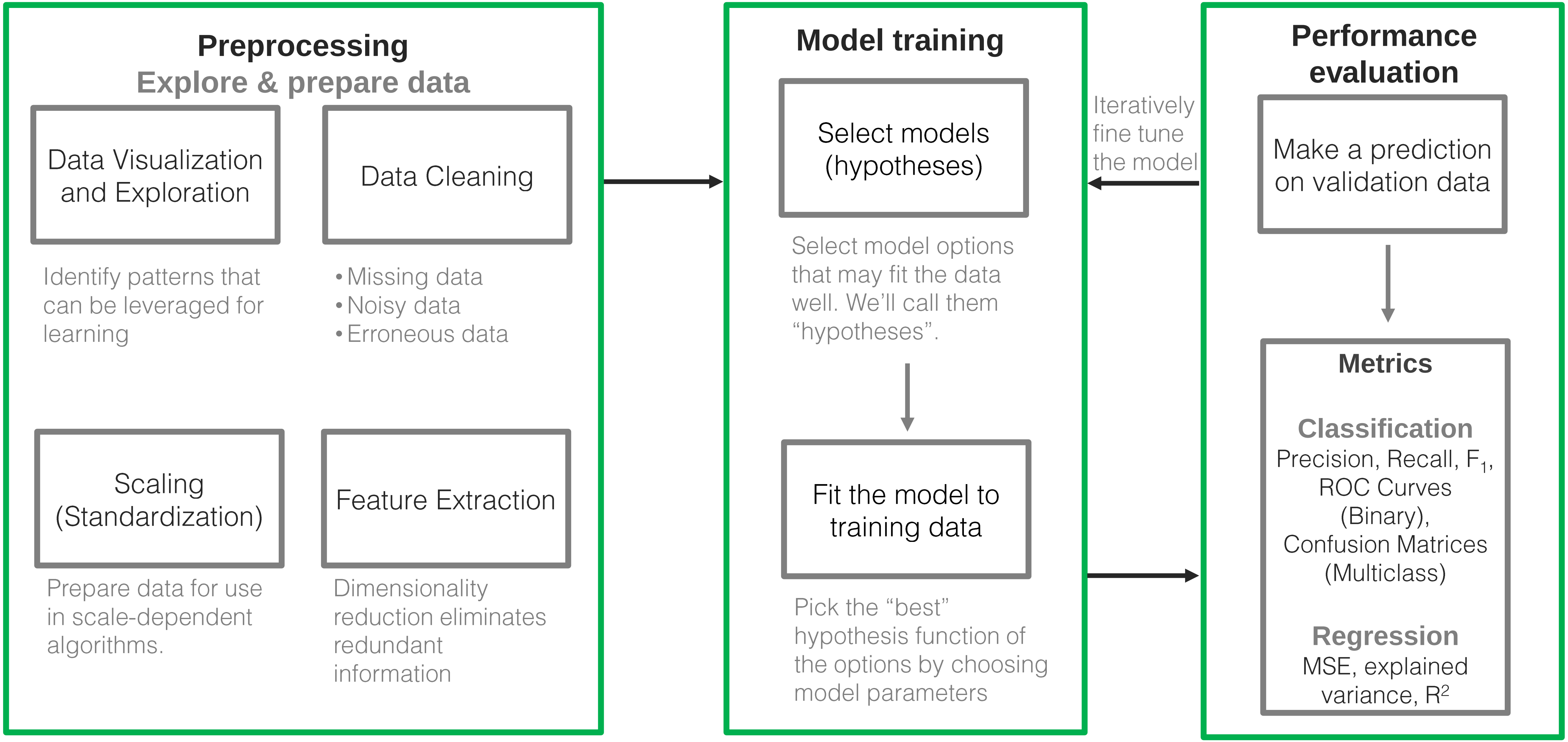
Linear models I

Regression

Supervised learning in practice



Preprocessing

Explore & prepare data

Data Visualization and Exploration

Identify patterns that can be leveraged for learning

Data Cleaning

- Missing data
- Noisy data
- Erroneous data

Scaling (Standardization)

Prepare data for use in scale-dependent algorithms.

Feature Extraction

Dimensionality reduction eliminates redundant information

Model training

Select models (hypotheses)

Select model options that may fit the data well. We'll call them "hypotheses".

Fit the model to training data

Pick the "best" hypothesis function of the options by choosing model parameters

Iteratively fine tune the model

Performance evaluation

Make a prediction on validation data

Metrics

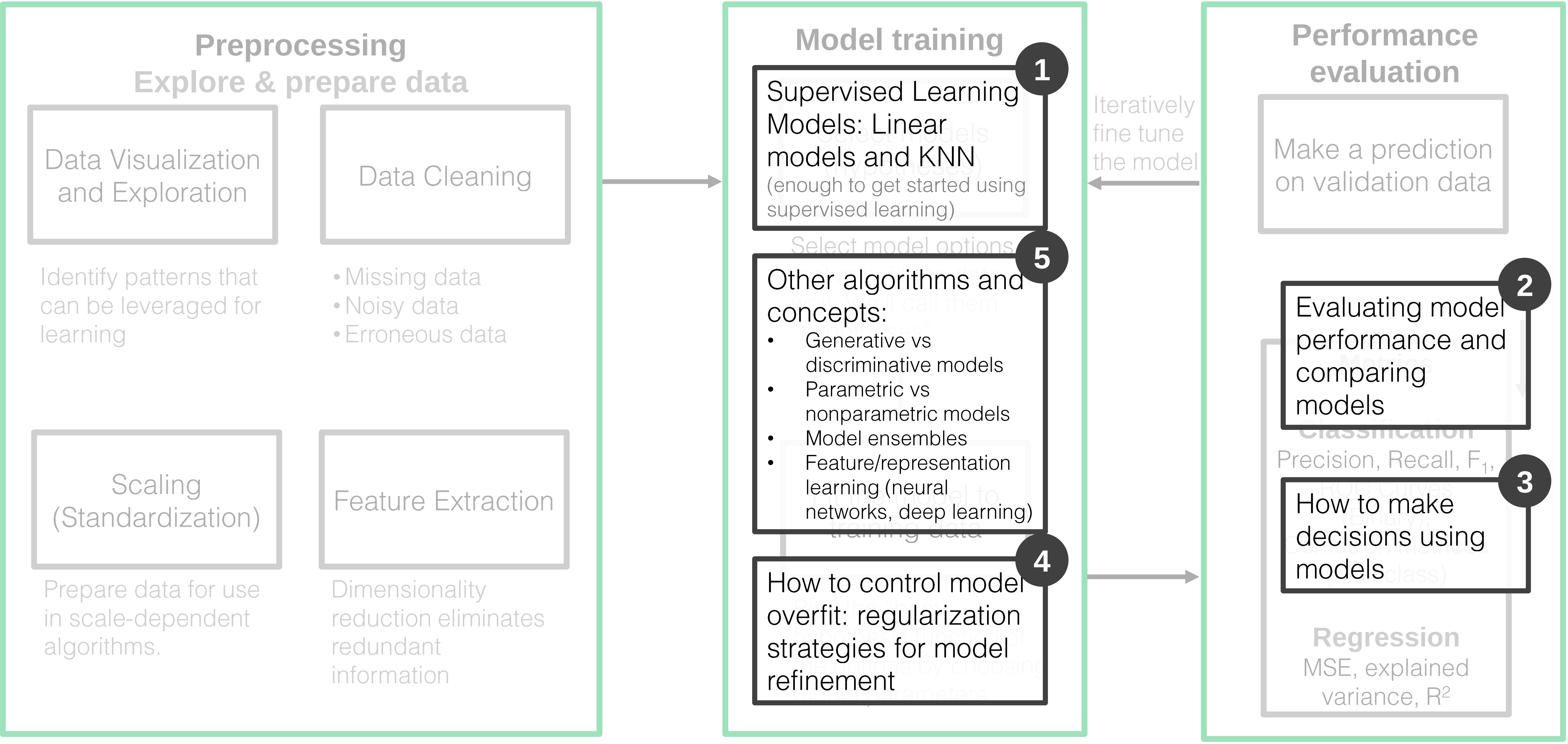
Classification

Precision, Recall, F_1, ROC Curves (Binary), Confusion Matrices (Multiclass)

Regression

MSE, explained variance, R^2

Supervised learning in practice



Preprocessing
Explore & prepare data

Data Visualization and Exploration

Identify patterns that can be leveraged for learning

Data Cleaning

- Missing data
- Noisy data
- Erroneous data

Scaling (Standardization)

Prepare data for use in scale-dependent algorithms.

Feature Extraction

Dimensionality reduction eliminates redundant information

Model training

1 Supervised Learning Models: Linear models and KNN (enough to get started using supervised learning)

Select model options

5 Other algorithms and concepts:
- Generative vs discriminative models
- Parametric vs nonparametric models
- Model ensembles
- Feature/representation learning (neural networks, deep learning)

4 How to control model overfit: regularization strategies for model refinement

Iteratively fine tune the model

Performance evaluation

Make a prediction on validation data

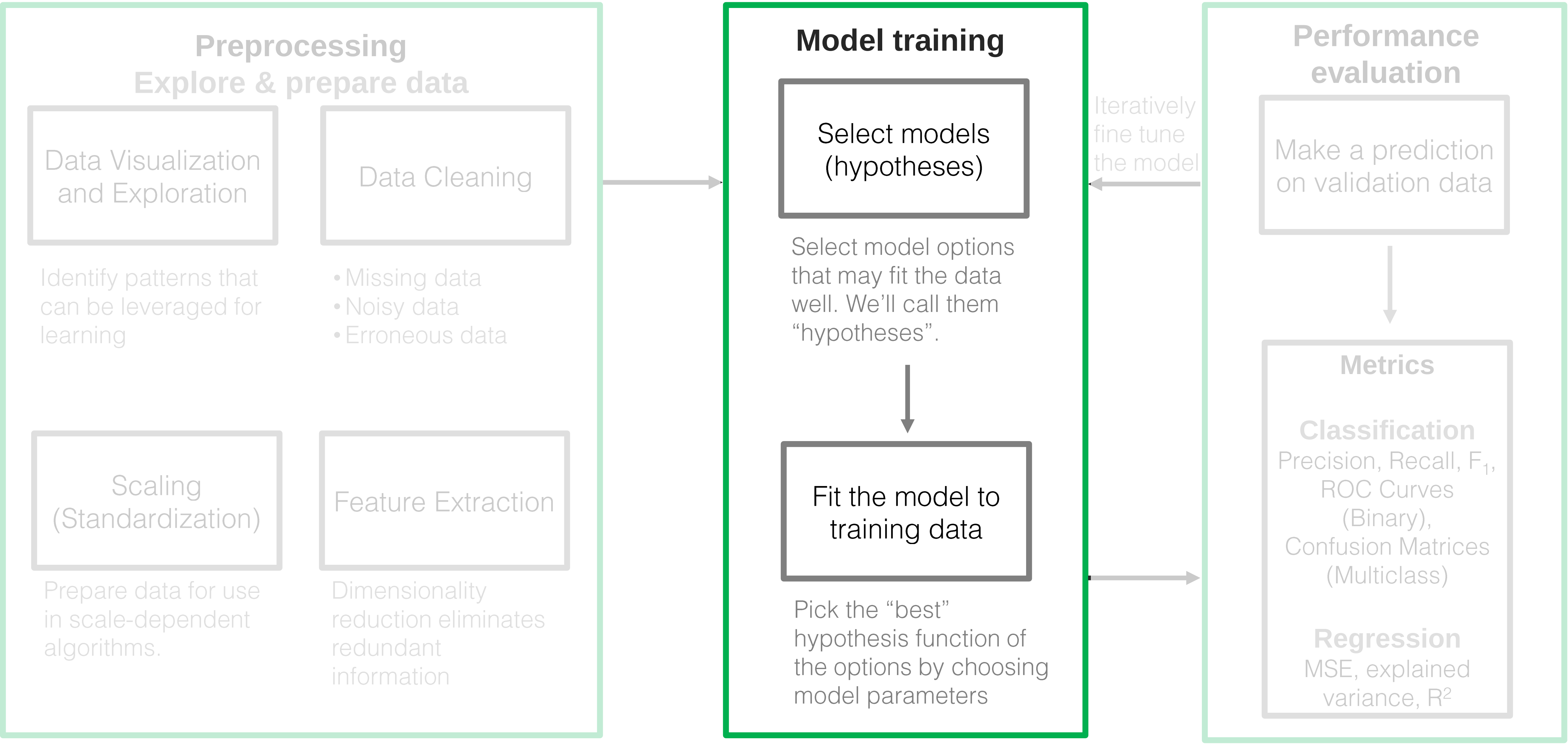
2 Evaluating model performance and comparing models

Precision, Recall, F_1,

3 How to make decisions using models

Regression
MSE, explained variance, R^2

Supervised learning in practice



Preprocessing
Explore & prepare data

Data Visualization and Exploration

Identify patterns that can be leveraged for learning

Data Cleaning

- Missing data
- Noisy data
- Erroneous data

Scaling (Standardization)

Prepare data for use in scale-dependent algorithms.

Feature Extraction

Dimensionality reduction eliminates redundant information

Model training

Select models (hypotheses)

Select model options that may fit the data well. We'll call them "hypotheses".

Fit the model to training data

Pick the "best" hypothesis function of the options by choosing model parameters

Iteratively fine tune the model

Performance evaluation

Make a prediction on validation data

Metrics

Classification
Precision, Recall, F_1, ROC Curves (Binary), Confusion Matrices (Multiclass)

Regression
MSE, explained variance, R^2

Model Fitting / Training Process

1. Choose a **hypothesis set of models** to train
   (e.g. linear regression with 4 predictor variables)

2. Identify a **cost function** to measure the model fit to the training data
   (e.g. mean square error)

3. **Optimize** model **parameters** to minimize cost
   (e.g. closed form solution using the normal equations for OLS)

★ We will use this procedure for ALL the parametric models we encounter

Parametric models = models where the number of parameters are fixed and independent of the training data size

How can we…

define what makes a model linear?

fit our model to our training data?

What makes a model linear?

Which of the following models are linear?

Model parameters are w_i

Target variable is y

Remaining components are features

A $y = w_0$

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + x_2^{w_2}$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1} dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$

Linear models are linear in the parameters

A linear combination is quantity where a set of terms are added together, each multiplied by a constant (parameter)

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

In other words, a linear combination is a sum of scalar multiples of vectors:

$$\boldsymbol{z} = 2\boldsymbol{x} + 99.9999\boldsymbol{y}$$

$$\boldsymbol{z} = 0.333\boldsymbol{x} + (-42)\boldsymbol{y}$$

$$\boldsymbol{z} = 0\boldsymbol{x} + \left(\frac{\pi^2}{e}\right)\boldsymbol{y}$$

Linear regression model

Parameters	w_j
Target	y_i
Features	$x_{i,j}$

Sample Feature

$$y_i = \sum_{j=0}^{p} w_j x_{i,j} = \mathbf{w}^\top \mathbf{x}_i$$

$$y_i = w_0 x_{i,0} + w_1 x_{i,1} + w_2 x_{i,2} + \cdots + w_p x_{i,p}$$

Intercept/bias term: $x_{i,0} \triangleq 1$

$$y_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \cdots + w_p x_{i,p}$$

Which of the following models are linear?

Model parameters are w_i

Target variable is y

Remaining components are features

A $y = w_0$

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + x_2^{w_2}$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1} dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2 , x_3)$

Which of the following models are linear?

A $y = w_0$

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + x_2^{w_2}$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1}\, dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$

Model parameters are w_i

Target variable is y

Remaining components are features

ALL except C are linear in the **parameters, w**

We can rewrite these models in terms of transformed features

It becomes clear this is a linear model when we rewrite the features:

$$y = w_0 + w_1 z_1 + w_2 z_2 + w_3 z_3$$

Transformed features:

$$z_1 = \int \sqrt[3]{x_1}\, dx_1 \qquad z_2 = g(x_2) \qquad z_2 = median(x_1, x_2, x_3)$$

F $$y = w_0 + w_1 \int \sqrt[3]{x_1}\, dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$$

$$y_i = \sum_{j=0}^{p} w_j x_{i,j} = \mathbf{w}^\top \mathbf{x}_i$$

Linear regression assumptions

1. **Linearity**. Linear relationship between feature and target variables
2. **Normality**. Error is normally distributed
3. **Independence**. Assumes observations are independent from one another (no autocorrelation)
4. **Homoscedascity**. Variance of the error is constant
5. **Little multicollinearity**. Features are not correlated with one another

Notation

$x_{i,j}$

Sample Feature

		Number of features			
		1		p	
Number of samples	1	$x_{i,j}$	Shorthand x_i	$\boldsymbol{x}_i = \begin{bmatrix} x_{i,0} \\ x_{i,1} \\ \vdots \\ x_{i,p} \end{bmatrix}$	Shorthand Assume $x_{i,0} = 1$ $\boldsymbol{x}_i = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{bmatrix}$
	N	$\boldsymbol{x} = \begin{bmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{N,j} \end{bmatrix}$	Shorthand $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$	$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,p} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,0} & x_{N,1} & \cdots & x_{N,p} \end{bmatrix}$	

Types of Linear Regression

Here, $x_0 \triangleq 1$

<table>
<tr><td></td><td>One feature variable</td><td>Two or more feature variables</td></tr>
<tr><td>One target variable</td><td>Simple Linear Regression
$y_i = w_0 + w_1 x_{i,1}$</td><td>Multiple Linear Regression
$y_i = \sum_{j=0}^{p} w_j x_{i,j}$ or $y_i = \boldsymbol{w}^\top \boldsymbol{x}_i$</td></tr>
<tr><td>Two or more target variables</td><td colspan="2">Multivariate (Multiple) Linear Regression
q = # target variables
[q x 1]
$\boldsymbol{y}_i = \boldsymbol{W}^\top \boldsymbol{x}_i$
[q x p] [p x 1]
p = # features (includes $x_0 = 1$)</td></tr>
</table>

Linear models: pros and cons

Pros

Simple/fast to implement and interpret

Excels if the relationship between features and targets is linear or can be expressed in terms of linear combinations of features

Often a good starting point or baseline model for many analyses

Cons

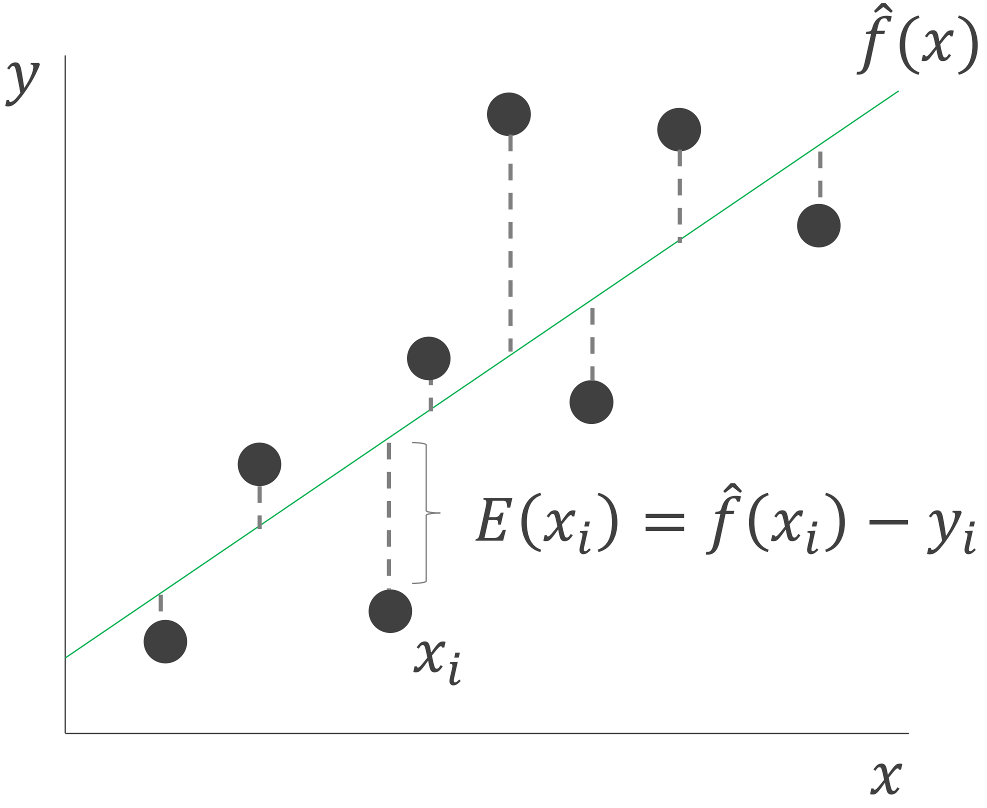
For many applications with complex feature-target relationships, underfits

Requires feature engineering for capturing more complex feature-target relationships

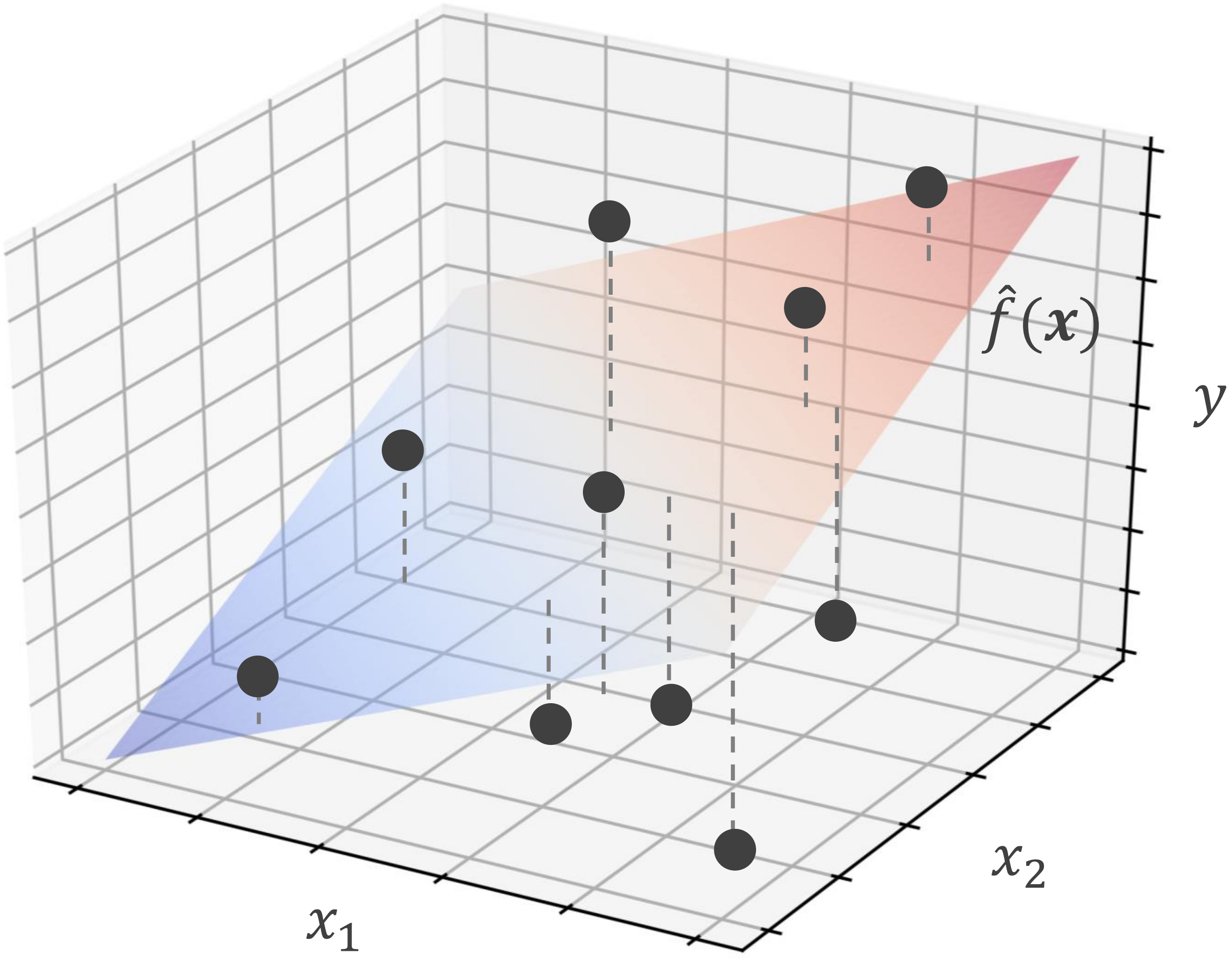
How do we fit the model to the training data?

A winding path to the least squares solution

Linear models and error



simple linear regression



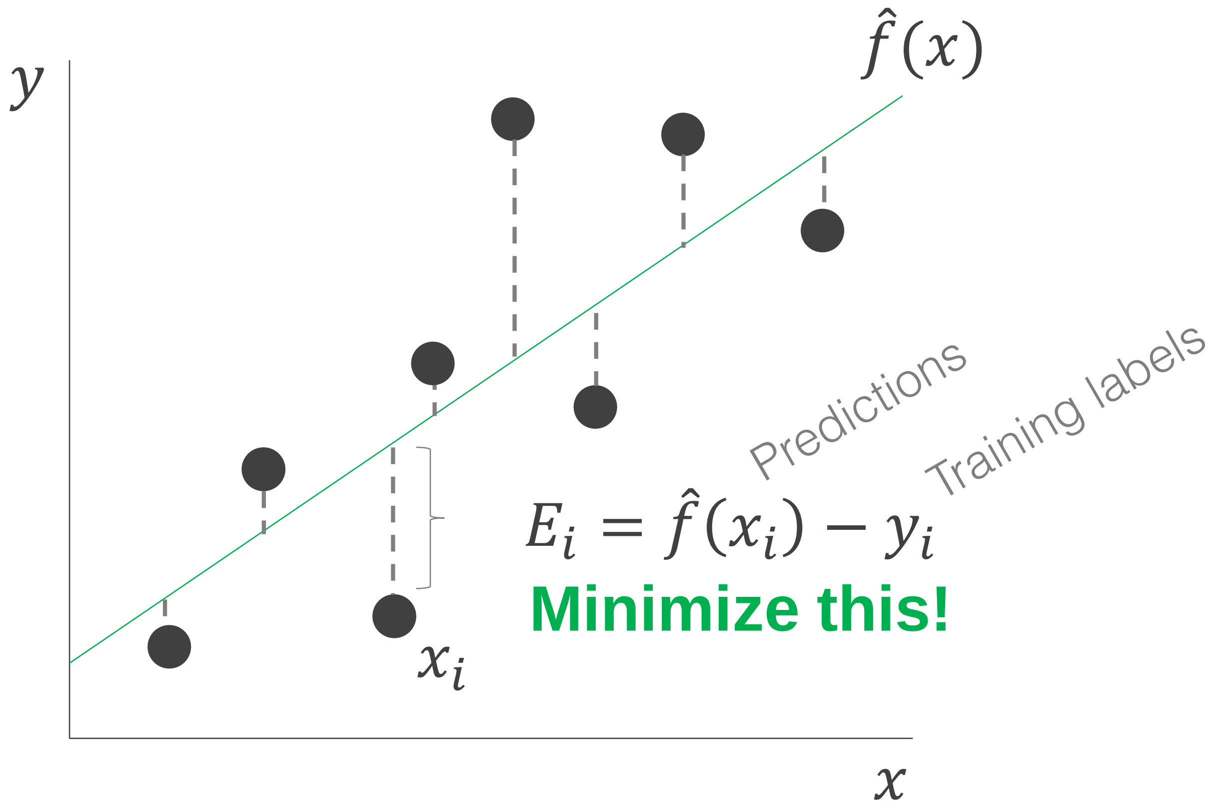
multiple linear regression

How do we fit a linear model to training data?

Training data:

$$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)\}$$

We want the error between our predictions and training data to be small



$E_i = \hat{f}(x_i) - y_i$

Minimize this!

How do we measure error?

The error for one sample is:

$$E_i = \hat{f}(\boldsymbol{x}_i) - y_i$$

$$\hat{y}_i = \hat{f}(\boldsymbol{x}_i) = \sum_{j=0}^{p} w_j x_{i,j}$$

We want to minimize the error across all our training data, so…
we use mean squared error to quantify training (in-sample) error:

Training (in-sample) error: $$E_{in}(\hat{f}, D) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

We call this our **Cost Function** (a.k.a. loss, error, or objective)

Cost Function: $$C(\overset{\text{model}}{\hat{f}}, \overset{\text{training data}}{D}) = E_{in}(\hat{f}, D) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\mathbf{x}_n) - y_n\right)^2$$

Training error is a function of our **model** and the **training data**

We can’t change the data; we adjust our model to minimize cost

To adjust the model, we choose model **parameters** that minimize cost

This is an **optimization** problem

How to fit our model to the training data?

Equivalently: how do we choose $\boldsymbol{w}$ to minimize cost (error)

$$E_{in}(\hat{f}, D) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

where $\hat{f}(\boldsymbol{x}_n) = \boldsymbol{w}^T\boldsymbol{x}_n$

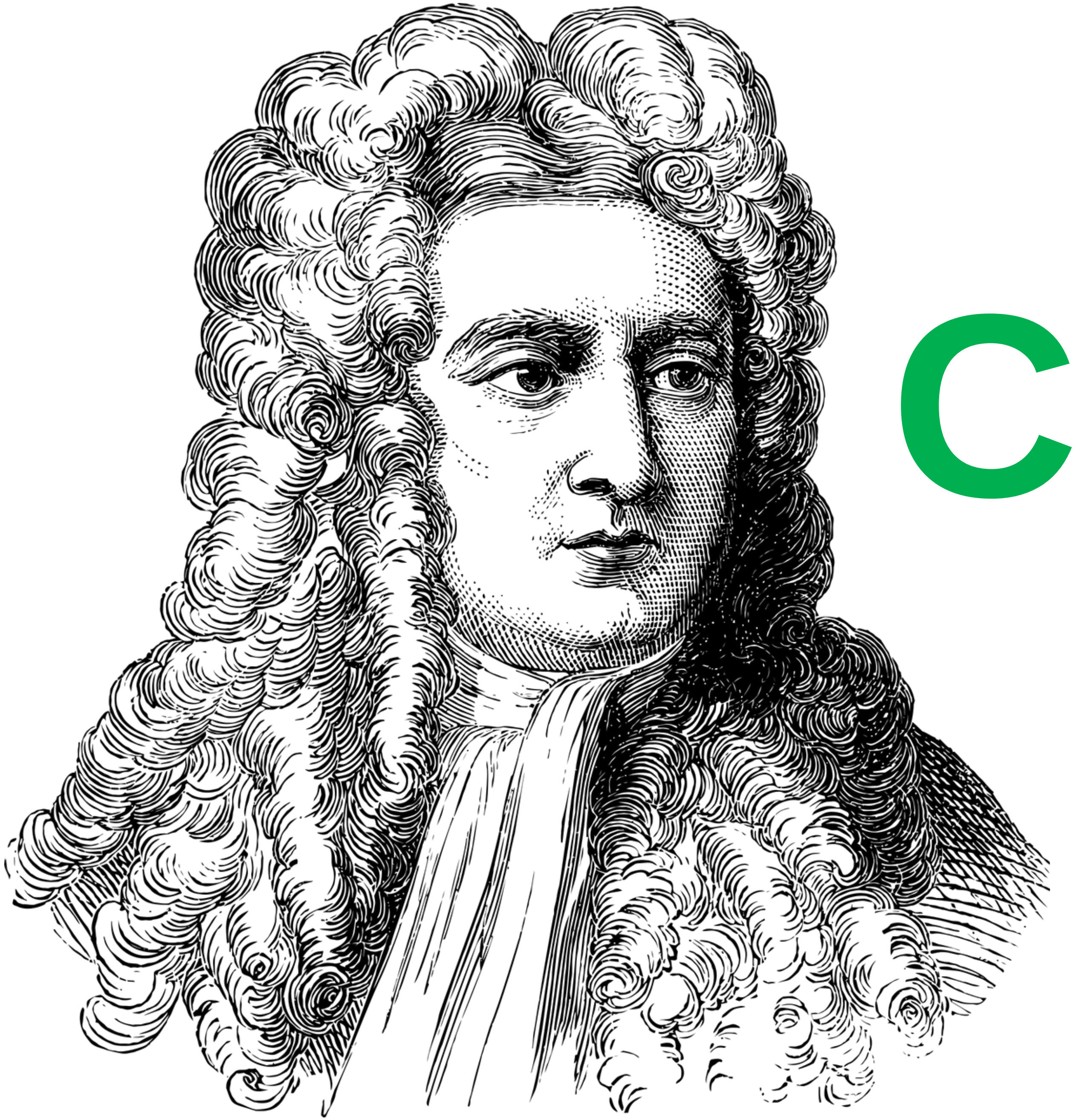
We want to minimize

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$

…by varying $\boldsymbol{w}$

How do we do that?

Once we've determined the form of the model and the training data, the model parameters $\boldsymbol{w}$ are the only parts we can adjust



Calculus

A moment of calculus

Function of one variable

$$f(x) = ax + bx^2$$

Function of multiple variables

$$f(x_1, x_2) = ax_1 + bx_2$$

Derivative

$$\frac{df}{dx} = a + 2bx$$

Partial Derivative

$$\frac{\partial f}{\partial x_1} = a$$

$$\frac{\partial f}{\partial x_2} = b$$

Gradient

$$\nabla_{\boldsymbol{x}} f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

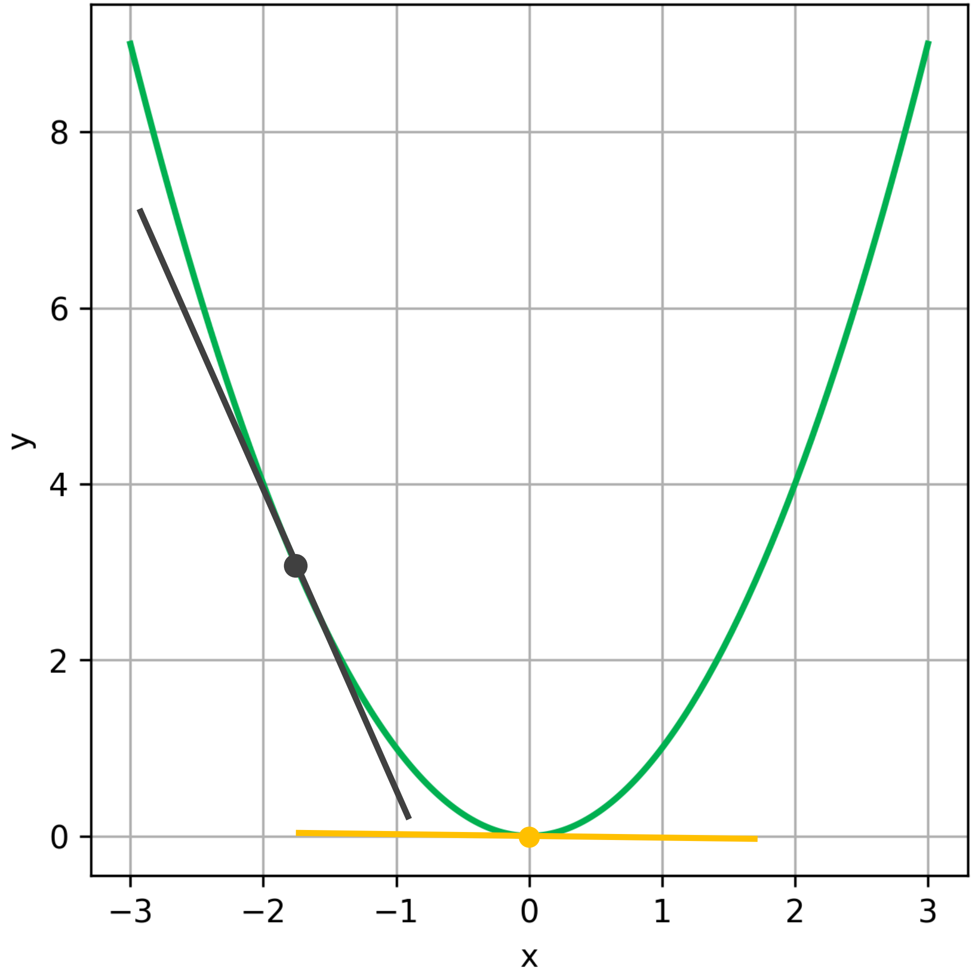
$$= \begin{bmatrix} a \\ b \end{bmatrix}$$

May also treat parameters as variables and take their partial derivative $\frac{\partial f}{\partial b} = x_2$

Why set the derivative to 0?

Remember the derivative (and gradient) is the direction and rate of fastest increase

For a continuous, convex function, the function is minimized when that derivative is zero



How to fit our model to the training data?

Take the gradient with respect to $\boldsymbol{w}$, set it to zero, and solve for $\boldsymbol{w}$

(think derivative)

$$\nabla_{\boldsymbol{w}} E_{in}(\boldsymbol{w}) = \nabla_{w} \left(\frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 \right)$$

$$\nabla_{\boldsymbol{w}} E_{in}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial E_{in}}{\partial w_0} \\ \frac{\partial E_{in}}{\partial w_1} \\ \vdots \\ \frac{\partial E_{in}}{\partial w_p} \end{bmatrix} = \boldsymbol{0}$$

Size: $[p + 1 \times 1]$ or $\mathbb{R}^{p+1\times1}$

p = number of predictors
N = number of data points

We will walk through the **ordinary least squares** (OLS) closed-form solution.

We could have used another optimization approach like **gradient descent**

How to fit our model to the training data?

Our cost function…

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$

Scalar

$$\boldsymbol{w}^T \in \mathbb{R}^{1\times p+1}$$
$$\boldsymbol{x}_n \in \mathbb{R}^{p+1\times 1}$$

…can be rewritten as:

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Convenient definitions:

$$\boldsymbol{y} \in \mathbb{R}^{N\times 1}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times p+1}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1\times 1}$$

p = number of predictors
N = number of data points

Aside on algebraic manipulations

p = number of predictors
N = number of data points

1 Assume $p = 2$
$N = 4$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \qquad \boldsymbol{x}_i = \begin{bmatrix} x_{i,0} \\ x_{i,1} \\ x_{i,2} \end{bmatrix}$$

2

$$\boldsymbol{w}^T \boldsymbol{x}_i = \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_{i,0} \\ x_{i,1} \\ x_{i,2} \end{bmatrix}$$

$$= w_0 x_{i,0} + w_1 x_{i,1} + w_2 x_{i,2}$$

3

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \boldsymbol{x}_3^T \\ \boldsymbol{x}_4^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \\ x_{3,0} & x_{3,1} & x_{3,2} \\ x_{4,0} & x_{4,1} & x_{4,2} \end{bmatrix}$$

4

$$\boldsymbol{X}\boldsymbol{w} = \begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \\ x_{3,0} & x_{3,1} & x_{3,2} \\ x_{4,0} & x_{4,1} & x_{4,2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}^T \boldsymbol{x}_1 \\ \boldsymbol{w}^T \boldsymbol{x}_2 \\ \boldsymbol{w}^T \boldsymbol{x}_3 \\ \boldsymbol{w}^T \boldsymbol{x}_4 \end{bmatrix}$$

Aside on algebraic manipulations

④

$$\boldsymbol{X}\boldsymbol{w} = \begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \\ x_{3,0} & x_{3,1} & x_{3,2} \\ x_{4,0} & x_{4,1} & x_{4,2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}^T\boldsymbol{x}_1 \\ \boldsymbol{w}^T\boldsymbol{x}_2 \\ \boldsymbol{w}^T\boldsymbol{x}_3 \\ \boldsymbol{w}^T\boldsymbol{x}_4 \end{bmatrix}$$

⑤

$$\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y} = \begin{bmatrix} \boldsymbol{w}^T\boldsymbol{x}_1 - \boldsymbol{y}_1 \\ \boldsymbol{w}^T\boldsymbol{x}_2 - \boldsymbol{y}_2 \\ \boldsymbol{w}^T\boldsymbol{x}_3 - \boldsymbol{y}_3 \\ \boldsymbol{w}^T\boldsymbol{x}_4 - \boldsymbol{y}_4 \end{bmatrix}$$

⑥

$$(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = \begin{bmatrix} \boldsymbol{w}^T\boldsymbol{x}_1 - \boldsymbol{y}_1 & \boldsymbol{w}^T\boldsymbol{x}_2 - \boldsymbol{y}_2 & \boldsymbol{w}^T\boldsymbol{x}_3 - \boldsymbol{y}_3 & \boldsymbol{w}^T\boldsymbol{x}_4 - \boldsymbol{y}_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^T\boldsymbol{x}_1 - \boldsymbol{y}_1 \\ \boldsymbol{w}^T\boldsymbol{x}_2 - \boldsymbol{y}_2 \\ \boldsymbol{w}^T\boldsymbol{x}_3 - \boldsymbol{y}_3 \\ \boldsymbol{w}^T\boldsymbol{x}_4 - \boldsymbol{y}_4 \end{bmatrix}$$

$$= \sum_{n=1}^{N} (\boldsymbol{w}^T\boldsymbol{x}_n - \boldsymbol{y}_n)^2$$

How to fit our model to the training data?

Our cost function…

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$

Scalar

$\boldsymbol{w}^T \in \mathbb{R}^{1\times p+1}$

$\boldsymbol{x}_n \in \mathbb{R}^{p+1\times 1}$

…can be rewritten as:

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Convenient definitions:

$$\boldsymbol{y} \in \mathbb{R}^{N\times 1}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times p+1}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1\times 1}$$

p = number of predictors
N = number of data points

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

(take gradient, set to 0)

$$\nabla_{\boldsymbol{w}} E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}) = \boldsymbol{0}$$

(solve for $\boldsymbol{w}$)

Univariate analogy:

$$f(w) = \frac{1}{N}(xw - y)^2$$

$$= \frac{1}{N}(x^2w^2 - 2xyw + y^2)$$

$$\frac{df(w)}{dw} = \frac{2}{N}(x^2w - xy)$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$ (normal equation)

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Pseudoinverse $\mathbf{X}^\dagger = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$

$$\boxed{\boldsymbol{w}^* = \mathbf{X}^\dagger\boldsymbol{y}}$$

What is the pseudoinverse?

Samples and features impact solutions

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} \\ N \times p \\ \\ \end{bmatrix} \end{array} \begin{bmatrix} \\ p \times 1 \\ \\ \end{bmatrix} = \begin{bmatrix} N \times 1 \\ \\ \\ \end{bmatrix}$$

$$\boldsymbol{X} \quad \boldsymbol{w} = \boldsymbol{y}$$

If $N = p$, then there are the same number of features as samples

(# equations = # unknowns)

If $N > p$, then the system of equations is **overdetermined**: more samples than features

(# equations > # unknowns)

Can’t invert $\boldsymbol{X}$ – it’s not square!

If $N < p$, then the system of equations is **underdetermined**: fewer samples than features

(# equations < # unknowns)

Overdetermined systems

Example 1

Fully determined system

equations = # unknowns

$$\begin{aligned} w_1 &= 2 \\ w_2 &= 1 \end{aligned}$$

Overdetermined system

equations > # unknowns

$$\begin{aligned} w_1 &= 2 \\ w_2 &= 1 \\ w_2 &= 5 \end{aligned}$$

Example 2

Fully determined system

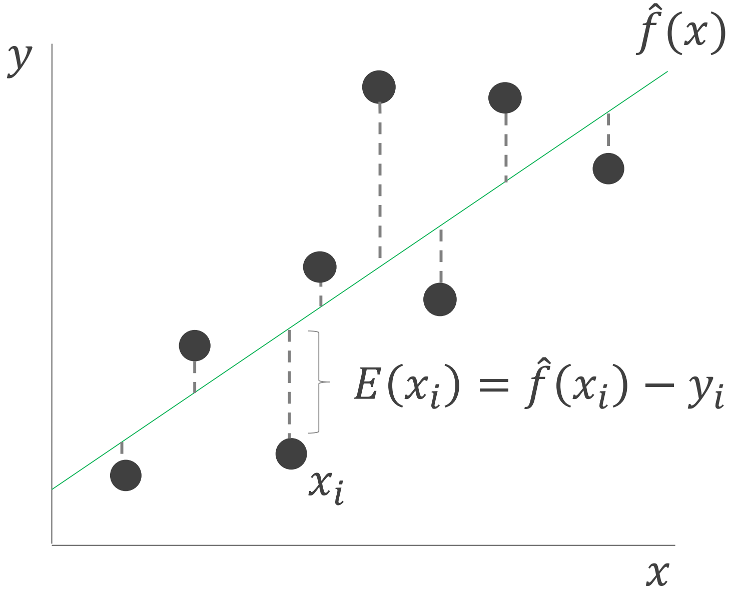
equations = # unknowns

$$\begin{aligned} 2w_1 + 3w_2 &= 2 \\ 6w_1 + 1w_2 &= 0.63 \end{aligned}$$

Overdetermined system

equations > # unknowns

$$\begin{aligned} 2w_1 + 3w_2 &= 2 \\ 6w_1 + 1w_2 &= 0.63 \\ 3w_1 + 2w_2 &= 14 \\ 16w_1 - w_2 &= 0.1 \end{aligned}$$



Underdetermined systems

Example 1

Fully determined system

equations = # unknowns

$$w_1 = 2$$
$$w_2 = 1$$

Underdetermined system

equations < # unknowns

$$w_1 = w_2$$

Example 2

Fully determined system

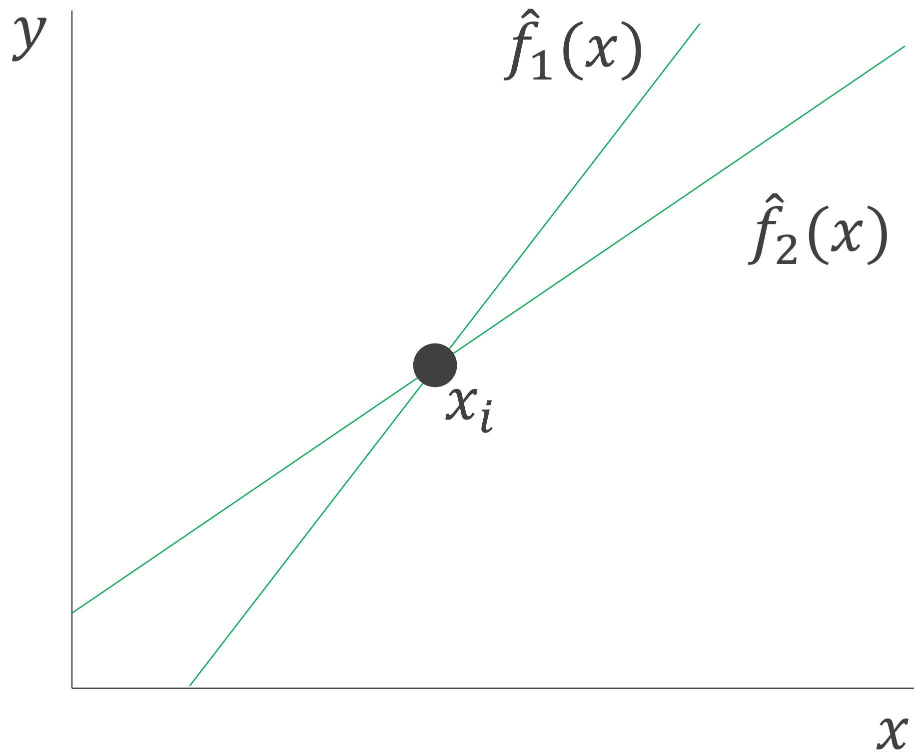
equations = # unknowns

$$2w_1 + 3w_2 = 2$$
$$6w_1 + 1w_2 = 0.63$$

Underdetermined system

equations < # unknowns

$$2w_1 + 3w_2 = 2$$



What is the pseudoinverse?

Consider the case when $N = 3, p = 2$

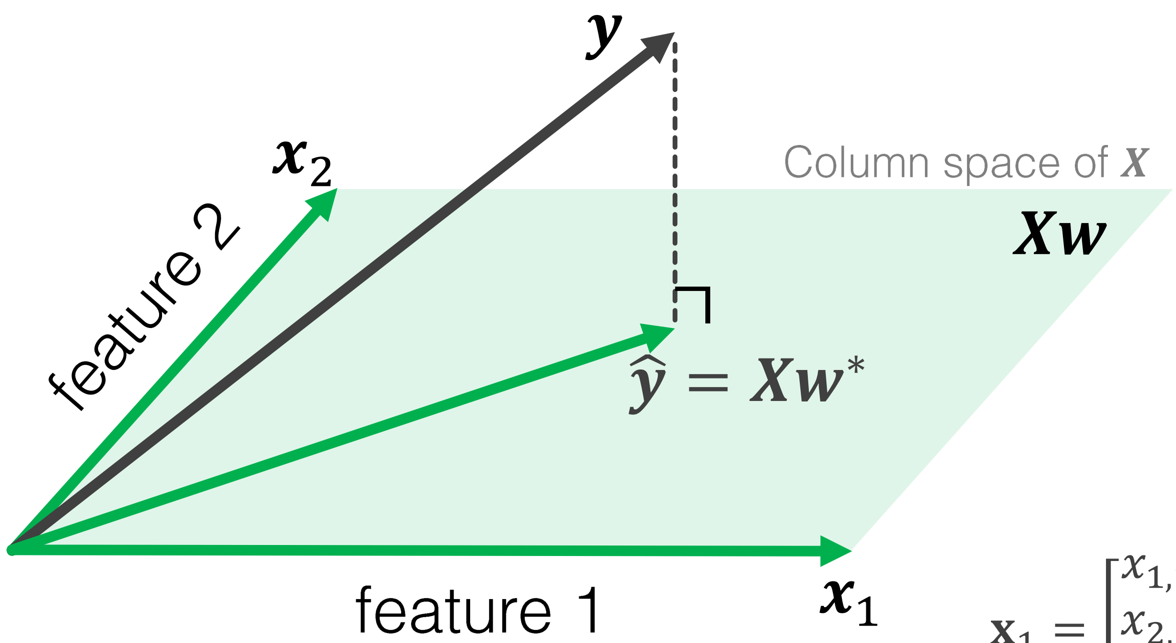
(Assume no bias term here)

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \\ \text{Feature vectors:} & \quad \boldsymbol{x}_1 \qquad\quad \boldsymbol{x}_2 \end{array} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\boldsymbol{X} \qquad \boldsymbol{w} \neq \boldsymbol{y}$$

We CAN use $\boldsymbol{X}^{\dagger}$ to obtain $\boldsymbol{w}^{*}$, the least squares solution: $$\boldsymbol{w}^{*} = \boldsymbol{X}^{\dagger}\boldsymbol{y}$$

The least squares solution is the best we can do given $N > p$



$$\mathbf{x}_1 = \begin{bmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \end{bmatrix}$$

Much of machine learning is **optimizing** a **cost function**

Least squares is one approach (when applicable), gradient descent is another, more generic method

Model Fitting / Training Process

1. Choose a **hypothesis set of models** to train
   (e.g. linear regression with 4 predictor variables)

2. Identify a **cost function** to measure the model fit to the training data
   (e.g. mean square error)

3. **Optimize** model **parameters** to minimize cost
   (e.g. closed form solution using the normal equations for OLS)

We now have our model parameters, we can make predictions on unseen test data!

The parameters learned from model fitting

$$\hat{y}_i = \hat{f}(\boldsymbol{x}_i) = \sum_{j=0}^{p} w_j^* x_{i,j}$$

Takeaways

Linear models are **linear in the weights**

Model fitting/training process (valid beyond linear models):

- Choose a hypothesis set of models to train
- Identify a cost function
- **Optimize the cost function** by adjusting model parameters
  (This is the "learning" process)

Optimize cost functions for linear regression using least squares

- Least squares allows us to generate approximate solutions to overdetermined systems (more samples than features/parameters), which are common
- Alternative optimization strategies exist such as gradient descent