model:
  - We want to correctly classify all cancerous patients
  - Which type of errors is most costly?
  - Deciding a patient does have cancer while they don't have it OR Deciding they have cancer while they don't have it?

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 1: We are constructing a cancer prediction model:
  - We want to correctly classify all cancerous patients
  - Which type of errors is most costly?
  - Deciding a patient does have cancer while they don't have it OR Deciding they have cancer while they don't have it?
  - It's obvious: we'd rather have a **False Positive** than a **False Negative**

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 1: We are constructing a cancer prediction model:
  - We want to correctly classify all cancerous patients
  - Which type of errors is most costly?
  - Deciding a patient does have cancer while they don't have it OR Deciding they have cancer while they don't have it?
  - It's obvious: we'd rather have a **False Positive** than a **False Negative**
  - Consequence: We want to minimize the **False Negatives**

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 2: We are constructing a spam prediction model:

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 2: We are constructing a spam prediction model:
  - We want to correctly classify spams/non-spams

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 2: We are constructing a spam prediction model:
  - We want to correctly classify spams/non-spams
  - Which type of errors is most costly?

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 2: We are constructing a spam prediction model:
  - We want to correctly classify spams/non-spams
  - Which type of errors is most costly?
  - Deciding a mail is spam while it's not or deciding a mail is not a spam while it is ?

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 2: We are constructing a spam prediction model:
  - We want to correctly classify spams/non-spams
  - Which type of errors is most costly?
  - Deciding a mail is spam while it's not or deciding a mail is not a spam while it is ?
  - It's obvious: we'd rather have a False Negative than a False Positive

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 2: We are constructing a spam prediction model:
  - We want to correctly classify spams/non-spams
  - Which type of errors is most costly?
  - Deciding a mail is spam while it's not or deciding a mail is not a spam while it is ?
  - It's obvious: we'd rather have a False Negative than a False Positive
  - Consequence: We want to minimize the False Positives

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 3: We have a heavily **unbalanced** datasets (e.g: Fraud detection: only 0.1% of transactions are Fraud)

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 3: We have a heavily **unbalanced** datasets (e.g: Fraud detection: only 0.1% of transactions are Fraud)

  - It doesn't make sense to evaluate the **Accuracy** as even a dumb model that always predicts 0 will have an accuracy of 99.9%

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

Evaluating a Hypothesis: Metrics

- Depending on the **use case** and the **context** we need to choose what **metric** to maximize/minimize

- Example 3: We have a heavily **unbalanced** datasets (e.g: Fraud detection: only 0.1% of transactions are Fraud)

  - It doesn't make sense to evaluate the **Accuracy** as even a dumb model that always predicts 0 will have an accuracy of 99.9%

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{\text{Number of correct classifications}}{\text{Total}}$$

  - Instead, we should focus on the **Recall** as it evaluates how good is our model at detecting the positives

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Number of correct positives}}{\text{Total number of actual positives}}$$

Model Selection and Cross Validation

ÉCOLE D'INGÉNIEUR-E-S

Model Selection and Cross Validation

- We've seen on the previous course that we want to **minimize** the **loss function** on the **training** set

Model Selection and Cross Validation

- We've seen on the previous course that we want to **minimize** the **loss function** on the **training** set

- But, just because a hypothesis has **low training error**, that doesn't mean it is necessarily a **good hypothesis**

Model Selection and Cross Validation

- We've seen on the previous course that we want to **minimize** the **loss function** on the **training** set

- But, just because a hypothesis has **low training error**, that doesn't mean it is necessarily a **good hypothesis**

- We need to evaluate how good the model performs on **predicting unseen data points**. This is what we call the **generalization power** of the model

Model Selection and Cross Validation

- We've seen on the previous course that we want to **minimize** the **loss function** on the **training** set

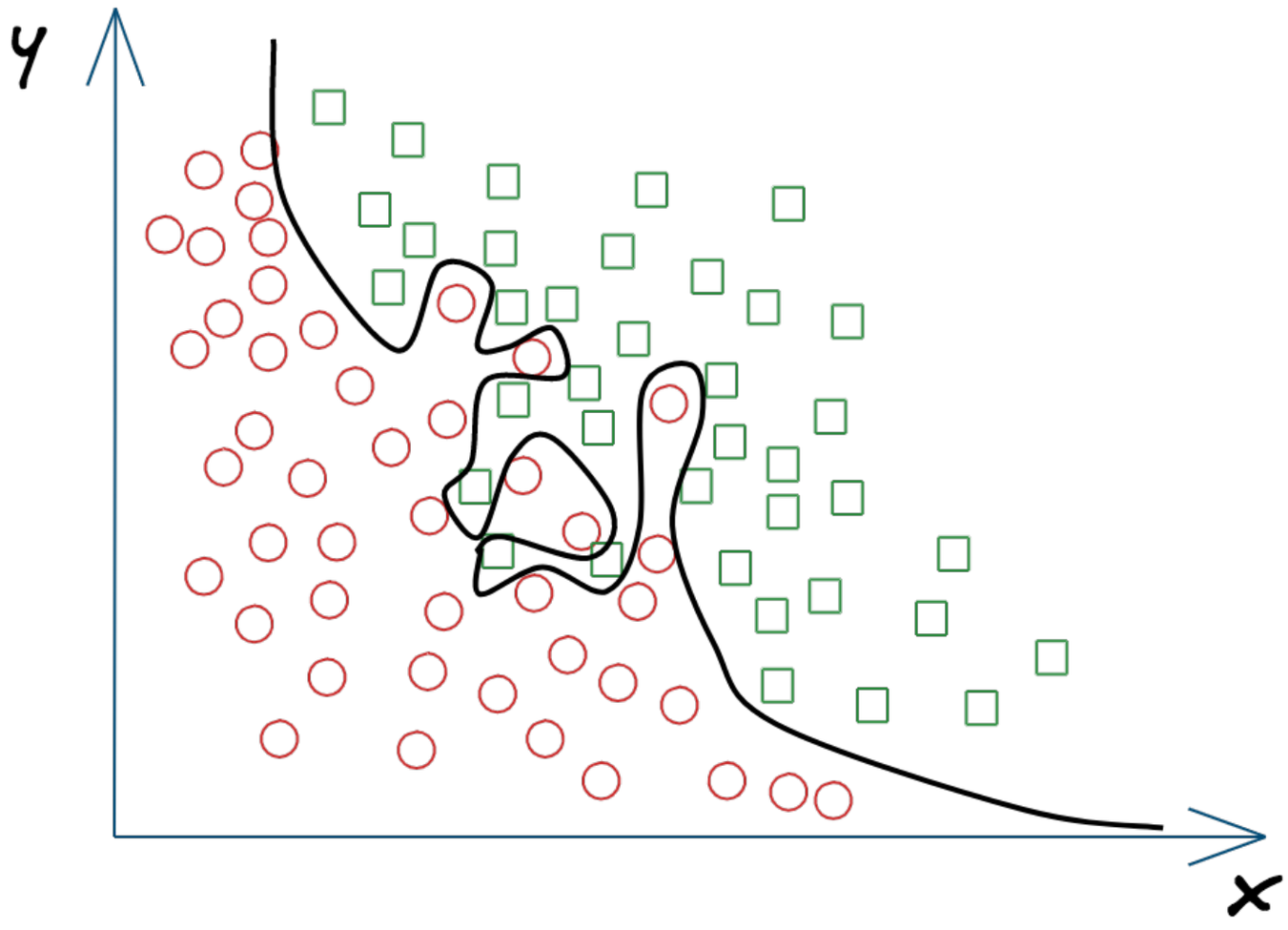
- But, just because a hypothesis has **low training error**, that doesn't mean it is necessarily a **good hypothesis**

- We need to evaluate how good the model performs on **predicting unseen data points**. This is what we call the **generalization power** of the model

- Let's see an example

Model Selection and Cross Validation

- Example: At first sight, the model seems to fit perfectly the training data

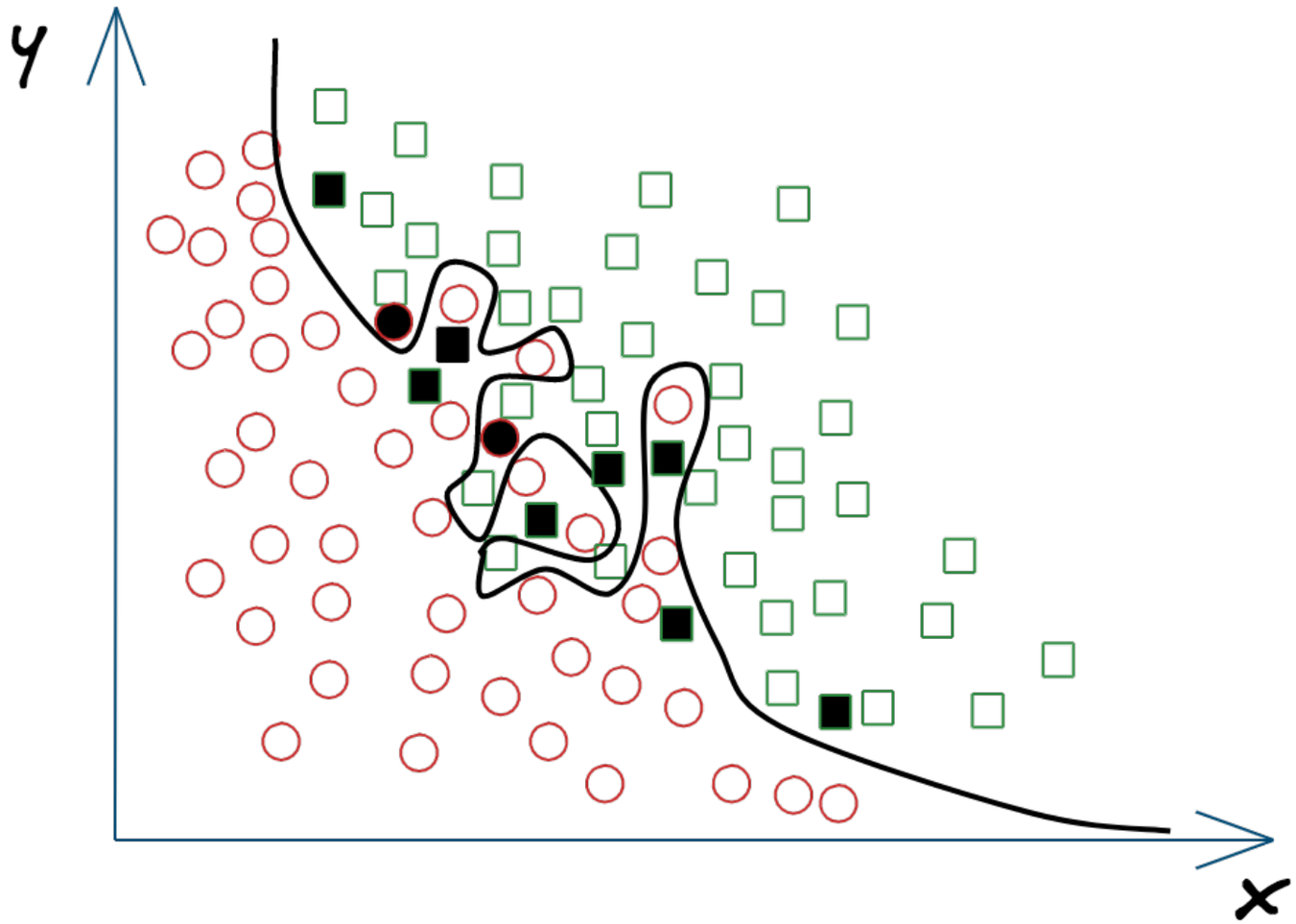


Model Selection and Cross Validation

- Example: How well will it perform on previously unseen data?

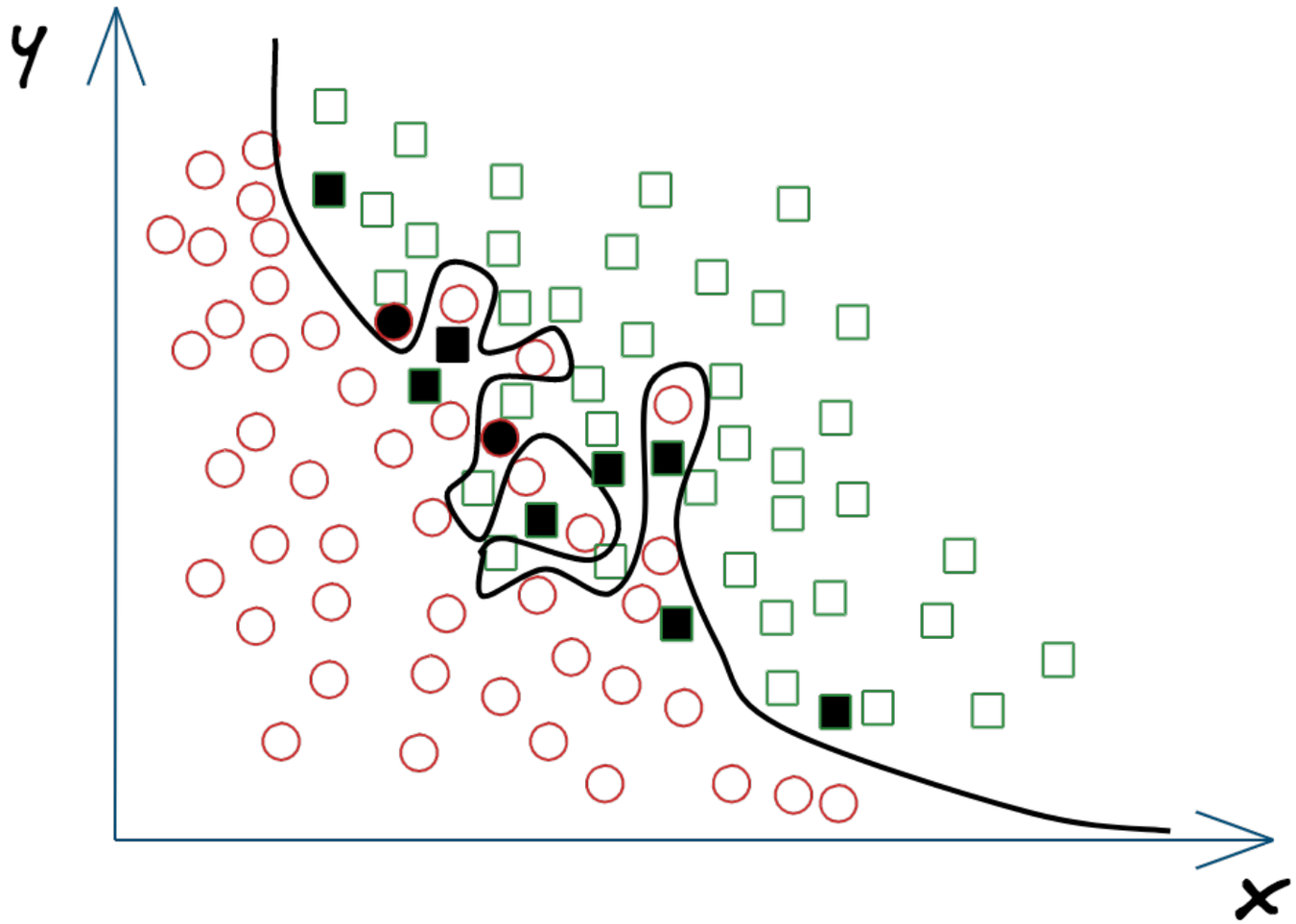
Model Selection and Cross Validation

- Example: How well will it perform on previously unseen data?



Model Selection and Cross Validation

- Example: How well will it perform on previously unseen data? Not so good! Why is that?



Model Selection and Cross Validation

- What happened?

Model Selection and Cross Validation

- What happened?

- The model **fits exactly** the training data but it **fails to predict on unseen data**

Model Selection and Cross Validation

- What happened?

- The model **fits exactly** the training data but it **fails to predict on unseen data**

- In other words: The model learned to **memorize** the training data rather than learning to **generalize** from it

Model Selection and Cross Validation

- What happened?

- The model **fits exactly** the training data but it **fails to predict on unseen data**

- In other words: The model learned to **memorize** the training data rather than learning to **generalize** from it

- This is called **overfitting**

Model Selection and Cross Validation

- What happened?

- The model **fits exactly** the training data but it **fails to predict on unseen data**

- In other words: The model learned to **memorize** the training data rather than learning to **generalize** from it

- This is called **overfitting**

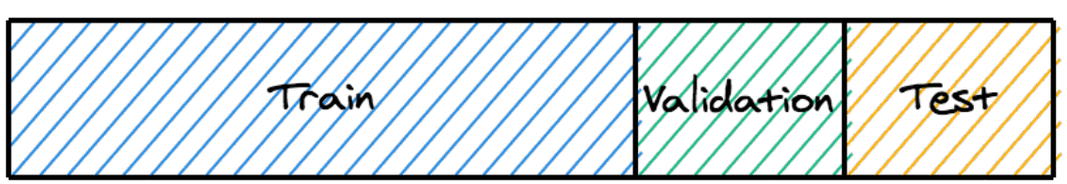
- This is one of the reasons we need to evaluate our models on **unseen data** (i.e test data)

Model Selection and Cross Validation

- Let's say we want to choose between these hypothesis functions

$$h_\theta^{(1)}(x) = \theta_0 + \theta_1 x$$
$$h_\theta^{(2)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$
$$h_\theta^{(3)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$
$$\dots$$
$$h_\theta^{(6)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6$$

- We need to split our data to 3 parts:
  - Train, Validation, Test (usually: 60%, 20%, 20%)



Model Selection and Cross Validation

- For each of the six hypothesis functions:
  - We train the models on the **training set** to find the best set of **parameters**
  - We then evaluate the models on the **validation** set to choose the best one
  - Finally, we evaluate the **generalization** error on the **test** set

Model Selection and Cross Validation

- For each of the six hypothesis functions:
  - We train the models on the **training set** to find the best set of **parameters**
  - We then evaluate the models on the **validation** set to choose the best one
  - Finally, we evaluate the **generalization** error on the **test** set
- Keep in mind that we have 2 goals here:
  - **Model Selection**: Choose the best model among different models
  - **Model Assessment:** Once the model chosen, assess its generalization error on new data

Model Selection and Cross Validation

- For each of the six hypothesis functions:
  - We train the models on the **training set** to find the best set of **parameters**
  - We then evaluate the models on the **validation** set to choose the best one
  - Finally, we evaluate the **generalization** error on the **test** set
- Keep in mind that we have 2 goals here:
  - **Model Selection**: Choose the best model among different models
  - **Model Assessment:** Once the model chosen, assess its generalization error on new data
- The **generalization** error needs to be evaluated on data that **hasn't been seen** by the model and that didn't serve for parameters tuning

Model Selection and Cross Validation

- In the **holdout** method seen earlier, the evaluation **depends heavily** on which data points end up in which split (train, test, validation)

Model Selection and Cross Validation

- In the **holdout** method seen earlier, the evaluation **depends heavily** on which data points end up in which split (train, test, validation)

- One way to improve the **reliability** of our evaluation is to use the **K-fold cross validation**

Model Selection and Cross Validation

- In the **holdout** method seen earlier, the evaluation **depends heavily** on which data points end up in which split (train, test, validation)

- One way to improve the **reliability** of our evaluation is to use the **K-fold cross validation**

- The data set is divided into k subsets. Each time one of the k subsets is used as a **validation** set and the k-1 remaining are put together to form the **training** set

Model Selection and Cross Validation

- In the **holdout** method seen earlier, the evaluation **depends heavily** on which data points end up in which split (train, test, validation)

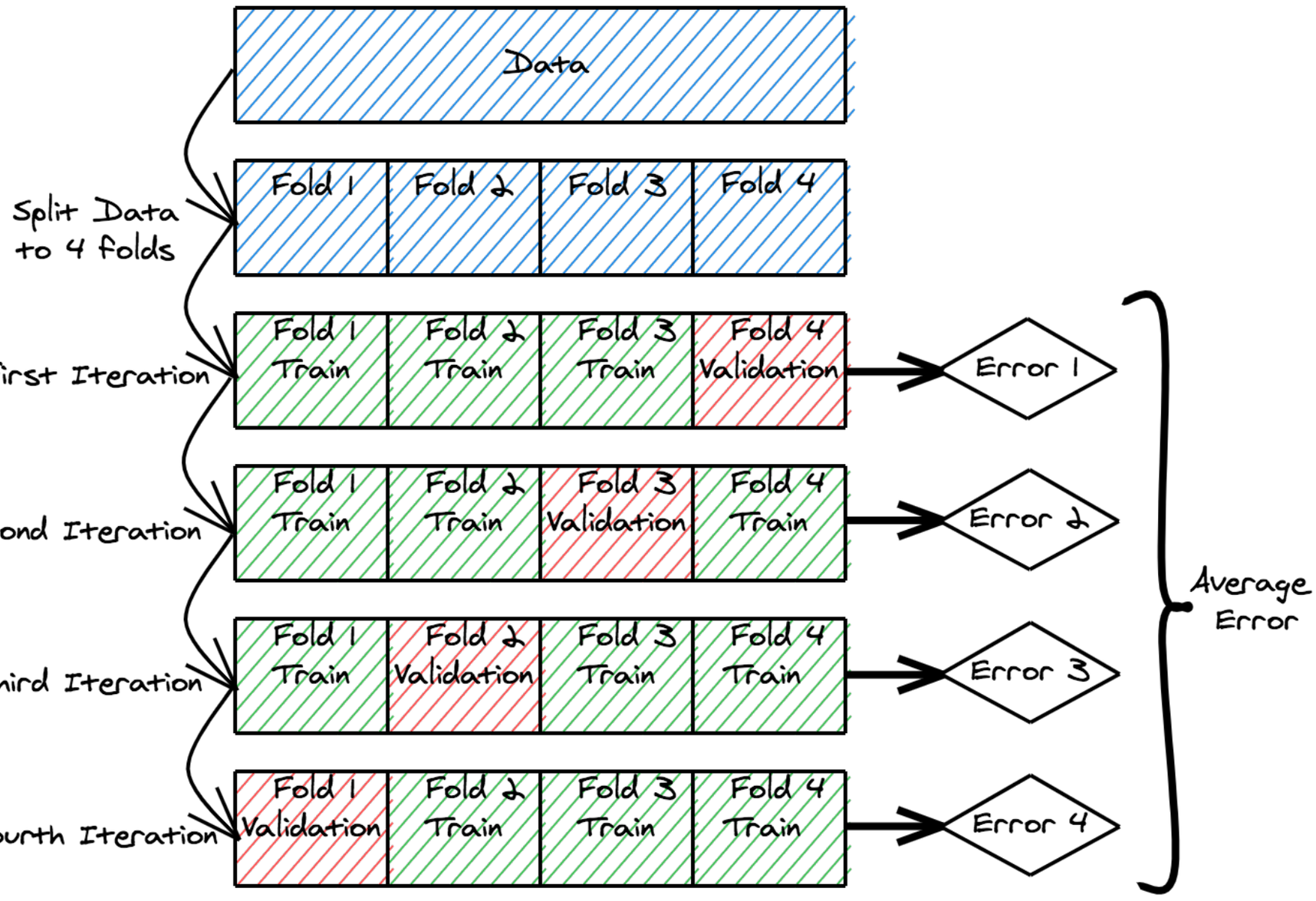
- One way to improve the **reliability** of our evaluation is to use the **K-fold cross validation**

- The data set is divided into k subsets. Each time one of the k subsets is used as a **validation** set and the k-1 remaining are put together to form the **training** set

- The method is **repeated** k times, in the end we compute the **average** error across the k trials

Model Selection and Cross Validation

- Example: 4-fold cross-validation



Bias Vs Variance

Bias Vs Variance

- Let's assume we have a function $Y = f(X) + \epsilon$ where $E(\epsilon) = 0, Var(\epsilon) = \sigma_\epsilon^2$ and we want to fit a model of it with the **hypothesis** $\hat{f}(X)$

Bias Vs Variance

- Let's assume we have a function $Y = f(X) + \epsilon$ where $E(\epsilon) = 0, Var(\epsilon) = \sigma_\epsilon^2$ and we want to fit a model of it with the **hypothesis** $\hat{f}(X)$

- We can show that the **squared error loss** can be **decomposed**:

$$\text{Err} = \text{Irreductible Error} + \text{Bias}^2 + \text{Variance}$$

Bias Vs Variance

- Let's assume we have a function $Y = f(X) + \epsilon$ where $E(\epsilon) = 0, Var(\epsilon) = \sigma_\epsilon^2$ and we want to fit a model of it with the **hypothesis** $\hat{f}(X)$

- We can show that the **squared error loss** can be **decomposed**:

$$\text{Err} = \text{Irreductible Error} + \text{Bias}^2 + \text{Variance}$$

- The **Bias** represents how well the model **captures** the data, and the **Variance** how much the model will **move** around its mean

Bias Vs Variance

- Let's assume we have a function $Y = f(X) + \epsilon$ where $E(\epsilon) = 0, Var(\epsilon) = \sigma_\epsilon^2$ and we want to fit a model of it with the **hypothesis** $\hat{f}(X)$

- We can show that the **squared error loss** can be **decomposed**:

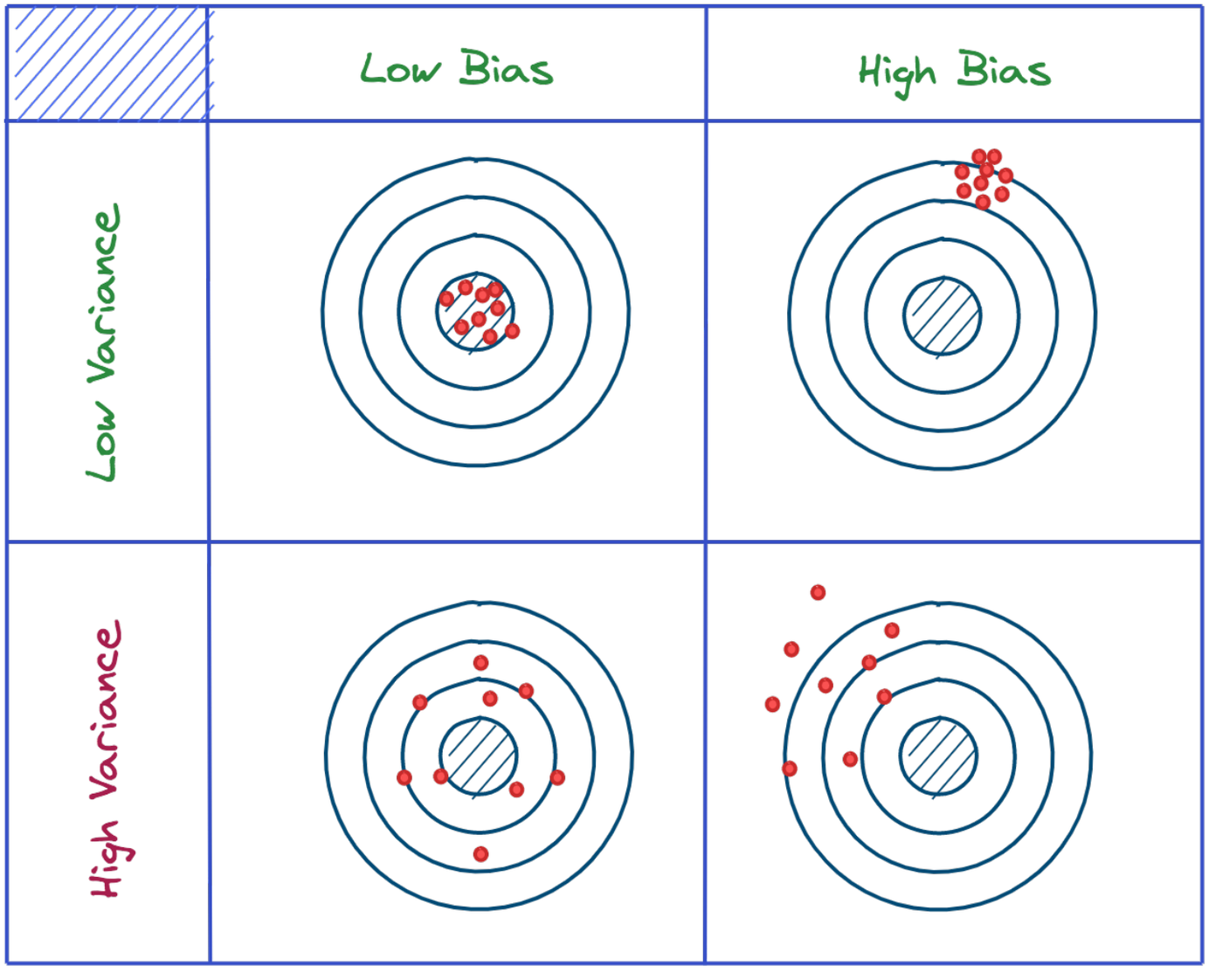
$$\text{Err} = \text{Irreductible Error} + \text{Bias}^2 + \text{Variance}$$

- The **Bias** represents how well the model **captures** the data, and the **Variance** how much the model will **move** around its mean

- Typically: The **more complex** the model is, the **lower the Bias**, but **the higher the Variance**. Which one should we choose?

Bias Vs Variance

- A good visualization tool for this is the bulls-eye diagram



Bias Vs Variance

- Let's take back the polynomial regression example

$$h_\theta^{(1)}(x) = \theta_0 + \theta_1 x$$
$$h_\theta^{(2)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$
$$h_\theta^{(3)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$
$$\dots$$
$$h_\theta^{(6)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6$$

Bias Vs Variance

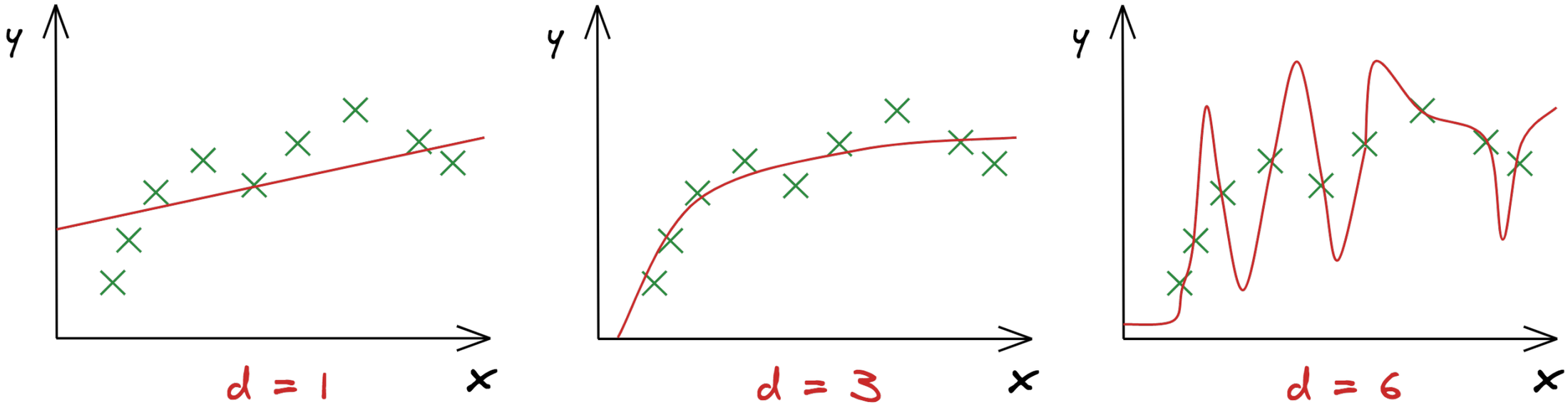
- Let's take back the polynomial regression example

$$h_\theta^{(1)}(x) = \theta_0 + \theta_1 x$$
$$h_\theta^{(2)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$
$$h_\theta^{(3)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$
$$\dots$$
$$h_\theta^{(6)}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6$$

- Here the **complexity** of the model is represented by the **degree** of the polynomial (d). The higher the degree, the higher the complexity

Bias Vs Variance

- We need to carefully choose d



Small d
High Bias (underfit)

Intermediate d
Equilibrium

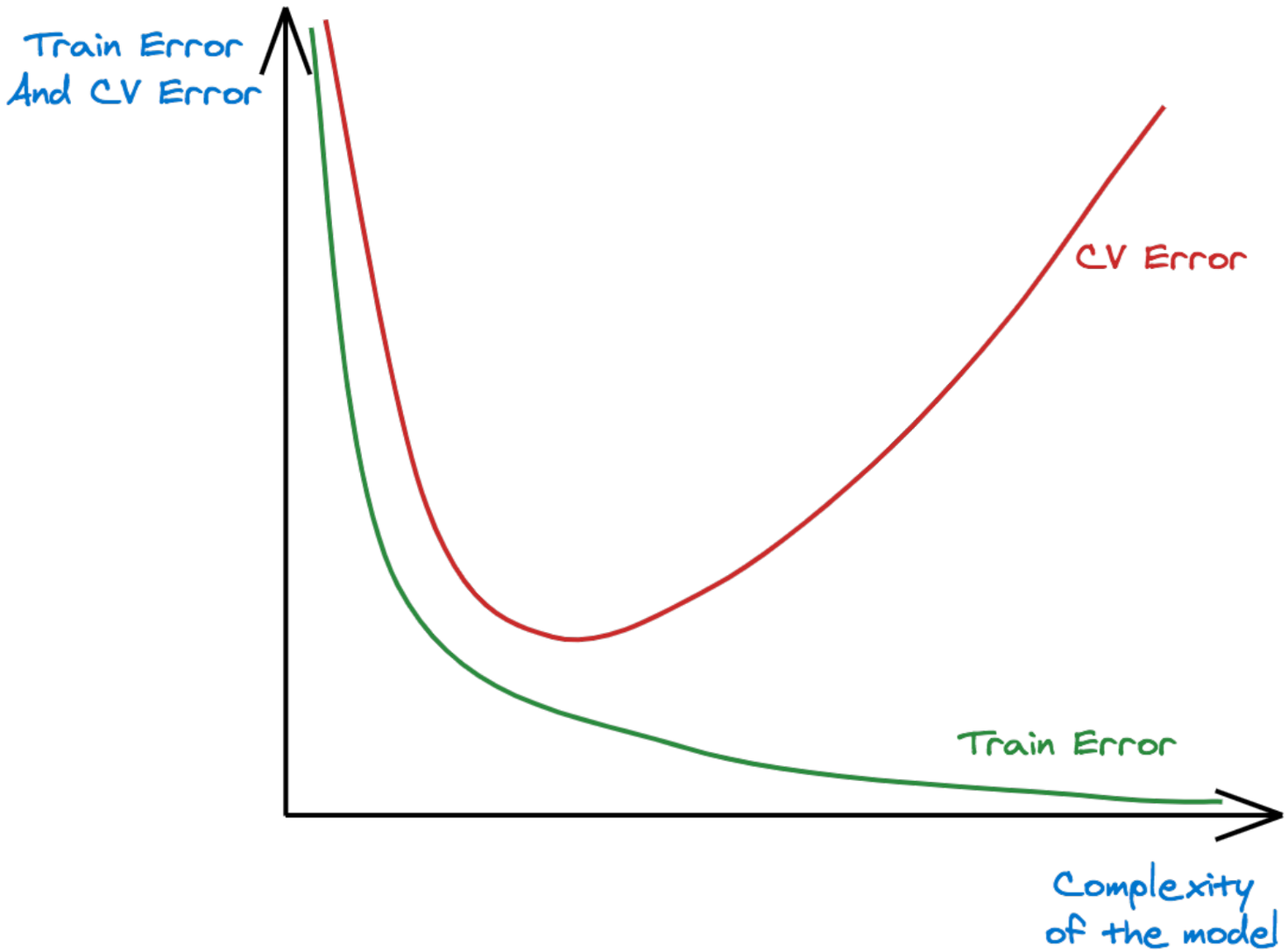
High d
High Variance (overfit)

Bias Vs Variance

- Let’s see how it translates over to the cross-validation error

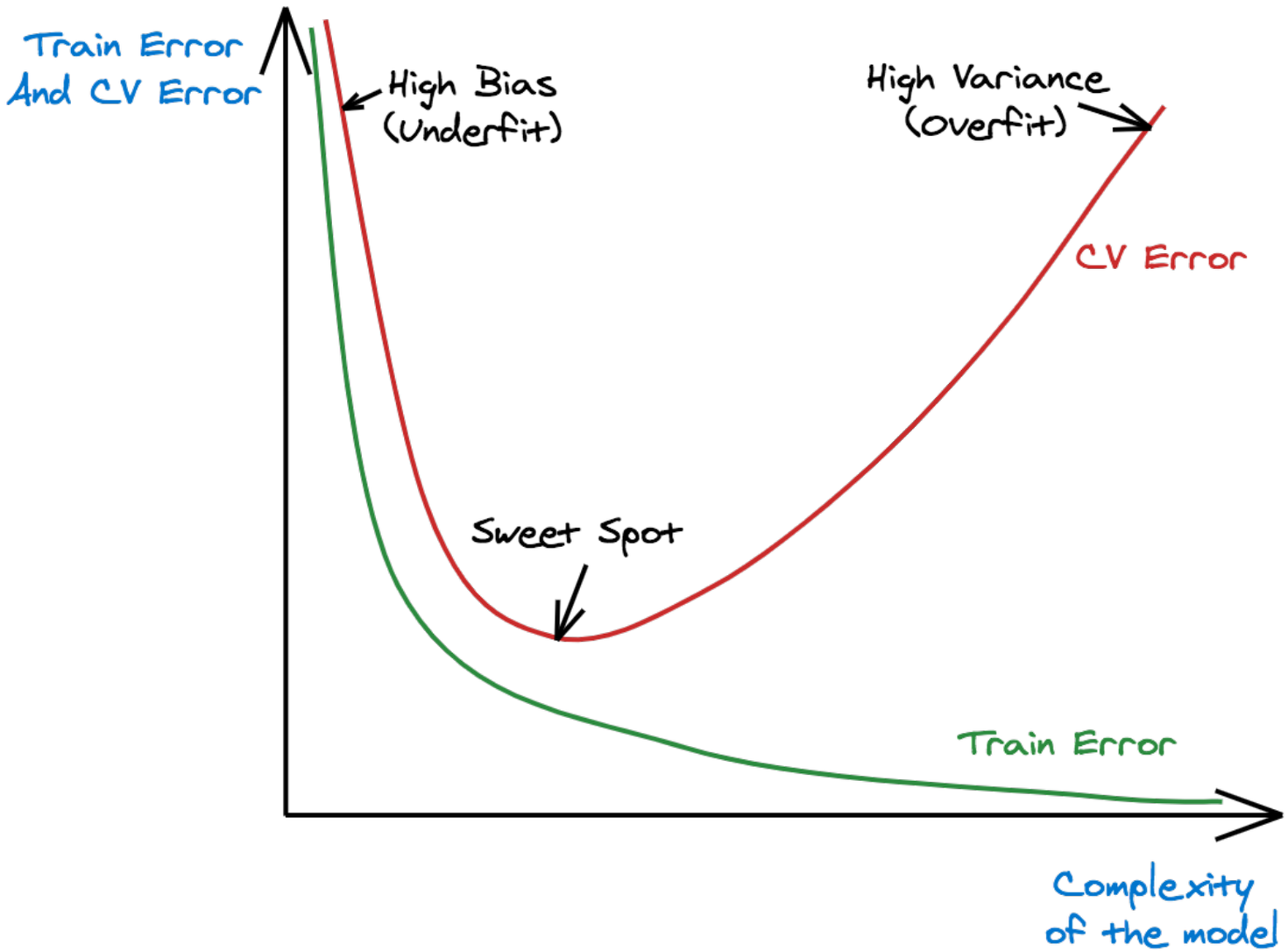
Bias Vs Variance

- Let's see how it translates over to the cross-validation error



Bias Vs Variance

- Let's see how it translates over to the cross-validation error



Bias Vs Variance

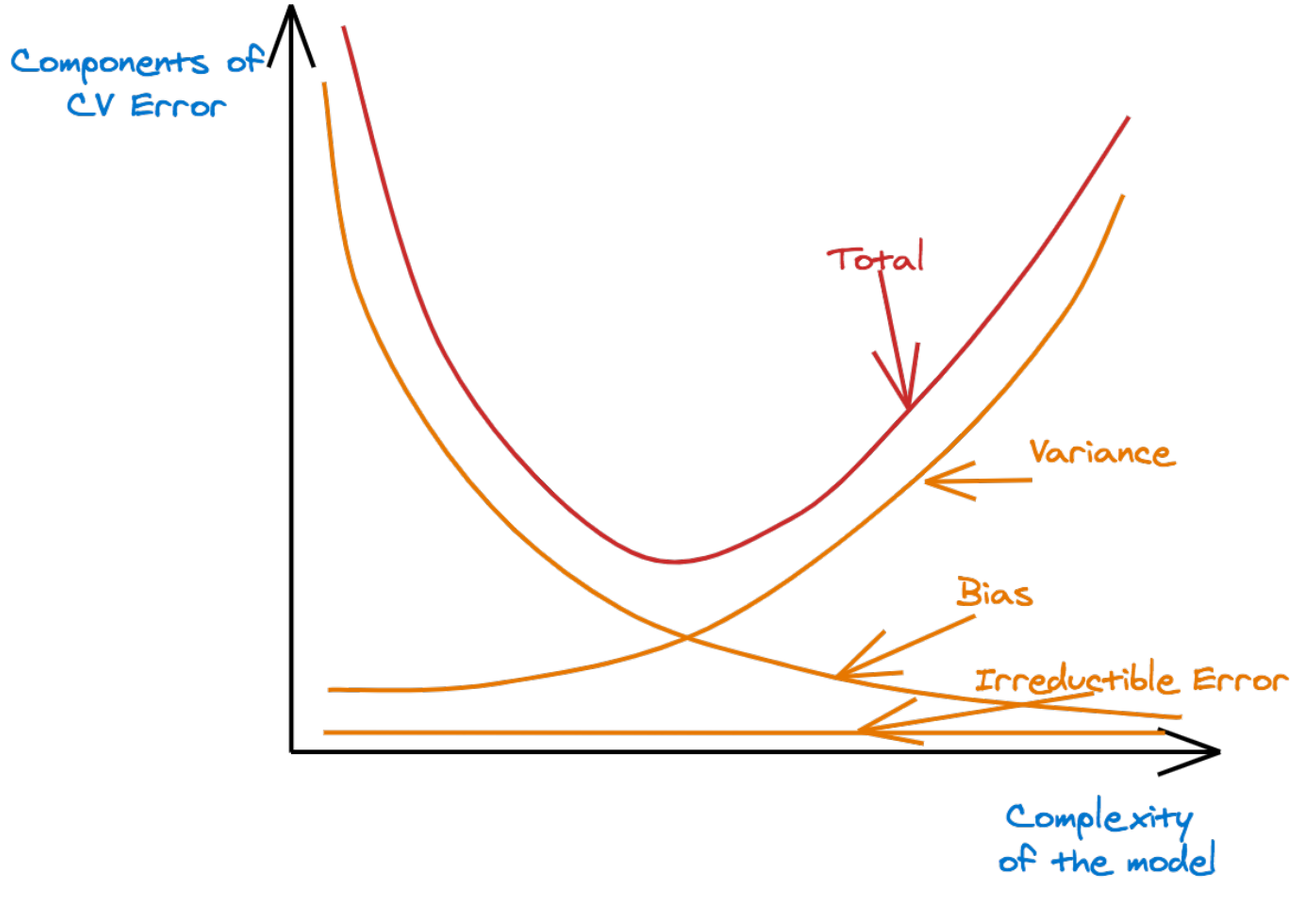
- Why does the CV Error have this **bowl-shaped** curve?

Bias Vs Variance

- Why does the CV Error have this **bowl-shaped** curve?
- Recall that: $\text{Err} = \text{Irreductible Error} + \text{Bias}^2 + \text{Variance}$

Bias Vs Variance

- Why does the CV Error have this **bowl-shaped** curve?
- Recall that: $\text{Err} = \text{Irreductible Error} + \text{Bias}^2 + \text{Variance}$



Regularization

Regularization

- We have seen that as the **complexity** of a model increases, so does it's **variance**

Regularization

- We have seen that as the **complexity** of a model increases, so does it's **variance**

- How can we deal with the **complexity** of our models?

Regularization

- We have seen that as the **complexity** of a model increases, so does it's **variance**

- How can we deal with the **complexity** of our models?

- We should **penalize** the learning of **complex models** and **encourage** the learning of **simpler models**

Regularization

- We have seen that as the **complexity** of a model increases, so does it's **variance**

- How can we deal with the **complexity** of our models?

- We should **penalize** the learning of **complex models** and **encourage** the learning of **simpler models**

- This can be done by adding a **regularization term** to the loss function we want to minimize $\text{Loss}_{regularized} = \text{Loss} + \lambda \text{ Regularizaion}$

Regularization

- We have seen that as the **complexity** of a model increases, so does it's **variance**

- How can we deal with the **complexity** of our models?

- We should **penalize** the learning of **complex models** and **encourage** the learning of **simpler models**

- This can be done by adding a **regularization term** to the loss function we want to minimize $\text{Loss}_{regularized} = \text{Loss} + \lambda \text{ Regularizaion}$

- The principle of **Regularization** is to accept a slight increase of Bias to get a bigger decrease of Variance => **The overall error is reduced**

Regularization

- For linear regression, recall how we defined our **loss function**:

$$J_\theta(x) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Regularization

- For linear regression, recall how we defined our **loss function**:

$$J_\theta(x) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- We add a **regularization** term to obtain the **regularized loss**:

$$J_\theta^{reg} = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}))^2 + \lambda \mathrm{Reg}(\theta)$$

Regularization

- For linear regression, recall how we defined our **loss function**:

$$J_\theta(x) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- We add a **regularization** term to obtain the **regularized loss**:

$$J_\theta^{reg} = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}))^2 + \lambda \mathrm{Reg}(\theta)$$

Ridge Regularization

$$\mathrm{Reg}(\theta) = \sum_{j=1}^{p}\theta_j^2$$

Regularization

- For linear regression, recall how we defined our **loss function**:

$$J_\theta(x) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- We add a **regularization** term to obtain the **regularized loss**:

$$J_\theta^{reg} = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}))^2 + \lambda \mathrm{Reg}(\theta)$$

Ridge Regularization

$$\mathrm{Reg}(\theta) = \sum_{j=1}^{p}\theta_j^2$$

Lasso Regularization

$$\mathrm{Reg}(\theta) = \sum_{j=1}^{p}|\theta_j|$$

Regularization

- What's the effect of **regularization** on a linear regression model?

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Ridge, we can show that

$$\min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}\theta_j^2\right)$$

$$\Leftrightarrow \min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\right) \text{ s.t} \sum_{j=1}^{p}\theta_j^2 \leq \tau$$

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Ridge, we can show that

$$\min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}\theta_j^2\right)$$

$$\Leftrightarrow \min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\right) \text{ s.t} \sum_{j=1}^{p}\theta_j^2 \leq \tau$$

- Ridge Regularization **shrinks** the parameters of the model, preventing them from getting **too high**

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Ridge, we can show that

$$\min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}\theta_j^2\right)$$

$$\Leftrightarrow \min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\right) \text{ s.t} \sum_{j=1}^{p}\theta_j^2 \leq \tau$$

- Ridge Regularization **shrinks** the parameters of the model, preventing them from getting **too high**
- The coefficients of the least important features will be **close** to 0

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Ridge, we can show that

$$\min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}\theta_j^2\right)$$

$$\Leftrightarrow \min_\theta\left(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\right) \text{ s.t} \sum_{j=1}^{p}\theta_j^2 \leq \tau$$

- Ridge Regularization **shrinks** the parameters of the model, preventing them from getting **too high**
- The coefficients of the least important features will be **close** to 0
- This leads to a **lower model complexity** (variance) and helps prevent **overfitting**

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Lasso, we can show that $\min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}|\theta_j|)$

$$\Leftrightarrow \min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2) \text{ s.t} \sum_{j=1}^{p}|\theta_j| \leq \tau$$

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Lasso, we can show that $\min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}|\theta_j|)$

$$\Leftrightarrow \min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2) \text{ s.t} \sum_{j=1}^{p}|\theta_j| \leq \tau$$

- Lasso Regularization **sets the weights of least import features to 0**

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Lasso, we can show that $\min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}|\theta_j|)$

$$\Leftrightarrow \min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2) \text{ s.t} \sum_{j=1}^{p}|\theta_j| \leq \tau$$

- Lasso Regularization **sets the weights of least import features to 0**
- It serves as both a **shrinkage** method and **feature selection** method

Regularization

- What's the effect of **regularization** on a linear regression model?
- For Lasso, we can show that $\min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{p}|\theta_j|)$

$$\Leftrightarrow \min_\theta(\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2) \text{ s.t} \sum_{j=1}^{p}|\theta_j| \leq \tau$$

- Lasso Regularization **sets the weights of least import features to 0**
- It serves as both a **shrinkage** method and **feature selection** method
- By only keeping the most important, the model is **less complex** and **more explainable**

Regularization

	Ridge	Lasso
Advantages	- Helps shrink parameters of variables that are less important - Handles well high number of parameters	- Serves as a shrinkage method - Serves as a feature selection method as unimportant features are set to 0
Inconvenients	- It helps shrink the parameters but does not set them to 0 - This leads to poor model interpretability	- It struggles when there are too many predictors - If there are two or more highly collinear variables, it will randomly select one and set the rest to 0

Regularization

- We defined our **regularized loss function** as

$$J_{reg}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{p}\theta_p^2$$

Regularization

- We defined our **regularized loss function** as

$$J_{reg}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{p}\theta_p^2$$

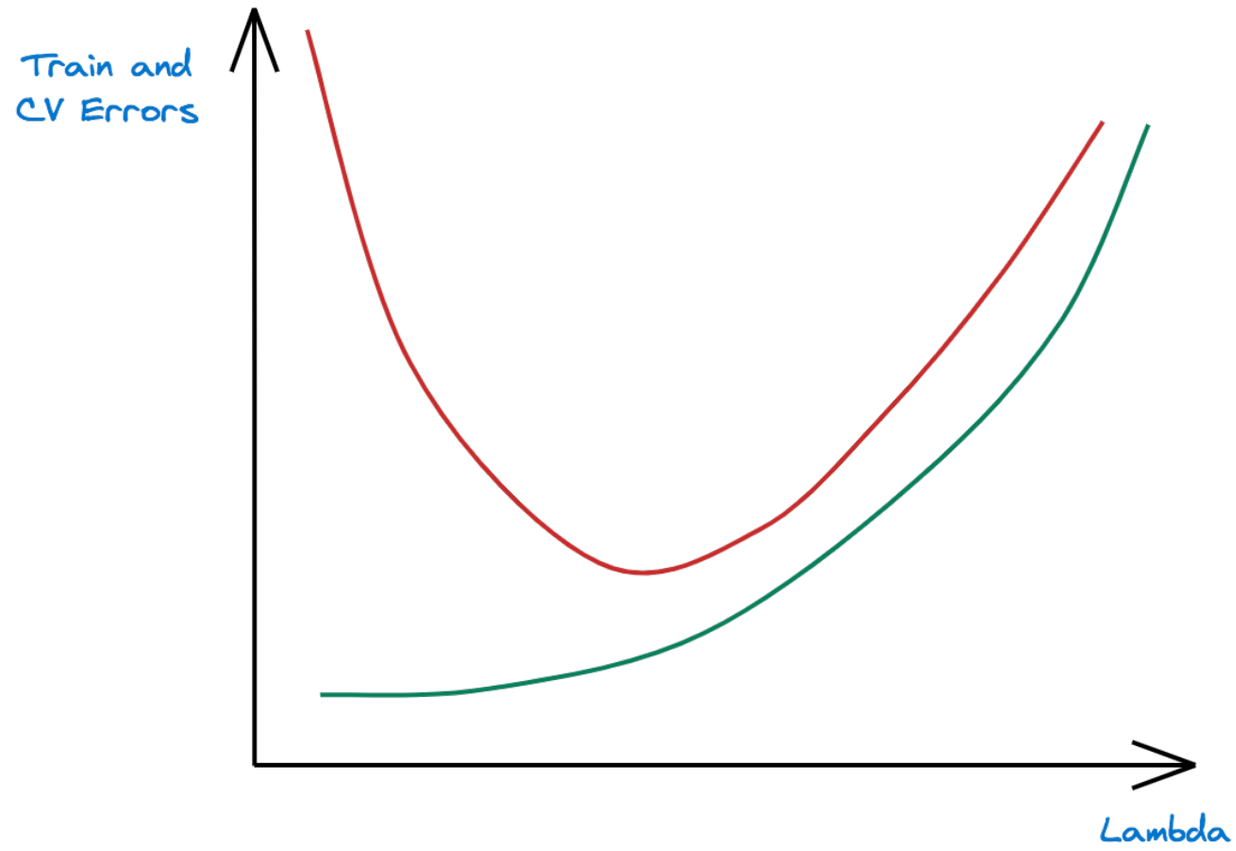
- Let's vary the parameter λ and evaluate our non-regularized loss function on the train and cross-validation sets

$$J_{train}(\theta) = \frac{1}{2m_{train}}\sum_{i=1}^{m_{train}}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x^{(i)}) - y^{(i)})^2$$

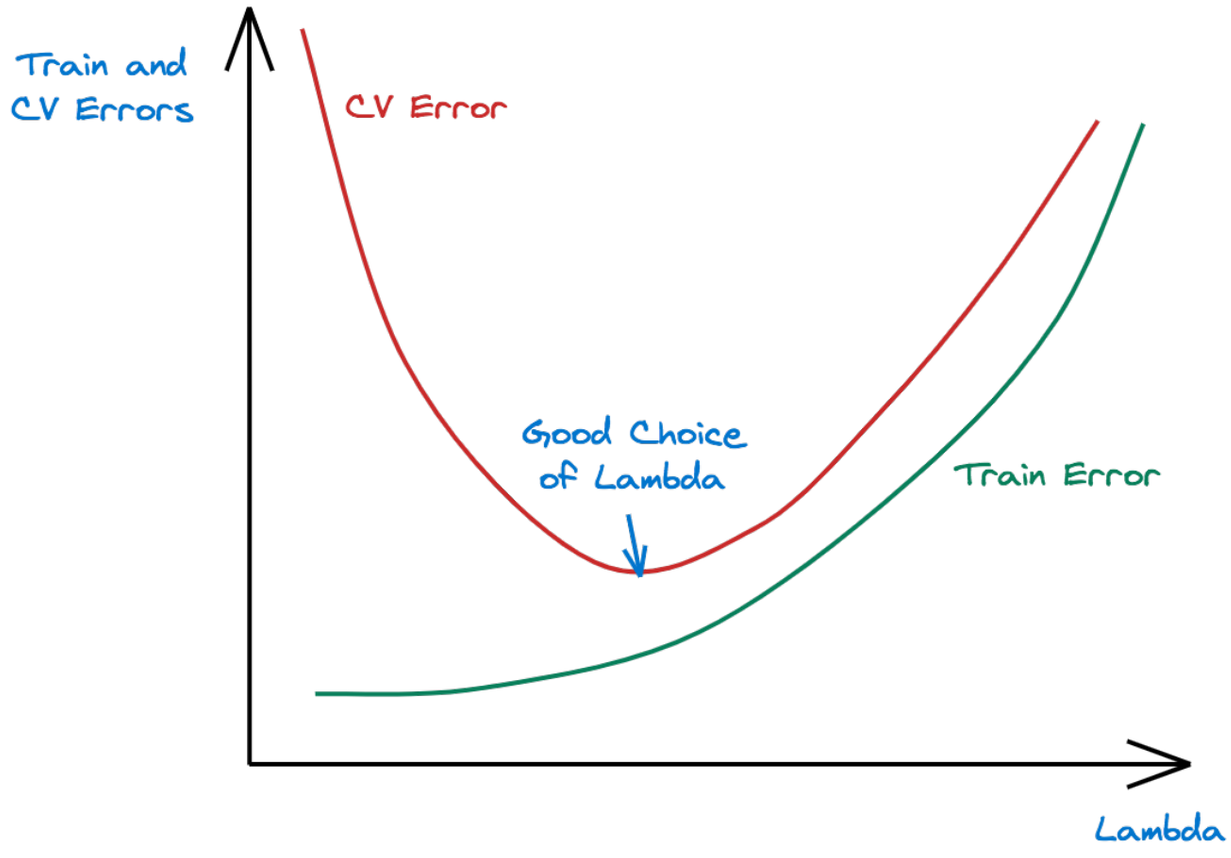
Regularization

- Which one is which? (as λ increases, what happens to train and cv errors ?)



Regularization

- As λ increases, the complexity (variance) of the model decreases, the cross validation error decreases, until a point where it starts increasing again. At some point, the decrease of variance is not enough to compensate for the bias increase



Regularization for Random Forests

- Recall that for Decision Trees, we defined some parameters

Regularization for Random Forests

- Recall that for Decision Trees, we defined some parameters
  - **Minimum Leaf Size**: Do not split *R* if its cardinality falls below a threshold

Regularization for Random Forests

- Recall that for Decision Trees, we defined some parameters
  - **Minimum Leaf Size**: Do not split *R* if its cardinality falls below a threshold
  - **Maximum Depth**: Do not split *R* if more than a fixed threshold of splits were already taken to reach *R*

Regularization for Random Forests

- Recall that for Decision Trees, we defined some parameters
  - **Minimum Leaf Size**: Do not split R if its cardinality falls below a threshold
  - **Maximum Depth**: Do not split R if more than a fixed threshold of splits were already taken to reach R
  - **Maximum Number of Nodes**: Stop if a tree has more leaf nodes than a fixed threshold

Regularization for Random Forests

- We have also defined some parameters for Random Forests

Regularization for Random Forests

- We have also defined some parameters for Random Forests
  - **M**: The max number of features considered for a tree

Regularization for Random Forests

- We have also defined some parameters for Random Forests
  - **M**: The max number of features considered for a tree
  - **N**: The max number of trees in the forest

Regularization for Random Forests

- We have also defined some parameters for Random Forests
  - **M**: The max number of features considered for a tree
  - **N**: The max number of trees in the forest
  - **Min Sample Leaf size**: The minimum number of data points in a leaf

Regularization for Random Forests

- We have also defined some parameters for Random Forests
  - **M**: The max number of features considered for a tree
  - **N**: The max number of trees in the forest
  - **Min Sample Leaf size**: The minimum number of data points in a leaf
  - etc…

Regularization for Random Forests

- We have also defined some parameters for Random Forests
  - **M**: The max number of features considered for a tree
  - **N**: The max number of trees in the forest
  - **Min Sample Leaf size**: The minimum number of data points in a leaf
  - etc…
- All these features help us **control the complexity** of our Decision Trees or Random Forest model. The more **complex** our model is, the more likely it is to be **overfitting**

Hyperparameters Tuning

EPF
ÉCOLE D'INGÉNIEUR-E-S

Hyperparameters Tuning

- We’ve seen earlier that to **improve the performances** of our model, we need to make some **design choices**

Hyperparameters Tuning

- We’ve seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

Hyperparameters Tuning

- We've seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

  - What degree of polynomial features should I use for my linear model?

Hyperparameters Tuning

- We've seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

  - What degree of polynomial features should I use for my linear model?

  - What should be the maximum depth allowed for my decision tree?

Hyperparameters Tuning

- We've seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

  - What degree of polynomial features should I use for my linear model?

  - What should be the maximum depth allowed for my decision tree?

  - What should be the minimum number of samples required at a leaf node in my decision tree?

Hyperparameters Tuning

- We've seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

  - What degree of polynomial features should I use for my linear model?
  - What should be the maximum depth allowed for my decision tree?
  - What should be the minimum number of samples required at a leaf node in my decision tree?
  - What should I set my learning rate to for gradient descent?

Hyperparameters Tuning

- We've seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

  - What degree of polynomial features should I use for my linear model?
  - What should be the maximum depth allowed for my decision tree?
  - What should be the minimum number of samples required at a leaf node in my decision tree?
  - What should I set my learning rate to for gradient descent?
  - How many trees should I include in my random forest?

Hyperparameters Tuning

- We've seen earlier that to **improve the performances** of our model, we need to make some **design choices**

- For example:

  - What degree of polynomial features should I use for my linear model?
  - What should be the maximum depth allowed for my decision tree?
  - What should be the minimum number of samples required at a leaf node in my decision tree?
  - What should I set my learning rate to for gradient descent?
  - How many trees should I include in my random forest?
  - …?

Hyperparameters Tuning

- Model **Parameters** and **Hyperparameters** are 2 **separate** things!

Hyperparameters Tuning

- Model **Parameters** and **Hyperparameters** are 2 **separate** things!

Parameters

- Are learned during the training process
- Are the result of the minimization of a given loss function with the help of an optimization algorithm
- Ex: The weights vector for a linear model, the splits of a tree

Hyperparameters Tuning

- Model **Parameters** and **Hyperparameters** are 2 **separate** things!

Parameters

- Are learned during the training process
- Are the result of the minimization of a given loss function with the help of an optimization algorithm
- Ex: The weights vector for a linear model, the splits of a tree

Hyperparameters

- Need to be specified outside of the training procedure
- They control the structure of our model
- They can be obtained thanks to a tuning meta-process
- Ex: Learning rate, Max Depth for a Tree…

Hyperparameters Tuning

- Model **Parameters** and **Hyperparameters** are 2 **separate** things!

Parameters

- Are learned during the training process
- Are the result of the minimization of a given loss function with the help of an optimization algorithm
- Ex: The weights vector for a linear model, the splits of a tree

Hyperparameters

- Need to be specified outside of the training procedure
- They control the structure of our model
- They can be obtained thanks to a tuning meta-process
- Ex: Learning rate, Max Depth for a Tree…

- Conceptually, Hyperparameters tuning can be seen as an **optimization loop** on top of ML model learning to find the set of hyper-parameters leading to the **lowest validation error**

Hyperparameters Tuning

- Hyperparameter settings can have a big impact on the prediction accuracy of a model

Hyperparameters Tuning

- Hyperparameter settings can have a big impact on the prediction accuracy of a model

- Optimal hyperparameter settings differ from a dataset to another. They should be tuned for each dataset

Hyperparameters Tuning

- Hyperparameter settings can have a big impact on the prediction accuracy of a model

- Optimal hyperparameter settings differ from a dataset to another. They should be tuned for each dataset

- Each trial of a particular hyperparameter setting involves **training a model** and **evaluating it** on a validation set. This can be very **costly**

Hyperparameters Tuning

- Hyperparameter settings can have a big impact on the prediction accuracy of a model

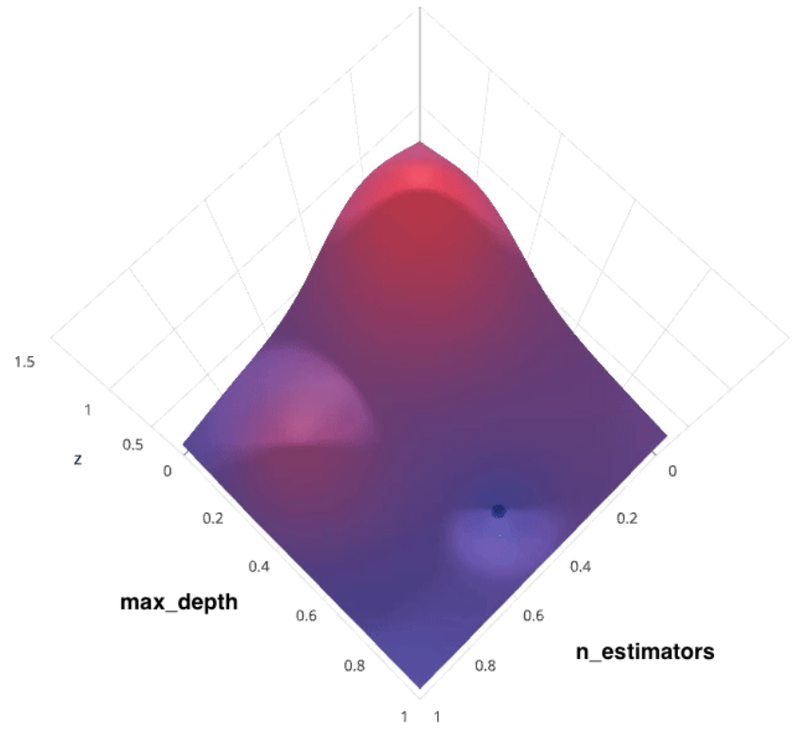
- Optimal hyperparameter settings differ from a dataset to another. They should be tuned for each dataset

- Each trial of a particular hyperparameter setting involves **training a model** and **evaluating it** on a validation set. This can be very **costly**

- We cannot rely on a gradient like we did for training the model. Instead, at each iteration the system must try **blindly** a new configuration in the search space, or make an educated guess of where the most interesting configuration might be

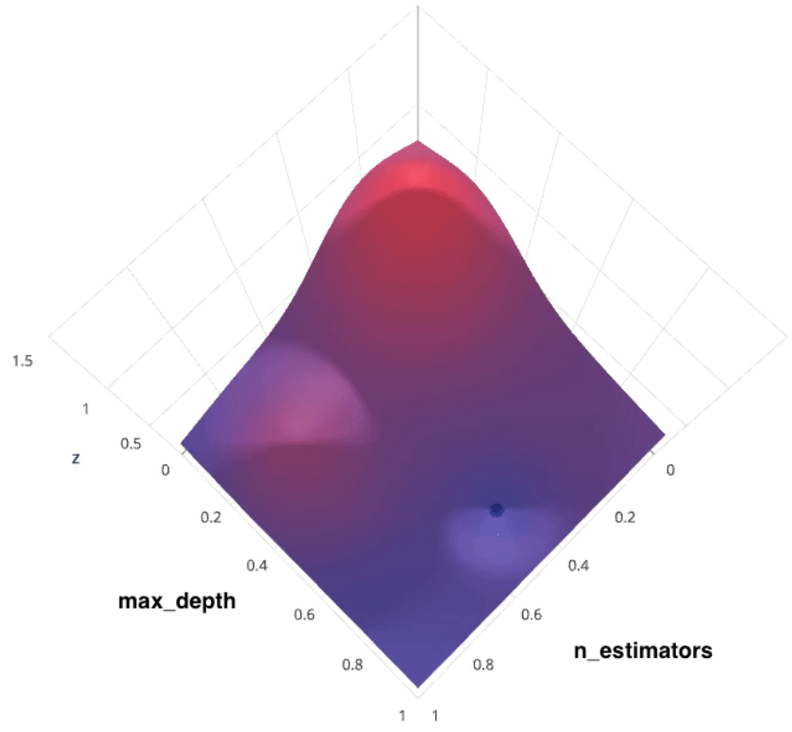
Hyperparameters Tuning

- Ideally, we would want to have an **analytical** expression of the model performance as a function of the **hyperparameters** and use a gradient descent algorithm



Hyperparameters Tuning

- Ideally, we would want to have an **analytical** expression of the model performance as a function of the **hyperparameters** and use a gradient descent algorithm



- The choice would be trivial. But we can’t have this kind of function, unless we evaluate the performance of each **possible combination**

Hyperparameters Tuning

- Instead, the most common methods are **Grid Search** & **Random Search** that use an **exhaustive** search of the space

Hyperparameters Tuning

- Instead, the most common methods are **Grid Search** & **Random Search** that use an **exhaustive** search of the space

- In Grid Search:

Hyperparameters Tuning

- Instead, the most common methods are **Grid Search** & **Random Search** that use an **exhaustive** search of the space

- In Grid Search:
  - We construct a **Grid** of the search space by specifying a **finite** set of possible values for each parameter and constructing the **cartesian product**

Hyperparameters Tuning

- Instead, the most common methods are **Grid Search** & **Random Search** that use an **exhaustive** search of the space

- In Grid Search:
  - We construct a **Grid** of the search space by specifying a **finite** set of possible values for each parameter and constructing the **cartesian product**
  - The algorithm launches a learning with **each** of the hyperparameters combinations

Hyperparameters Tuning

- Instead, the most common methods are **Grid Search** & **Random Search** that use an **exhaustive** search of the space

- In Grid Search:
  - We construct a **Grid** of the search space by specifying a **finite** set of possible values for each parameter and constructing the **cartesian product**
  - The algorithm launches a learning with **each** of the hyperparameters combinations
  - We select the **best** combination

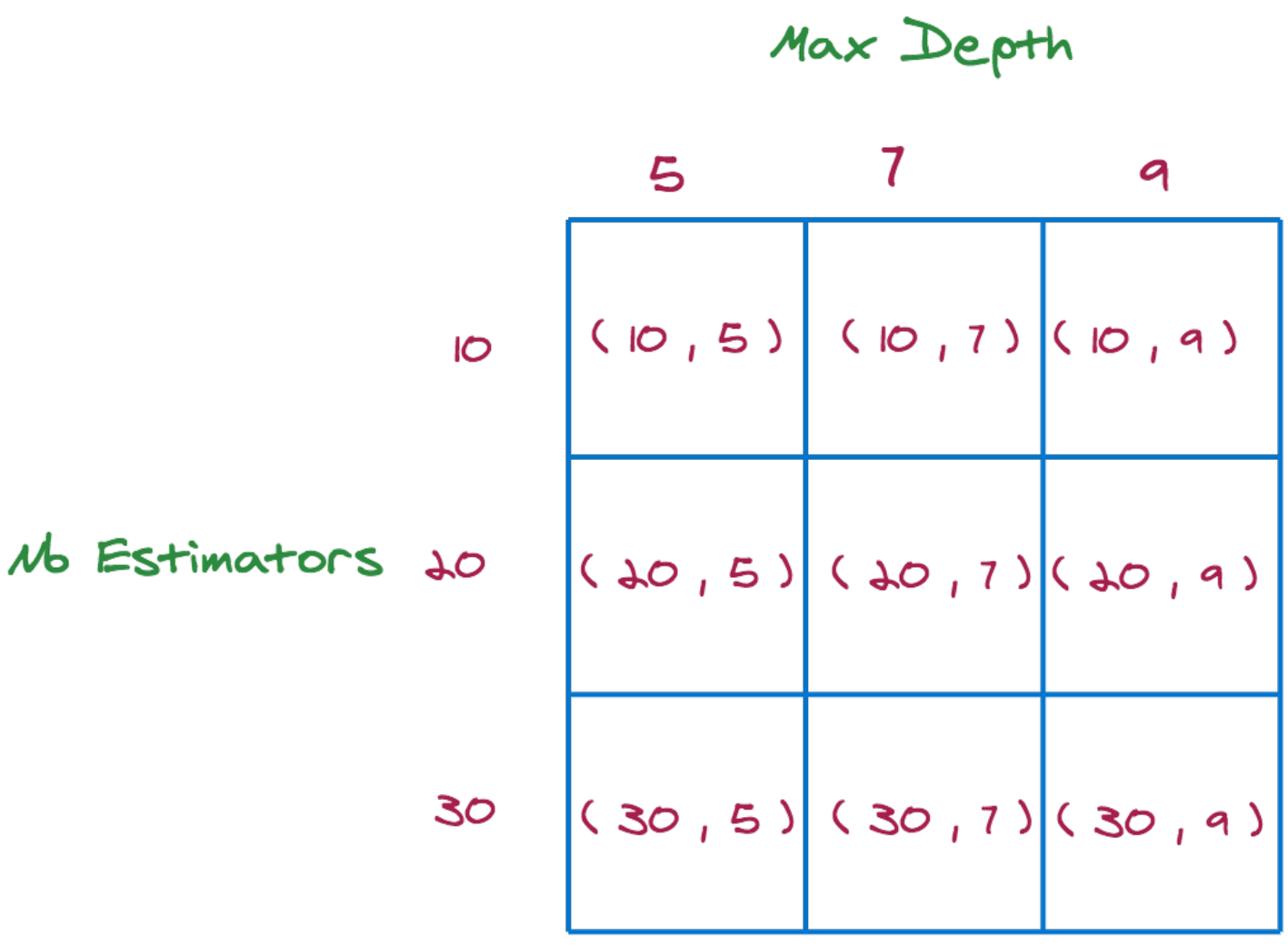
Hyperparameters Tuning

- Example: Random Forest Grid Search

Nb Estimators \ Max Depth	5	7	9
10	(10, 5)	(10, 7)	(10, 9)
20	(20, 5)	(20, 7)	(20, 9)
30	(30, 5)	(30, 7)	(30, 9)

Hyperparameters Tuning

- Example: Random Forest Grid Search



- We will train and evaluate 9 models: one for each parameters' combination
- We will keep the best performing one

Hyperparameters Tuning

- The **Random Search** is a variation of the **Grid Search** where we **randomly sample** the search space instead of **discretizing** it with a Cartesian Grid

Bengio & Bergstra, 2012

Hyperparameters Tuning

- The **Random Search** is a variation of the **Grid Search** where we **randomly sample** the search space instead of **discretizing** it with a Cartesian Grid

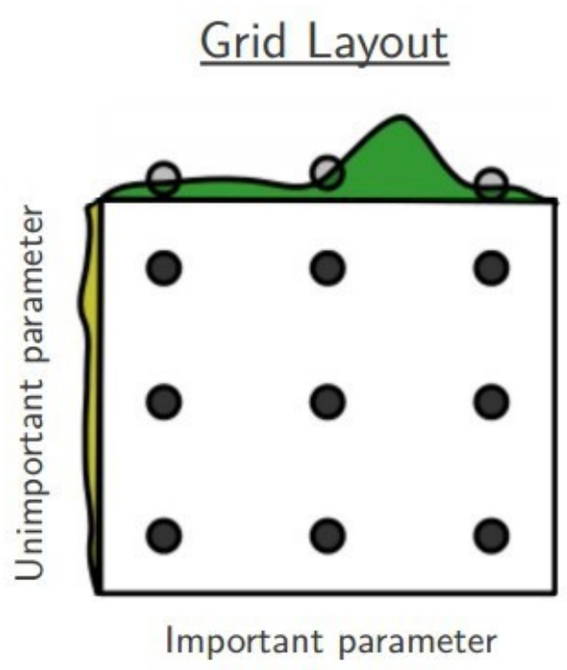
- It can be **more efficient** than Grid Search as in most cases, the hyperparameters are **not equally important**

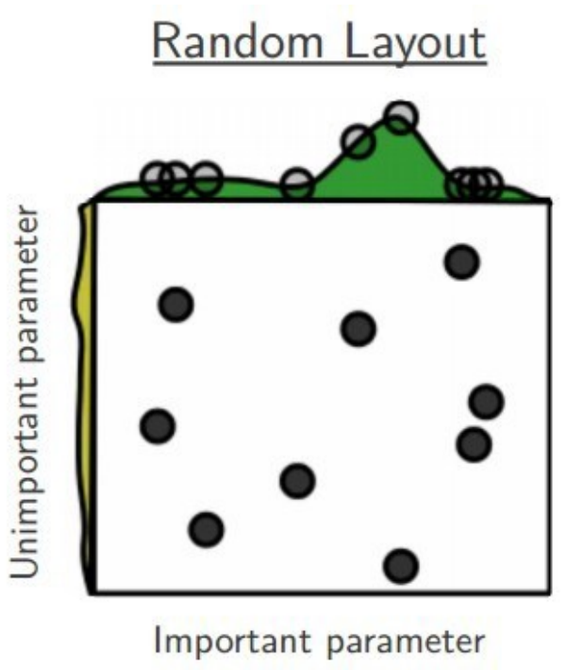
Bengio & Bergstra, 2012

Hyperparameters Tuning

- The **Random Search** is a variation of the **Grid Search** where we **randomly sample** the search space instead of **discretizing** it with a Cartesian Grid

- It can be **more efficient** than Grid Search as in most cases, the hyperparameters are **not equally important**





Bengio & Bergstra, 2012

Hyperparameters Tuning

- Grid Search and Random Search yield **acceptable performances**

Hyperparameters Tuning

- Grid Search and Random Search yield **acceptable performances**
- They are **easy to code** and **to parallelize** and they don’t need tuning

Hyperparameters Tuning

- Grid Search and Random Search yield **acceptable performances**
- They are **easy to code** and **to parallelize** and they don't need tuning
- However they can be quite **costly** if the number of combinations to test is **high**

Hyperparameters Tuning

- Grid Search and Random Search yield **acceptable performances**
- They are **easy to code** and **to parallelize** and they don't need tuning
- However they can be quite **costly** if the number of combinations to test is **high**
- The major drawback is that there are **no guarantee of finding a local minimum** except if the space is thoroughly searched

Hyperparameters Tuning

- Grid Search and Random Search yield **acceptable performances**
- They are **easy to code** and **to parallelize** and they don't need tuning
- However they can be quite **costly** if the number of combinations to test is **high**
- The major drawback is that there are **no guarantee of finding a local minimum** except if the space is thoroughly searched
- There are some **smarter** algorithms that can yield better results through **intelligent sampling**, but this is out of the scope of this class