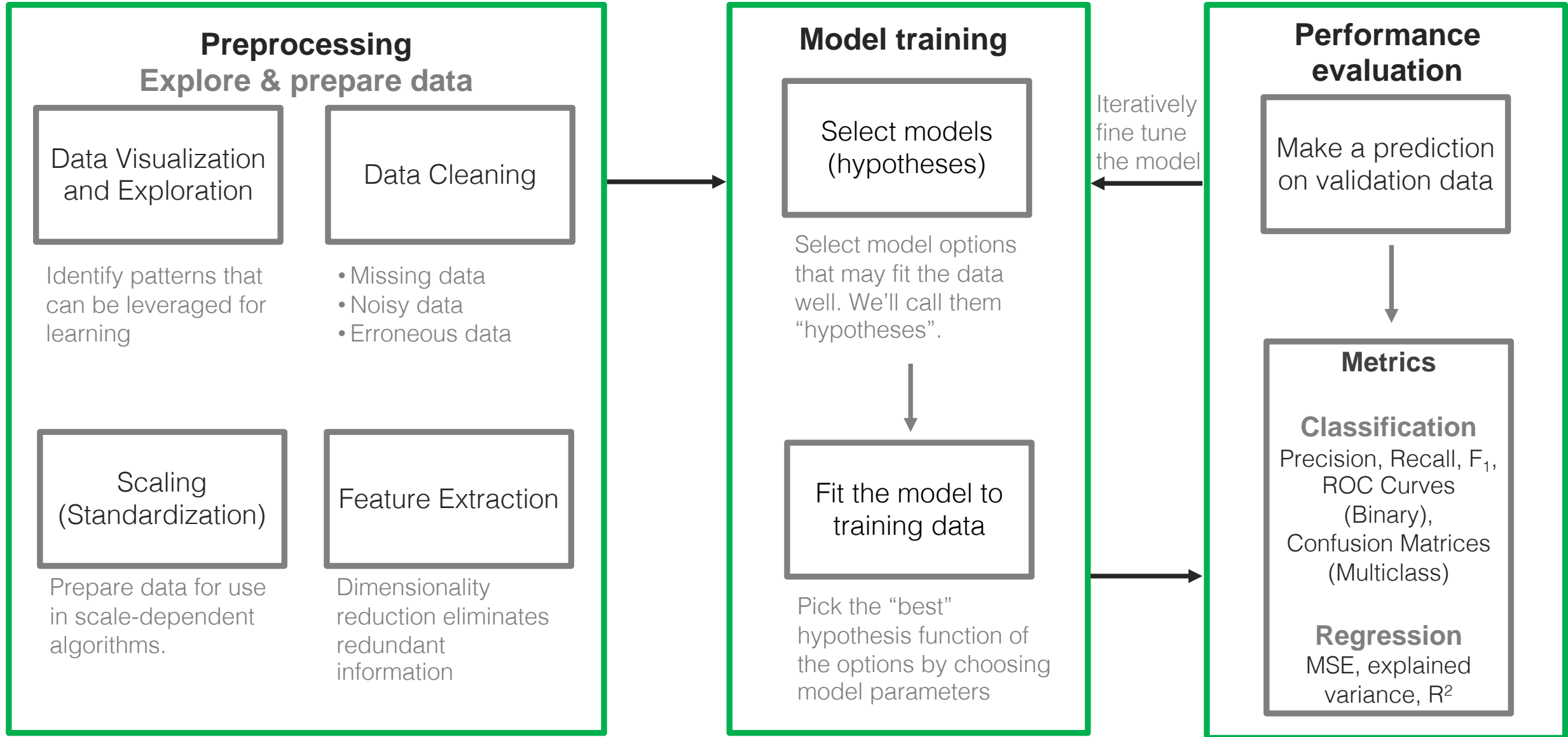


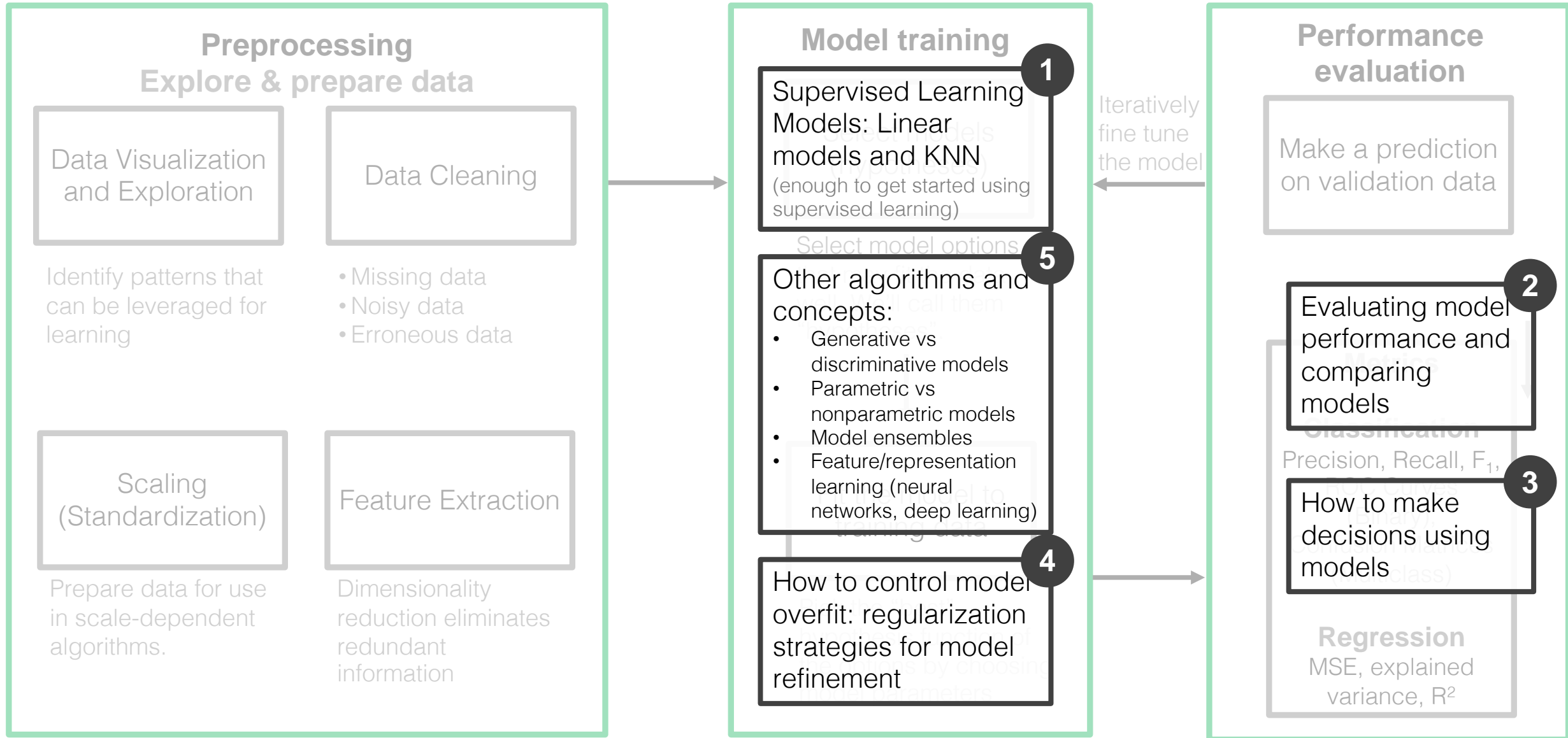
Linear models I

Regression

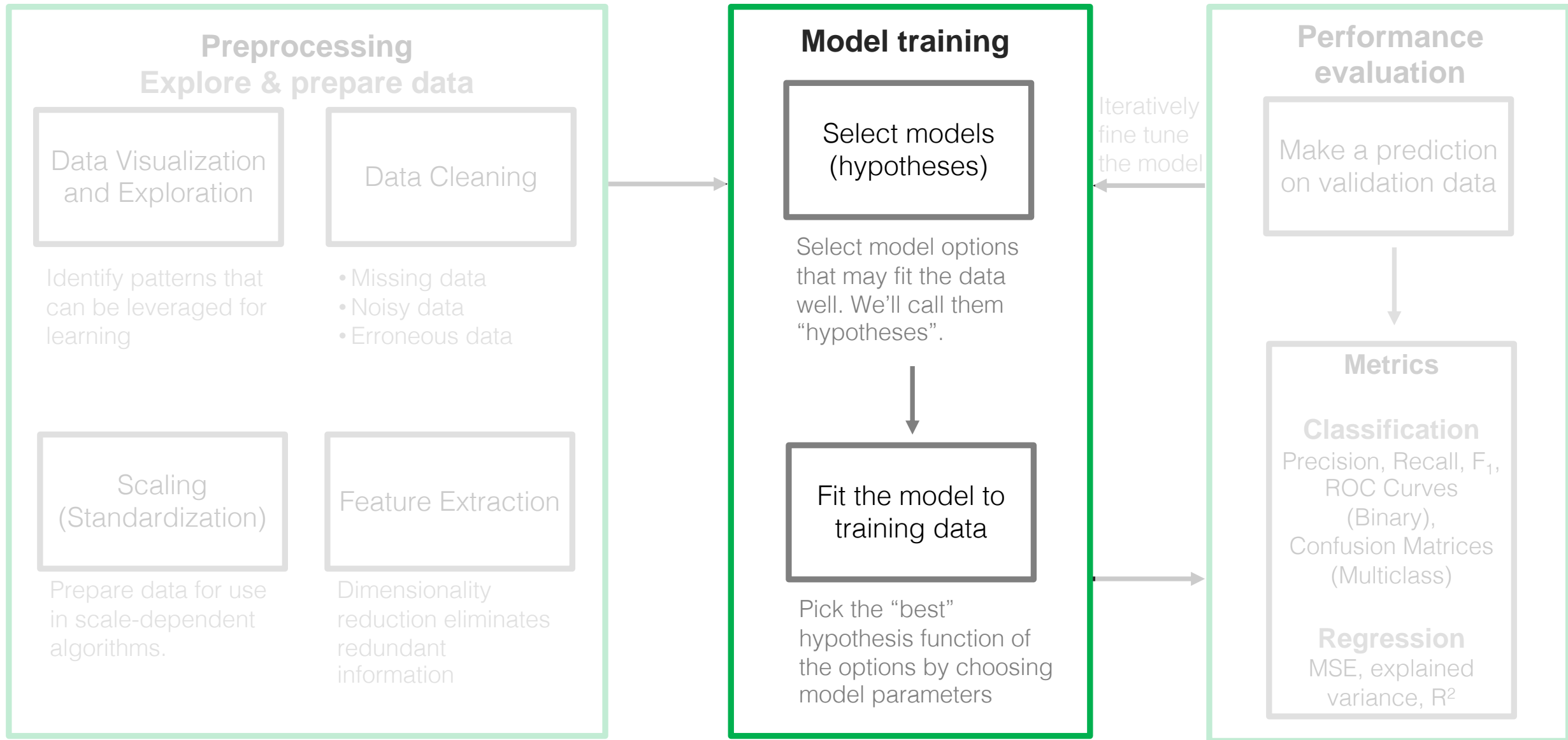
Supervised learning in practice



Supervised learning in practice



Supervised learning in practice



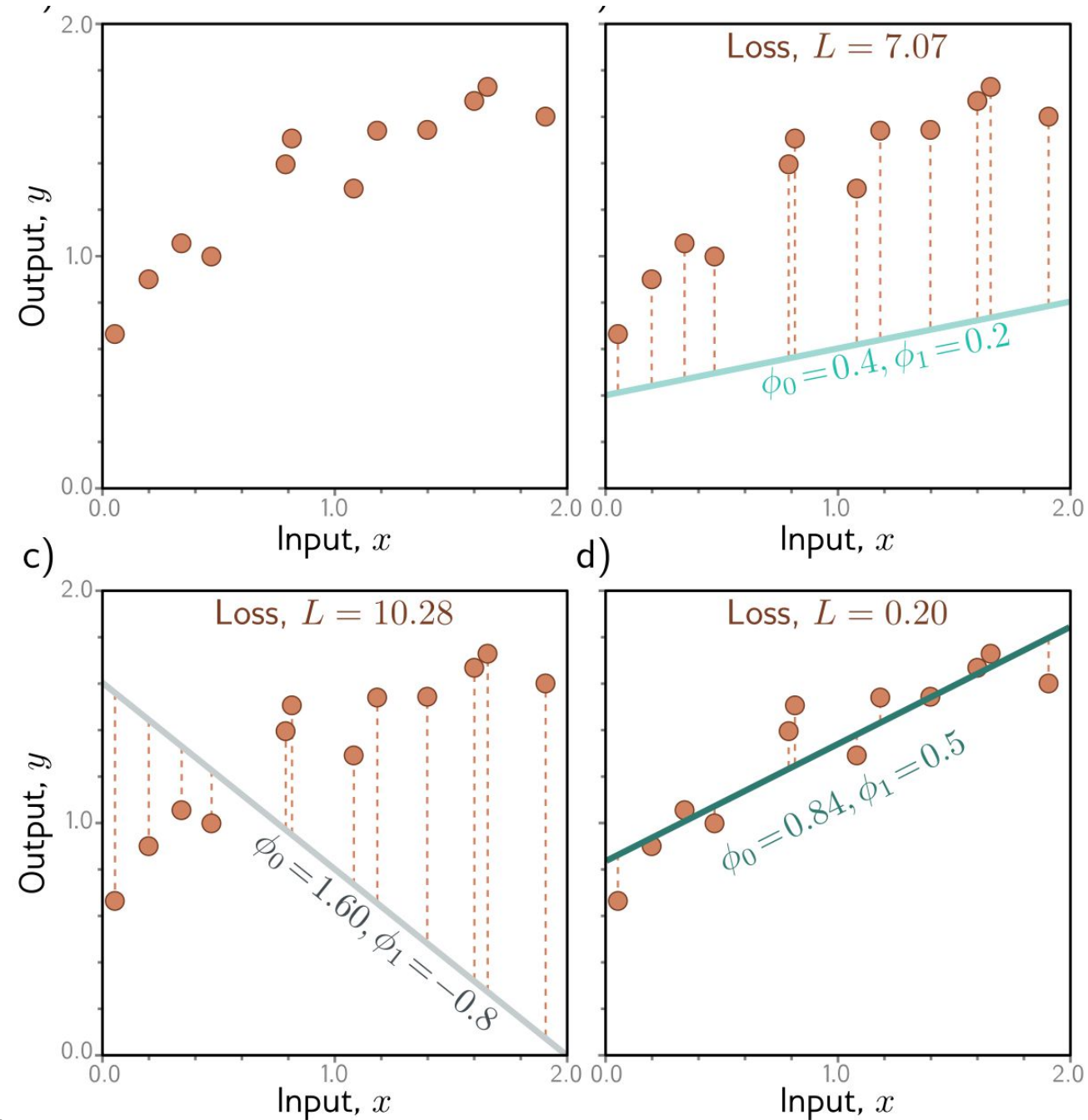
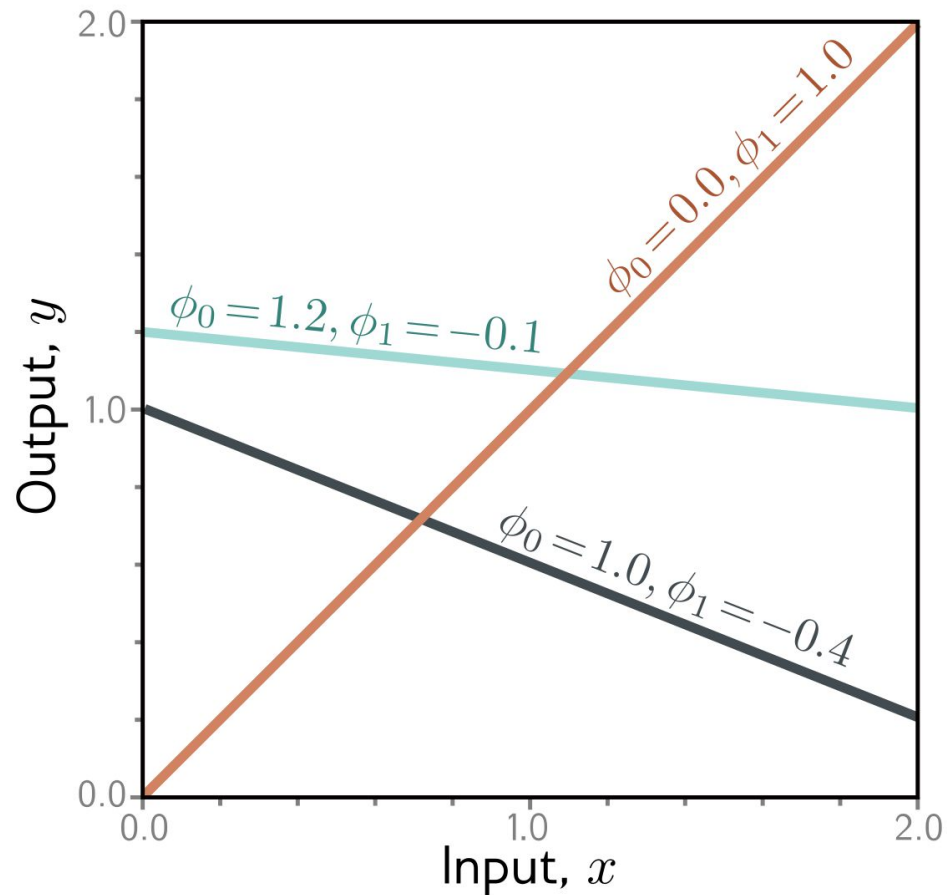
Model Fitting / Training Process

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)

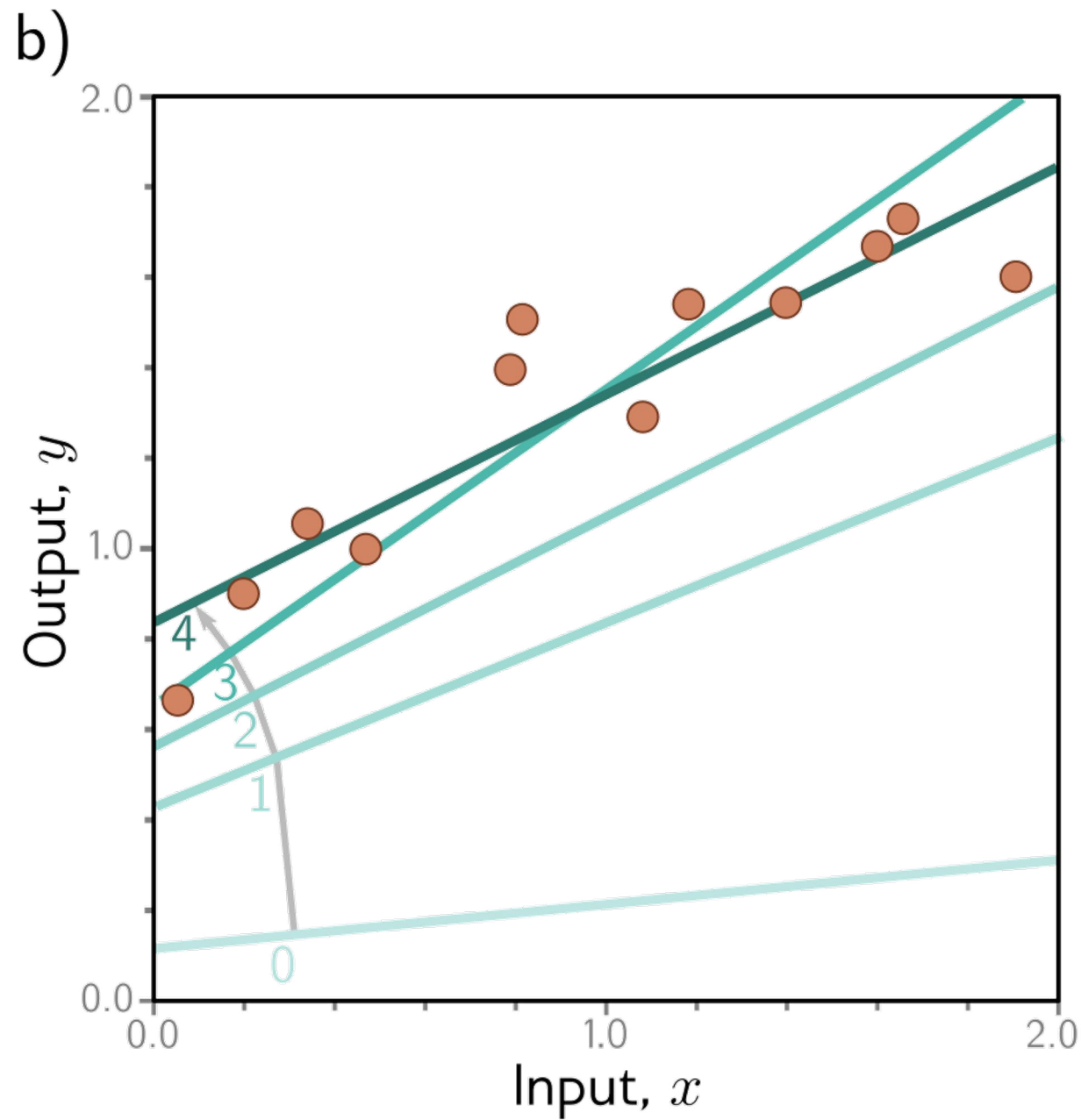
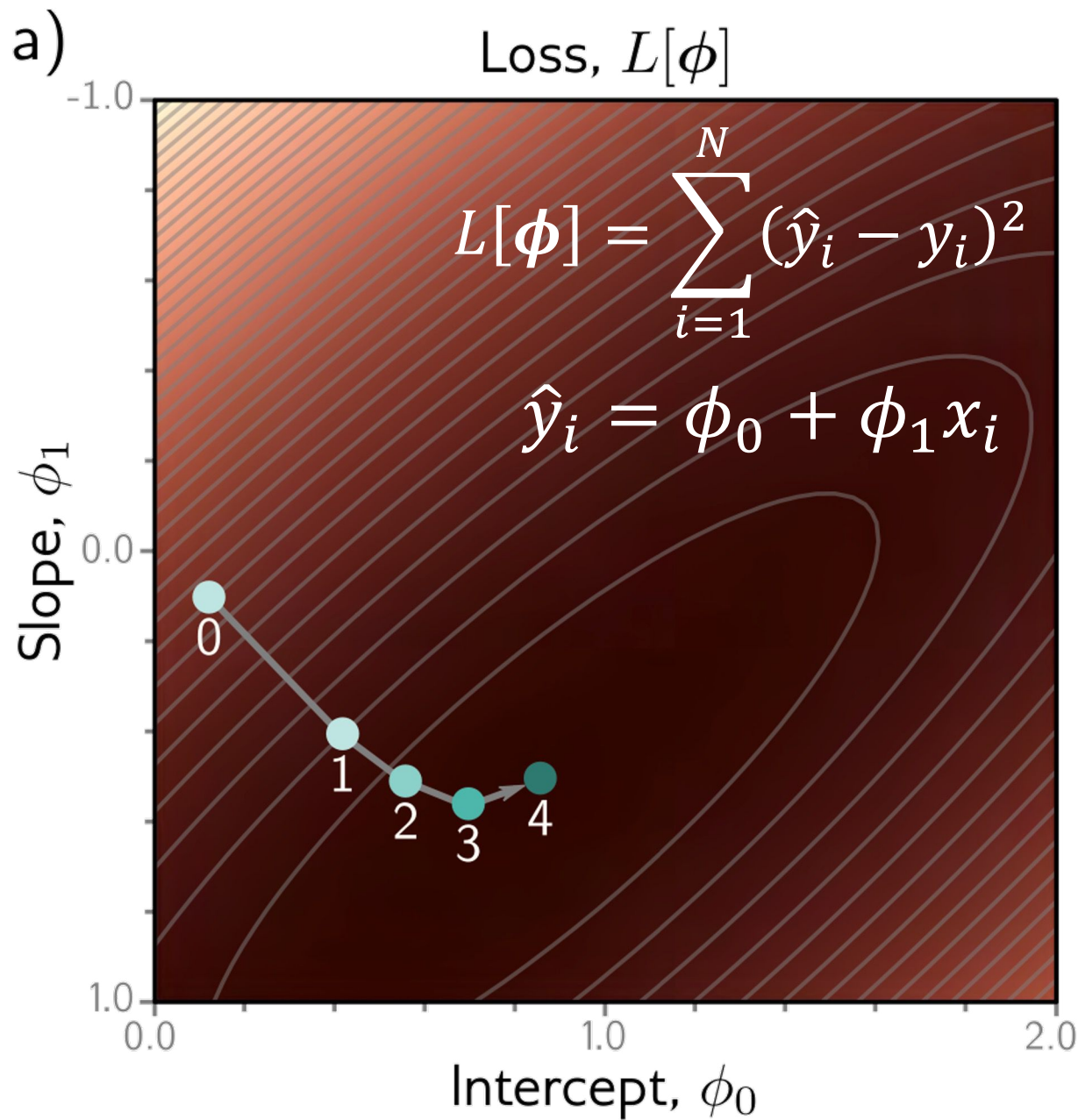
★ We will use this procedure for ALL the parametric models we encounter
Parametric models = models where the number of parameters are fixed and independent of the training data size

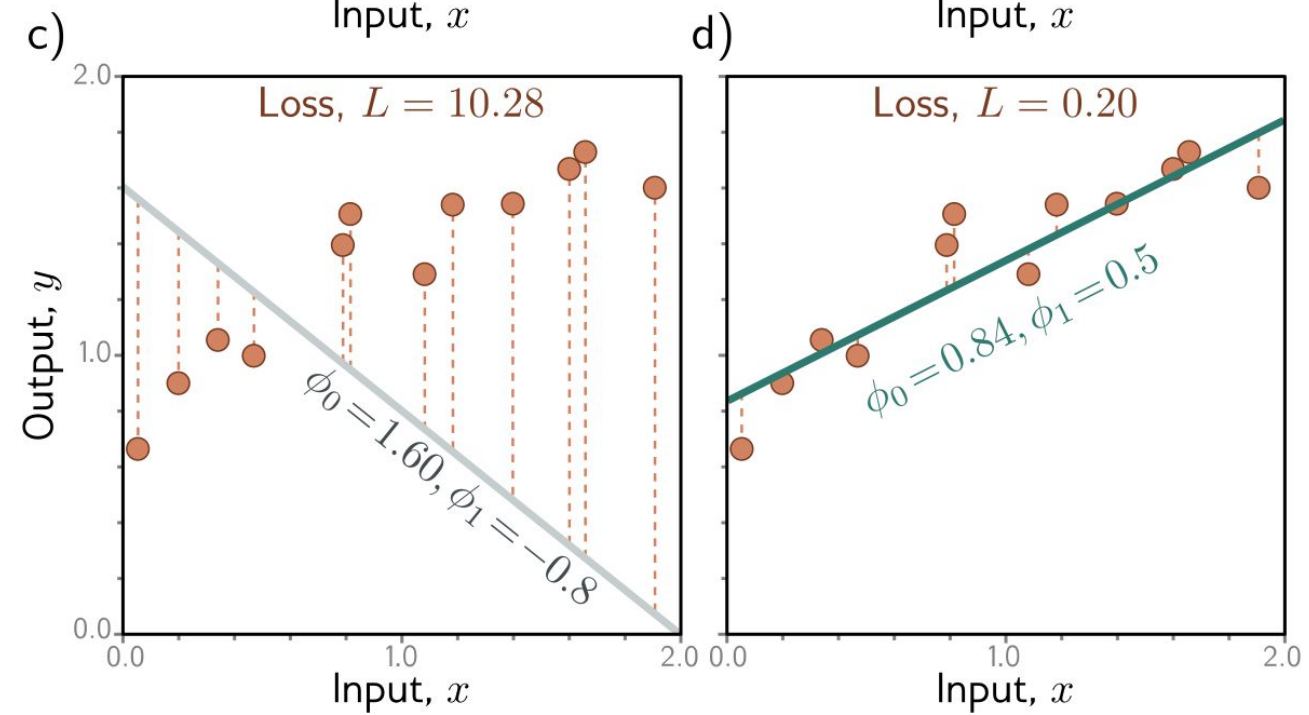
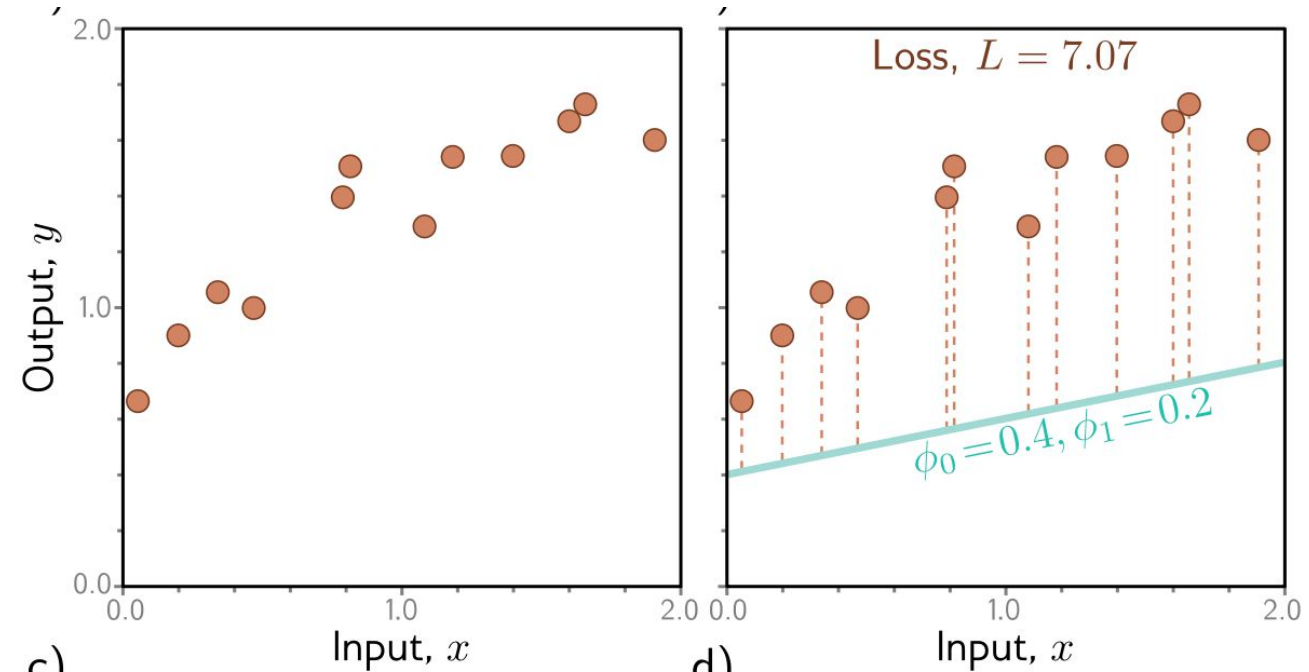
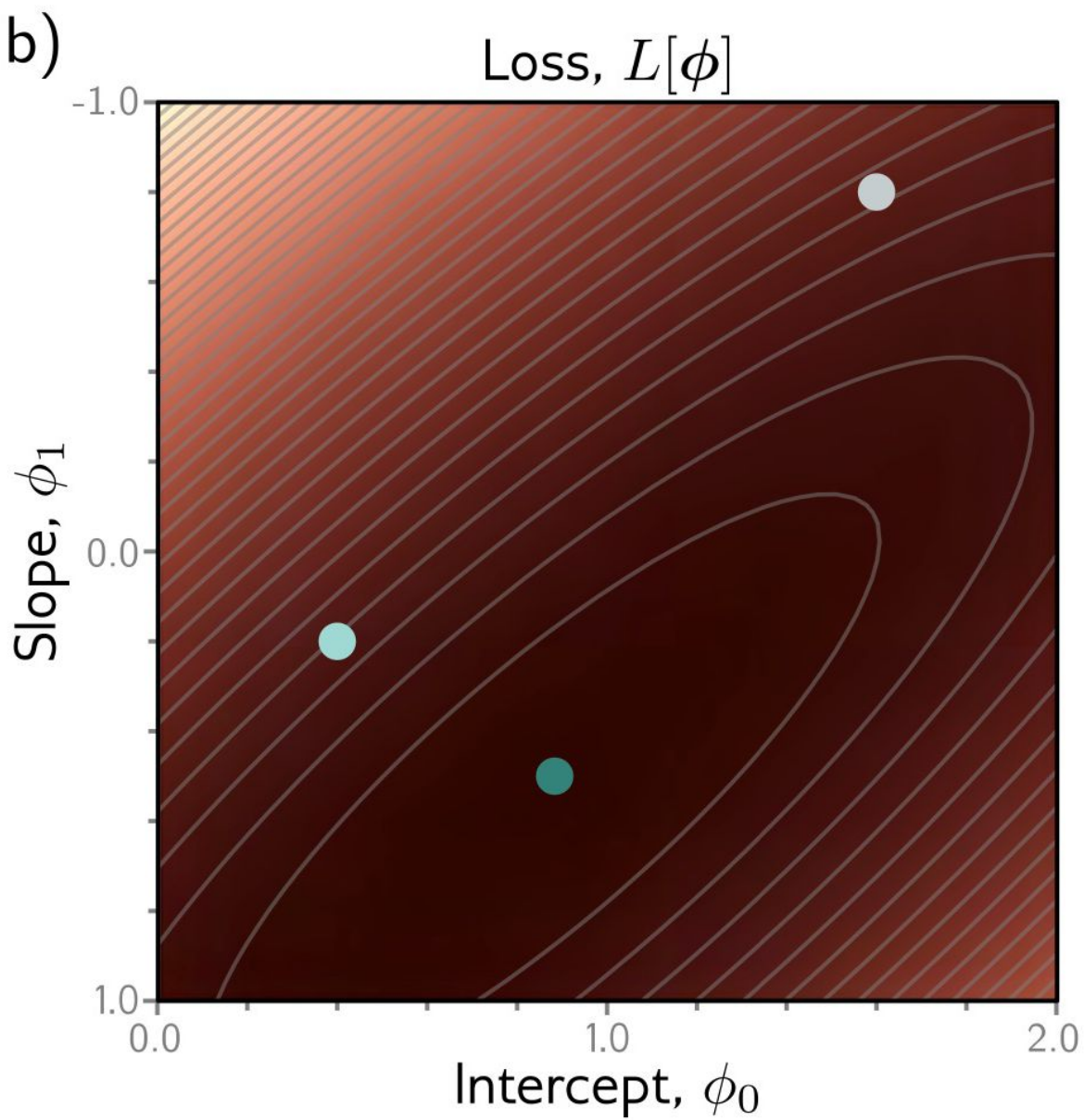
Simple Linear Regression Model Fitting Process

$$\hat{y} = \phi_0 + \phi_1 x$$



Images from Prince, S.J., 2023. *Understanding Deep Learning*. MIT press.





Images from Prince, S.J., 2023. *Understanding Deep Learning*. MIT press.

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 $y = w_0 + w_1x_1$

C $y = w_0 + w_1x_1 + x_2^{w_2}$

D $y = w_0 + w_1x_1^2 + w_2x_2^{0.4}$

E $y = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2$

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

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)

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

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

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

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

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

Linear regression model


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

Parameters w_j

Target y_i

Features $x_{i,j}$

Sample Feature



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

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 $y = w_0 + w_1x_1$

C $y = w_0 + w_1x_1 + x_2^{w_2}$

D $y = w_0 + w_1x_1^2 + w_2x_2^{0.4}$

E $y = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2$

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

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

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:

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

F

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

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

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	<div>1</div> <div>$x_{i,j}$</div>	<div><div>Shorthand</div>x_i</div> <div>$x_i = \begin{bmatrix} x_{i,0} \\ x_{i,1} \\ \vdots \\ x_{i,p} \end{bmatrix}$</div>
	<div>N</div> <div>$x = \begin{bmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{N,j} \end{bmatrix}$</div>	<div><div>Shorthand</div>X</div> <div>$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,p} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,0} & x_{N,1} & \cdots & x_{N,p} \end{bmatrix}$</div>

Assume $x_{i,0} = 1$

$$x_i = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{bmatrix}$$

Types of Linear Regression

Here, $x_0 \triangleq 1$

	One feature variable	Two or more feature variables
One target variable	Simple Linear Regression $y_i = w_0 + w_1 x_{i,1}$	Multiple Linear Regression $y_i = \sum_{j=0}^p w_j x_{i,j} \text{ or } y_i = \mathbf{w}^T \mathbf{x}_i$
Two or more target variables	Multivariate (Multiple) Linear Regression $q = \# \text{ target variables}$ $[q \times 1]$ $\mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i$ $[q \times p] [p \times 1]$ $p = \# \text{ features (includes } x_0 = 1)$	

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

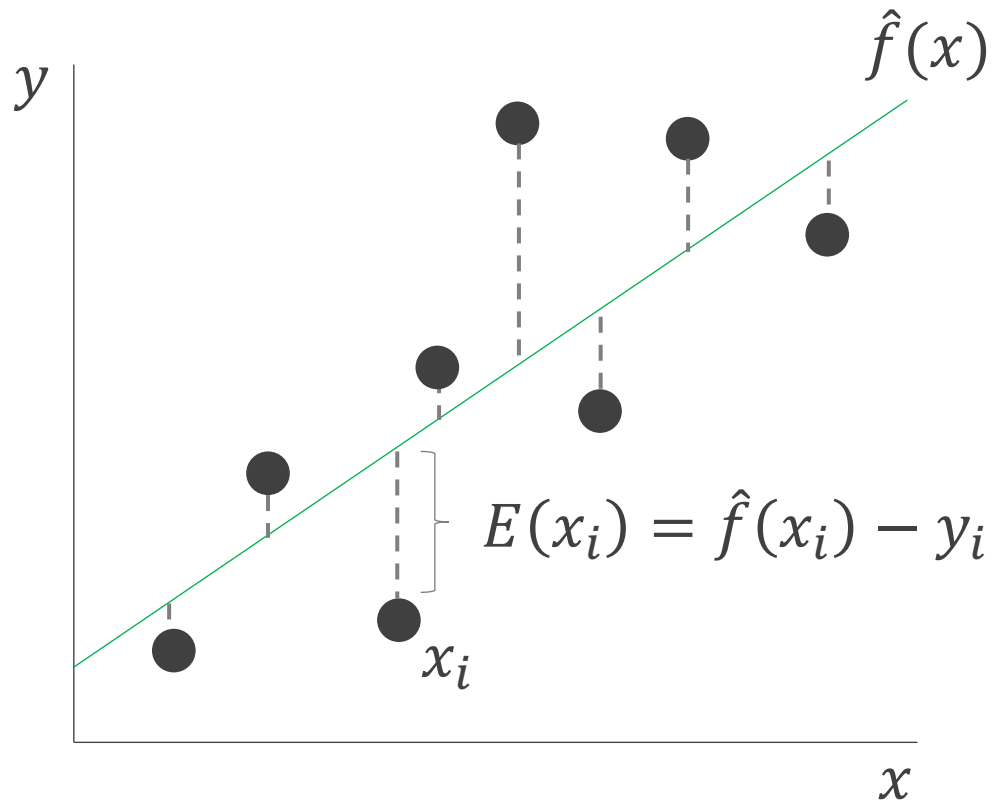
For many applications with complex feature-target relationships, underfits

Requires feature engineering for capturing more complex feature-target relationships

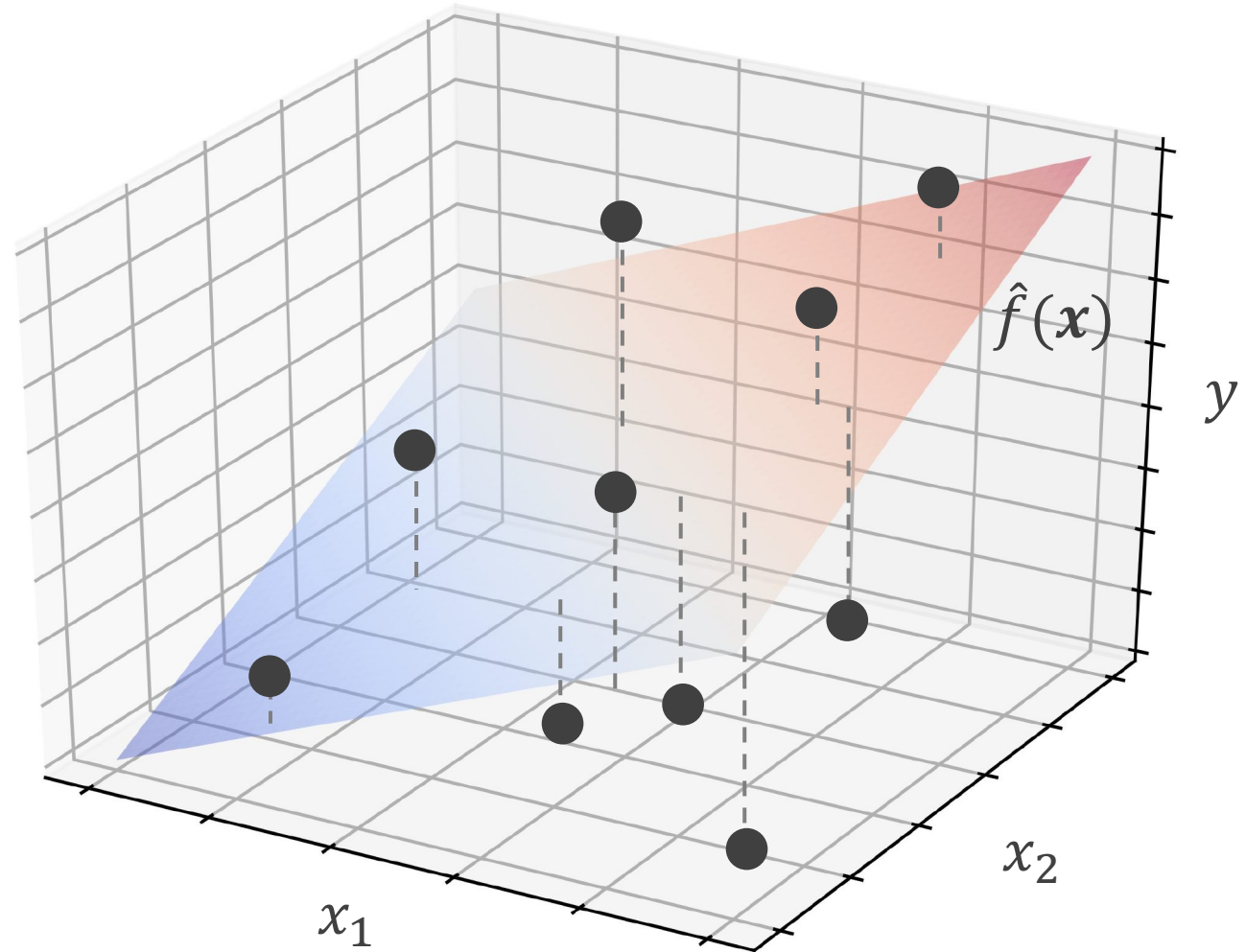
How do we fit the model to the training data?

A winding path to the least squares solution

Linear models and error



simple linear regression



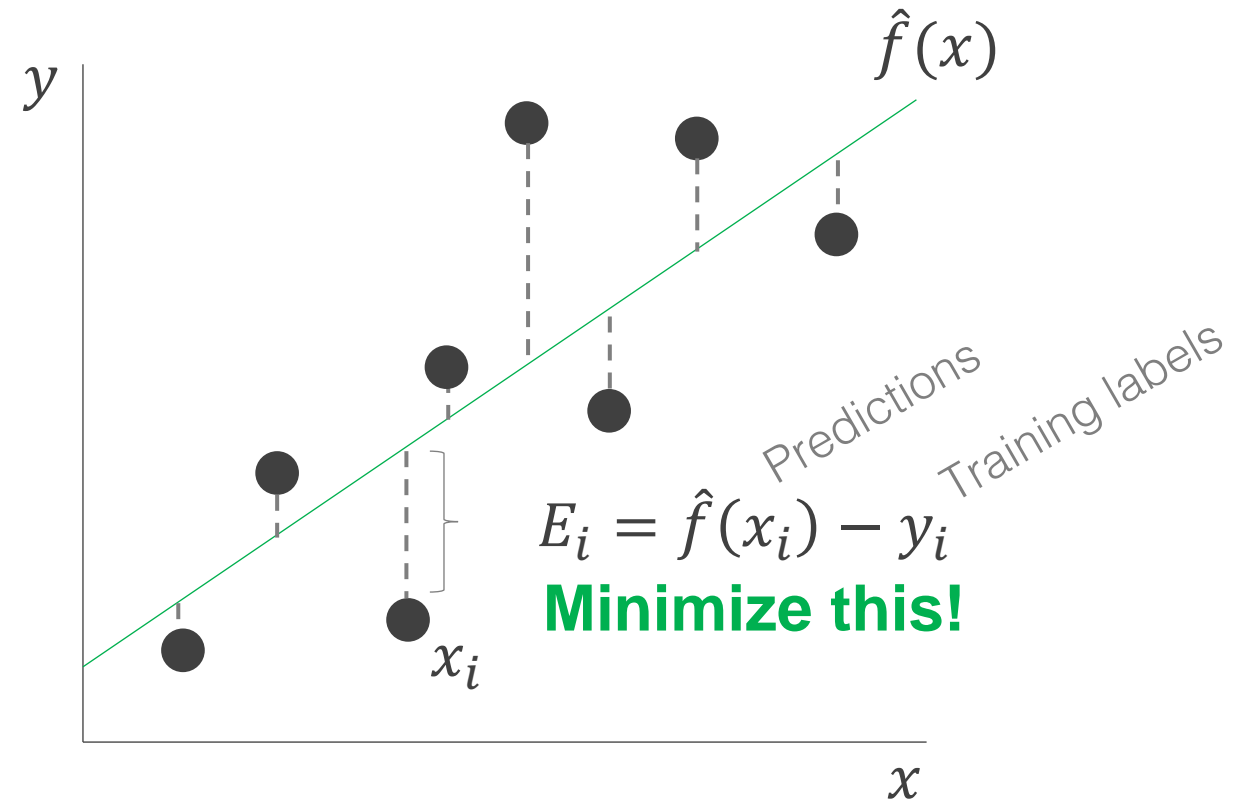
multiple linear regression

How do we fit a linear model to training data?

Training data:

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

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



How do we measure error?

The error for one sample is:

$$E_i = \hat{f}(\mathbf{x}_i) - y_i \qquad \hat{y}_i = \hat{f}(\mathbf{x}_i) = \sum_{j=0}^p w_j x_{i,j}$$

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

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

Why mean squared error? See *Understanding Deep Learning* Chapter 5 (particularly 5.3)

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

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 \mathbf{w} to minimize cost (error)

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

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

We want to minimize

...by varying \mathbf{w}

How do we do that?

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

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



Calculus

A moment of calculus

Function of one variable

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

Derivative

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

Function of multiple variables

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

Partial Derivative

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

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

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

Gradient

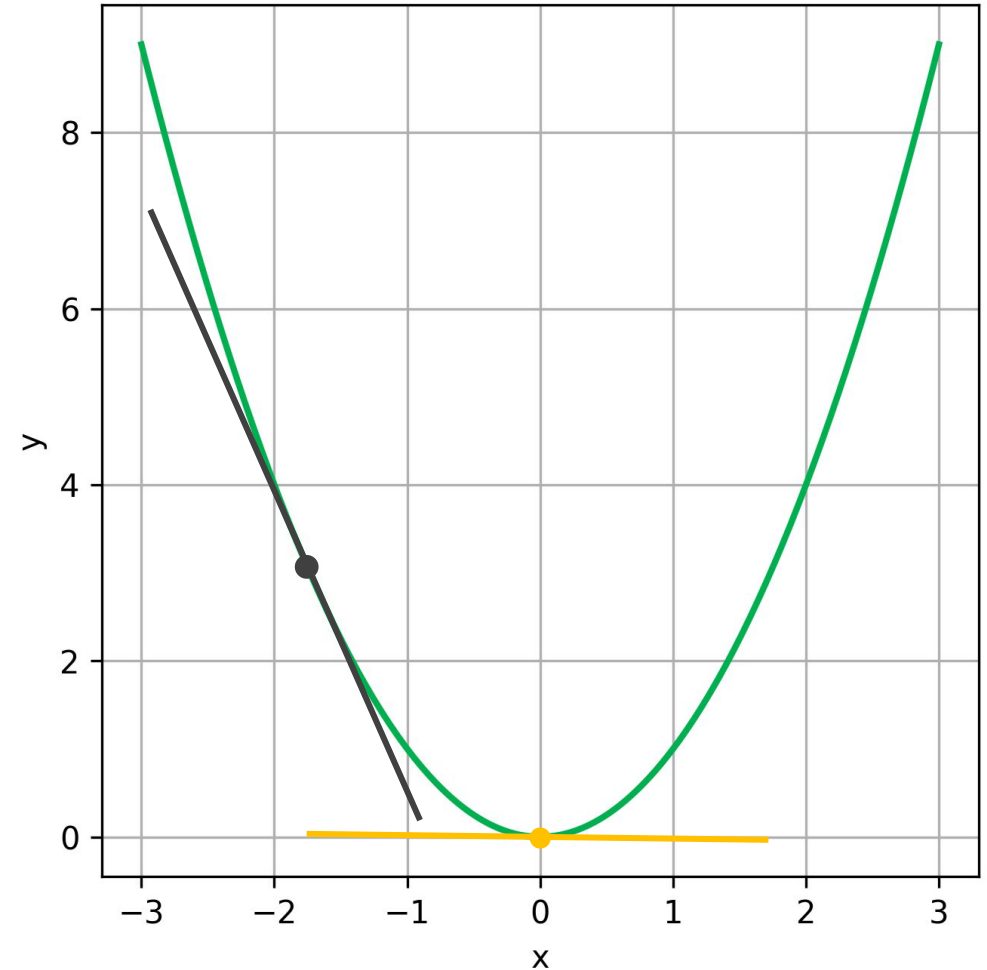
$$\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

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

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 \mathbf{w} , set it to zero, and solve for \mathbf{w}
(think derivative)

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

p = number of predictors
 N = number of data points

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

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

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...

$$E_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (\underbrace{\mathbf{w}^T \mathbf{x}_n}_{\text{Scalar}} - y_n)^2$$

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

...can be rewritten as:

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

Convenient definitions:

$$\begin{aligned} \mathbf{y} &\in \mathbb{R}^{N \times 1} \\ \mathbf{X} &= \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times p+1} \\ \mathbf{w} &= \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1 \times 1} \end{aligned}$$

p = number of predictors
 N = number of data points

1

Assume $p = 2$
 $N = 4$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$$\mathbf{x}_i = \begin{bmatrix} x_{i,0} \\ x_{i,1} \\ x_{i,2} \end{bmatrix}$$

p = number of predictors
 N = number of data points

2

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

3

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

4

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

Aside on
algebraic
manipulations

4

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

5

$$X\mathbf{w} - \mathbf{y} = \begin{bmatrix} \mathbf{w}^T \mathbf{x}_1 - y_1 \\ \mathbf{w}^T \mathbf{x}_2 - y_2 \\ \mathbf{w}^T \mathbf{x}_3 - y_3 \\ \mathbf{w}^T \mathbf{x}_4 - y_4 \end{bmatrix}$$

6

$$(\mathbf{w}^T \mathbf{x}_1 - y_1 \quad \mathbf{w}^T \mathbf{x}_2 - y_2 \quad \mathbf{w}^T \mathbf{x}_3 - y_3 \quad \mathbf{w}^T \mathbf{x}_4 - y_4) \begin{bmatrix} \mathbf{w}^T \mathbf{x}_1 - y_1 \\ \mathbf{w}^T \mathbf{x}_2 - y_2 \\ \mathbf{w}^T \mathbf{x}_3 - y_3 \\ \mathbf{w}^T \mathbf{x}_4 - y_4 \end{bmatrix}$$

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

**Aside on
algebraic
manipulations**

How to fit our model to the training data?

Our cost function...

$$E_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (\underbrace{\mathbf{w}^T \mathbf{x}_n}_{\text{Scalar}} - y_n)^2$$

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

...can be rewritten as:

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

Convenient definitions:

$$\mathbf{y} \in \mathbb{R}^{N \times 1}$$
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times p+1}$$
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1 \times 1}$$

p = number of predictors
 N = number of data points

How to fit our model to the training data?

$$E_{in}(\mathbf{w}) = \frac{1}{N} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) \quad \text{(take gradient, set to 0)}$$

$$\nabla_{\mathbf{w}} E_{in}(\mathbf{w}) = \frac{2}{N} (\mathbf{X}^T \mathbf{X}\mathbf{w} - \mathbf{X}^T \mathbf{y}) = \mathbf{0} \quad \text{(solve for } \mathbf{w} \text{)}$$

Univariate analogy:

$$\begin{aligned} f(w) &= \frac{1}{N} (xw - y)^2 \\ &= \frac{1}{N} (x^2 w^2 - 2xyw + y^2) \\ \frac{df(w)}{dw} &= \frac{2}{N} (x^2 w - xy) \end{aligned}$$

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

$$\mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{y} \quad \text{(normal equation)}$$

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

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

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

What is the pseudoinverse?

Samples and features impact solutions

$$\begin{array}{c} \text{Features} \\ \left[\begin{array}{c} N \times p \end{array} \right] \begin{array}{c} \left[\begin{array}{c} p \times 1 \end{array} \right] = \left[\begin{array}{c} N \times 1 \end{array} \right] \\ \text{Samples} \end{array} \\ \\ \mathbf{X} \quad \mathbf{w} = \mathbf{y} \end{array}$$

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 \mathbf{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

$$w_1 = 2$$

$$w_2 = 1$$

Overdetermined system

equations > # unknowns

$$w_1 = 2$$

$$w_2 = 1$$

$$w_2 = 5$$

Example 2

Fully determined system

equations = # unknowns

$$2w_1 + 3w_2 = 2$$

$$6w_1 + 1w_2 = 0.63$$

Overdetermined system

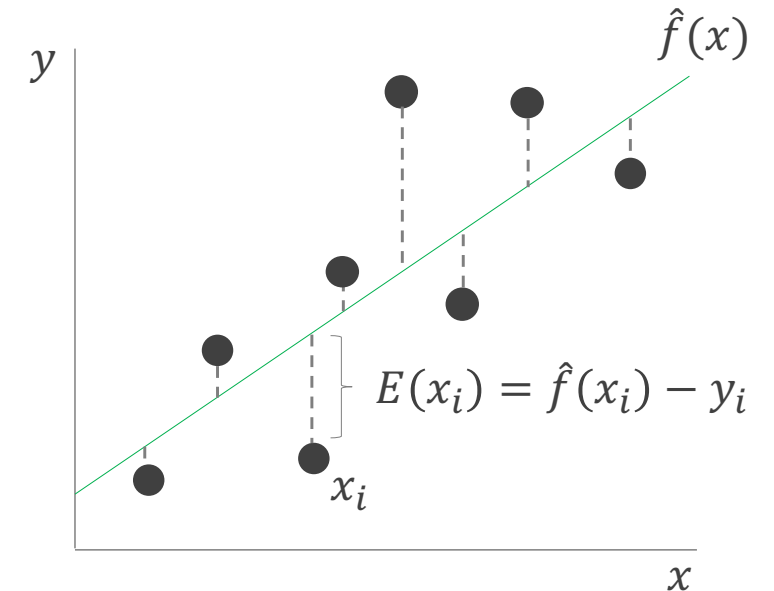
equations > # unknowns

$$2w_1 + 3w_2 = 2$$

$$6w_1 + 1w_2 = 0.63$$

$$3w_1 + 2w_2 = 14$$

$$16w_1 - w_2 = 0.1$$



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

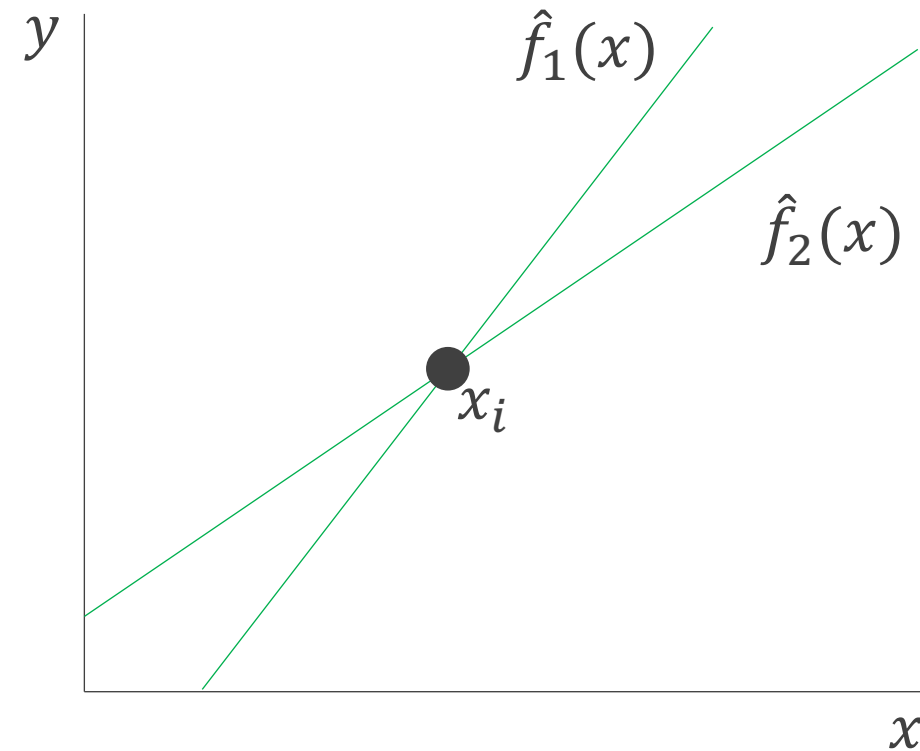
$$2w_1 + 3w_2 = 2$$

$$6w_1 + 1w_2 = 0.63$$

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)

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

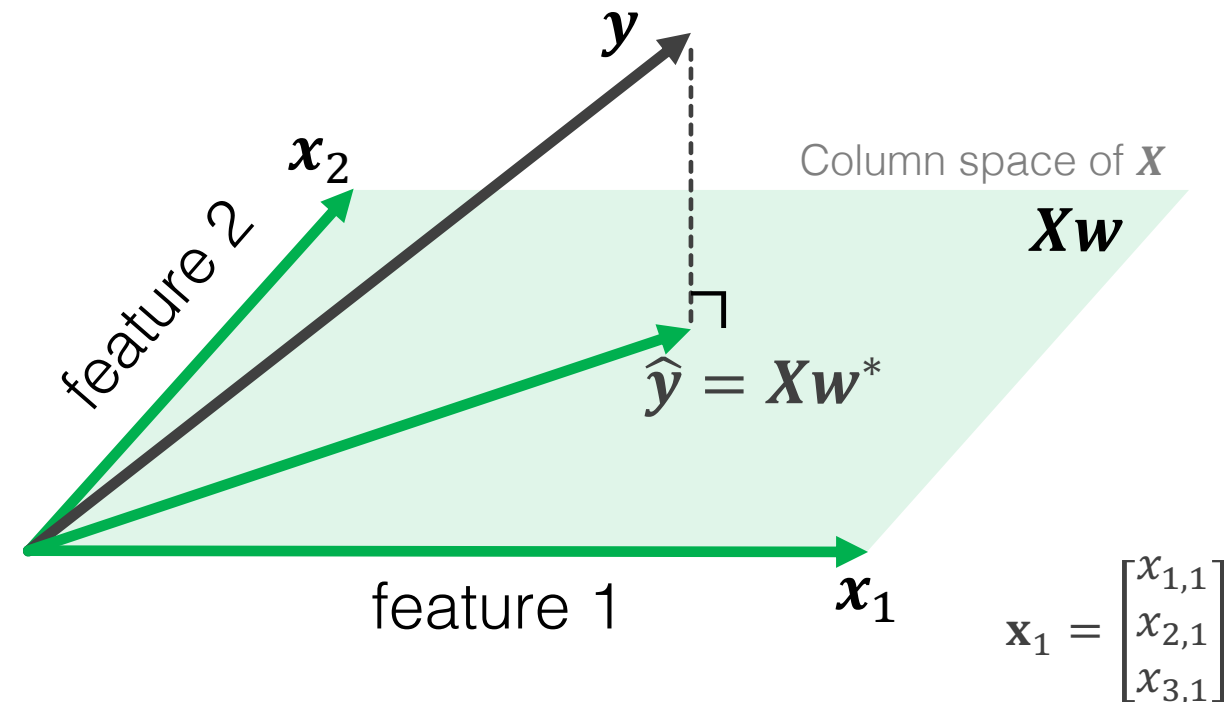
$$\begin{matrix} & \text{Features} \\ \text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \end{matrix}$$

Feature vectors: \mathbf{x}_1 \mathbf{x}_2

$$\mathbf{X} \mathbf{w} \neq \mathbf{y}$$

We CAN use \mathbf{X}^\dagger to obtain \mathbf{w}^* , the least squares solution:

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



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

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

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