Laboratory 2
Parametric models

1. Linear Regression

- **linear regression** method predicts a real-valued output (*label*/*target*) $y \in \mathbb{R}$, given a set of real-valued inputs (*features*) $x_1, x_2, \ldots, x_d$;
- the relationship between the independent variables $x_1, x_2, \ldots, x_d$ and the dependent variable y is *linear*;
- to develop a model for predicting $y \in \mathbb{R}$, we need to obtain a *dataset* consisting of known outputs for corresponding inputs;
- the dataset is called a *training dataset/set*, and each row is called *example/data point/sample*;
- the linear regression model is expressed as:

$$\underset{\substack{\uparrow \\ \text{predicted} \\ \text{value}}}{\hat{y}} = \underset{\substack{\uparrow \\ \textit{bias}}}{w_0} + \underset{\substack{\uparrow \\ \text{feature} \\ \text{weight}}}{w_1} \overset{\substack{\textit{feature} \\ \downarrow}}{x_1} + \cdots + \underset{\substack{\uparrow \\ \text{feature} \\ \text{weight}}}{w_d} \overset{\substack{\textit{feature} \\ \downarrow}}{x_d}.$$

1. Linear Regression

- the weights determine the influence of each feature on our prediction;
- given a dataset, our goal is to choose the weights $w_1, w_2, \ldots, w_d$ and the bias w_0 such that, on average, the predictions $\hat{y}$ made by our model *best fit* the true labels y observed in the data;

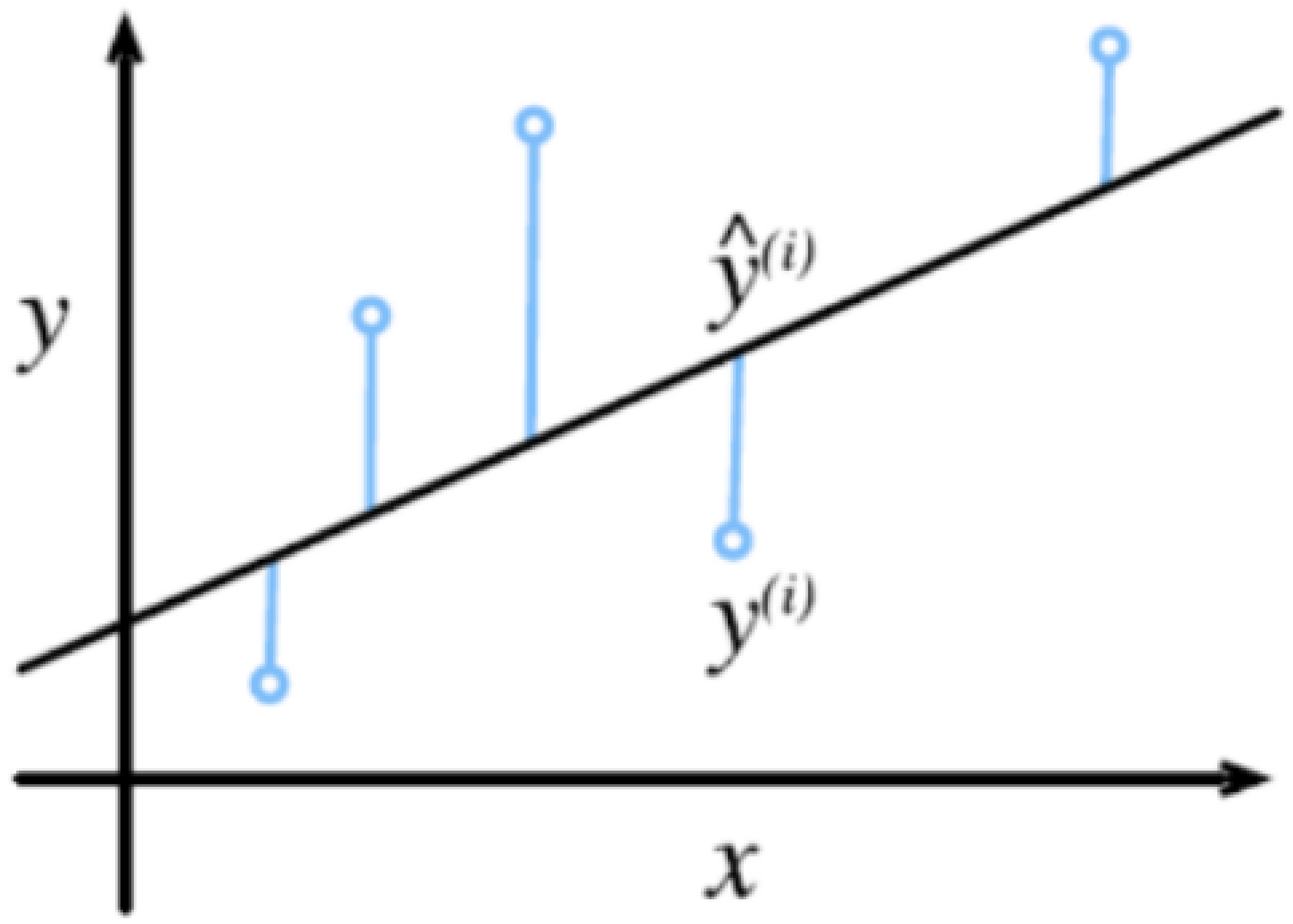


Figure 1: Linear regression with $d = 1$.

- **normal equation**: $\mathbf{w}^* = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$.

1. Linear Regression

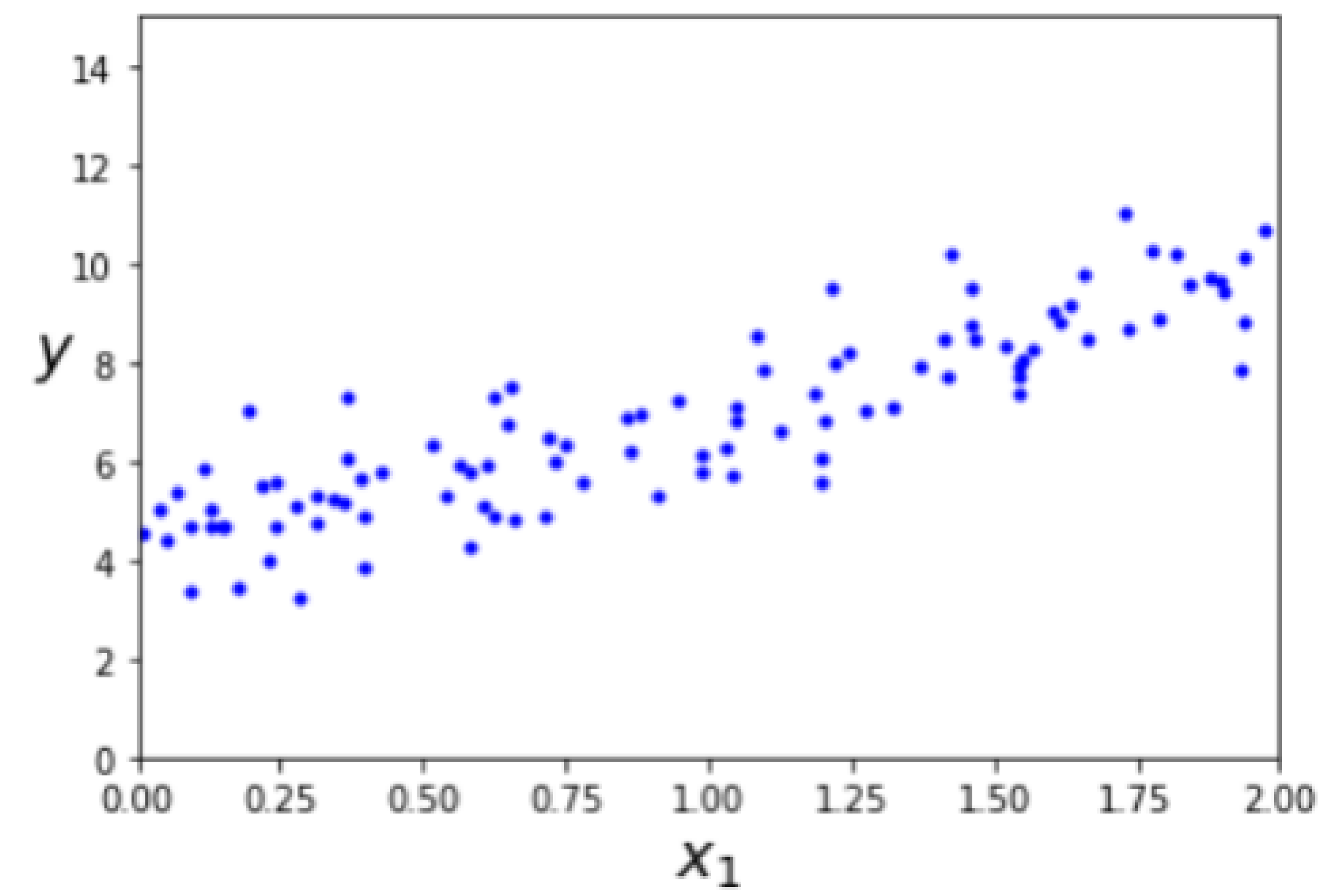
- generate data ($y = 4 + 3x_1 +$ *Gaussian noise*) and compute w^* using normal equation;

```
X = 2 * np.random.rand(100, 1) # d=1
y = 4 + 3 * X + np.random.randn(100, 1)

X_1 = np.c_[np.ones((100, 1)), X] # add x0 = 1 to each instance
w_star = np.linalg.inv(X_1.T.dot(X_1)).dot(X_1.T).dot(y)

w_star
```

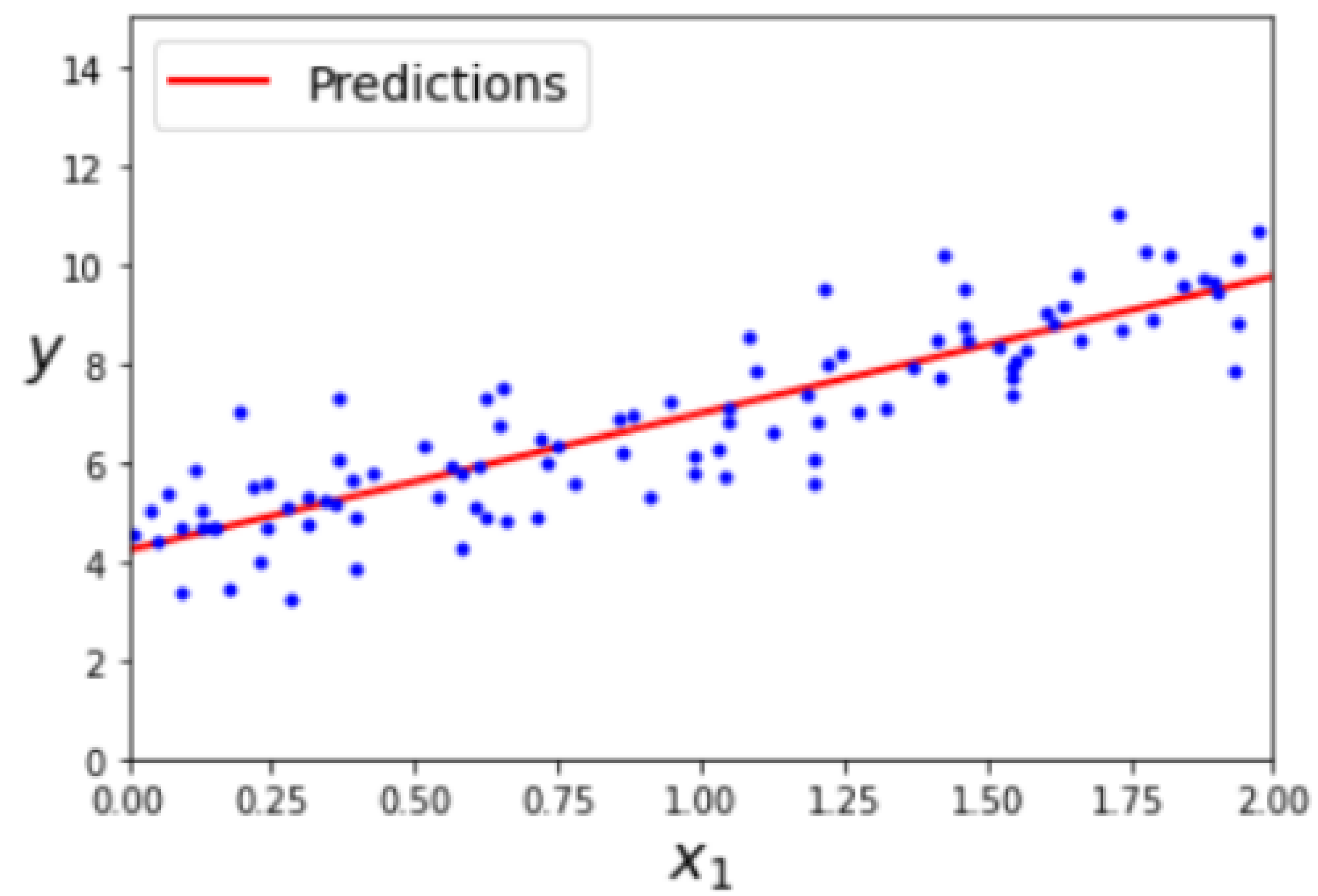
```
array([[4.21509616],
       [2.77011339]])
```



- generate predictions using w^*;

```
X_new = np.array([[0], [2]])
X_new_1 = np.c_[np.ones((2, 1)), X_new]  # add x0 = 1 to each instance
y_predict = X_new_1.dot(w_star)
y_predict
```

```
array([[4.21509616],
       [9.75532293]])
```



6. Softmax Regression

3. **Prediction**: the class with the highest probability is the output class;

$$\hat{y} = \underset{k}{argmax}\, \hat{p}_k.$$

- training softmax regression involves minimizing a loss function, called the *cross entropy* loss, that captures the difference between predicted probabilities and the actual class labels;

$$\mathcal{L}(\mathbf{W}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{q} y_k^{(i)} \log \hat{p}_k^{(i)},$$

where n denotes the number of examples in the dataset.

6. Softmax Regression

- classify the iris flowers into the three classes:

```
softmax_reg = LogisticRegression(max_iter=1000, multi_class="multinomial", solver="lbfgs", C=10, random_state=42)
softmax_reg.fit(X_train, y_train)
softmax_reg.score(X_test, y_test)
```

  - multi_class="multinomial" $\rightarrow$ softmax regression;
  - solver="lbfgs" $\rightarrow$ variant of stochastic gradient descent;
  - ℓ_2 regularization, $C = \frac{1}{\lambda}$;