Regression and Classification

Linear Regression

Regression model:

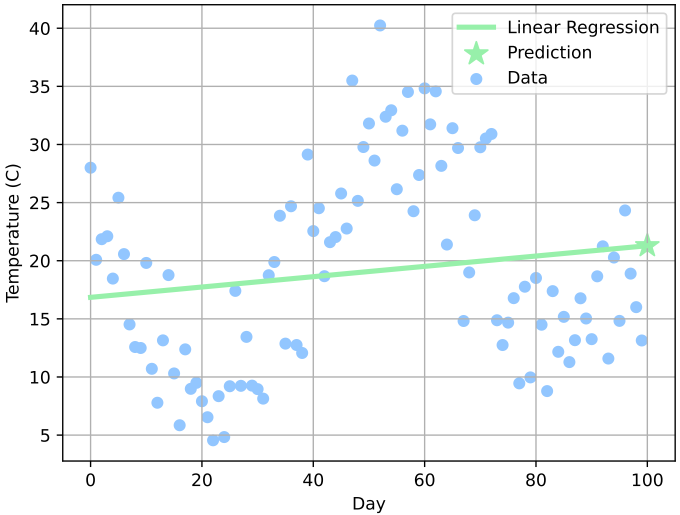
$$f_\theta : \mathbb{R}^n \to \mathbb{R}^d$$

Linear regression:

$$f_\theta(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Parameters:

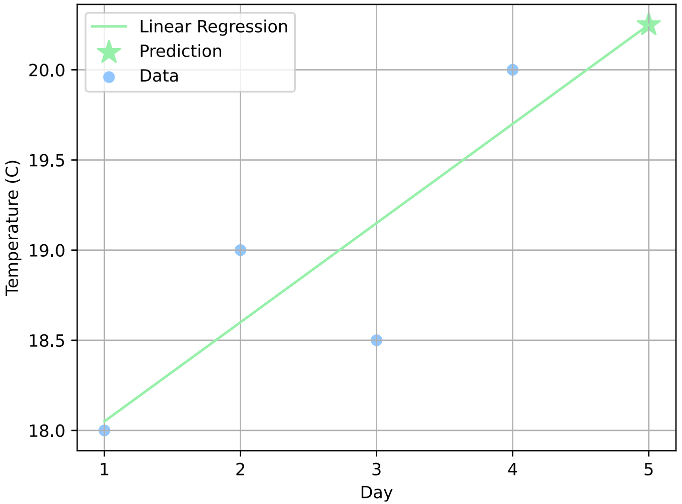
$$\theta = (\mathbf{W}, \mathbf{b})$$



Linear Regression: Example

Temperature forecast:

$f(x)$: average temperature on day x



Linear Regression in PyTorch

Notebook Demo

Linear Regression in PyTorch

Define a linear regression model:

```
linear = torch.nn.Linear(4, 2)
print(f"{linear.weight=}")
print(f"{linear.bias=}")

x = torch.as_tensor([1, 2, 3, 4], dtype=torch.float32)
print(f"{linear(x)=}")
```

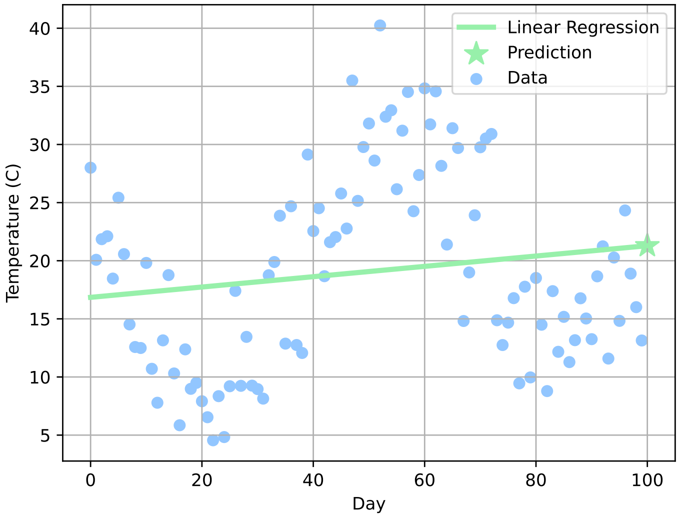
Output:

```
linear.weight=Parameter containing:
tensor([[-0.1368, -0.0667,  0.1794, -0.2696],
        [-0.3077,  0.4574, -0.0037, -0.4138]], requires_grad=True)
linear.bias=Parameter containing:
tensor([-0.1660, -0.4015], requires_grad=True)
linear(x)=tensor([-0.9764, -1.4608], grad_fn=<AddBackward0>)
```

Linear Regression: Limitation

Cannot deal with non-linear patterns:

- cyclic functions
- quadratic functions
- ...



Linear Binary Classification

Binary classification model:

$$f_\theta : \mathbb{R}^n \to [0, 1]$$

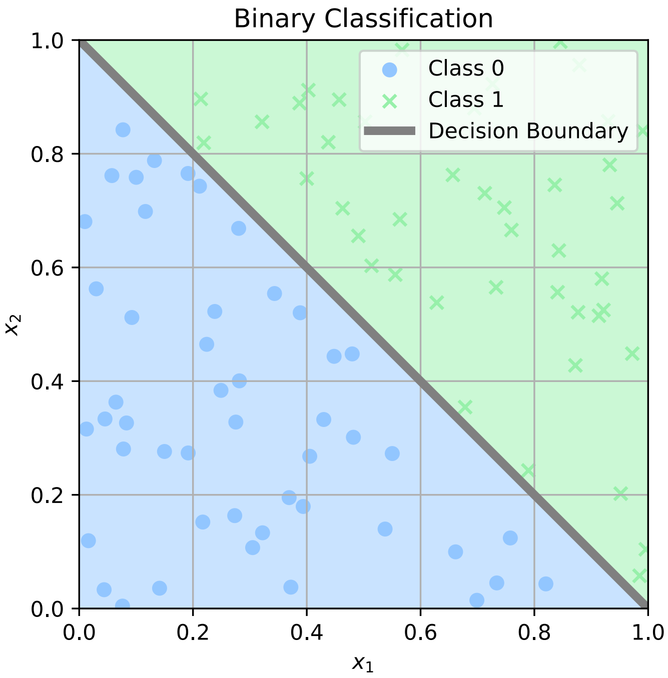
Linear binary classification:

$$f_\theta(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

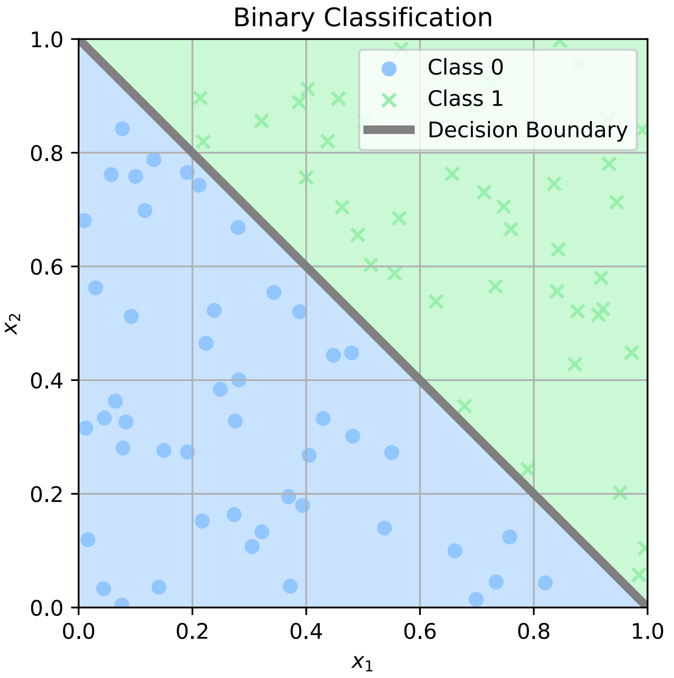
Parameters:

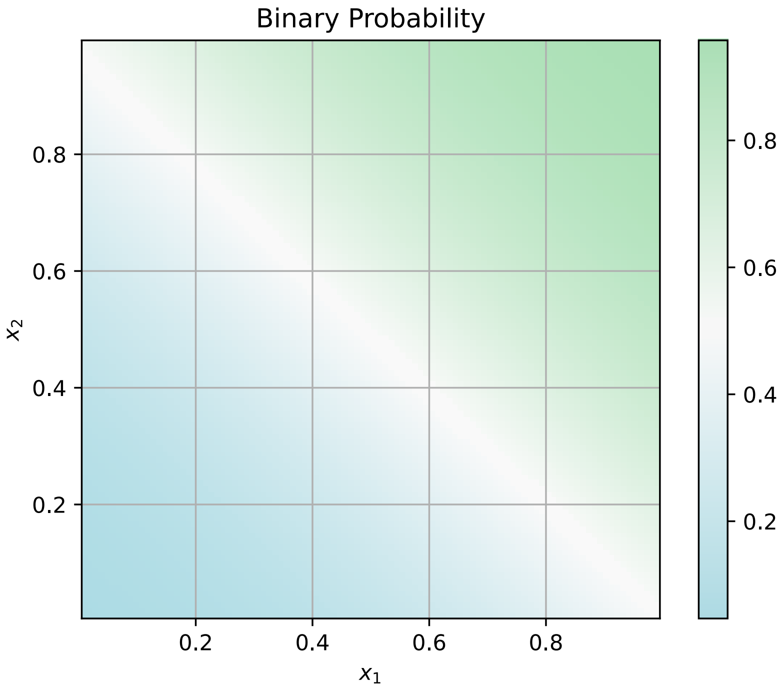
$$\theta = (\mathbf{W}, \mathbf{b})$$



Linear Binary Classification: Decision Boundary

$$\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$





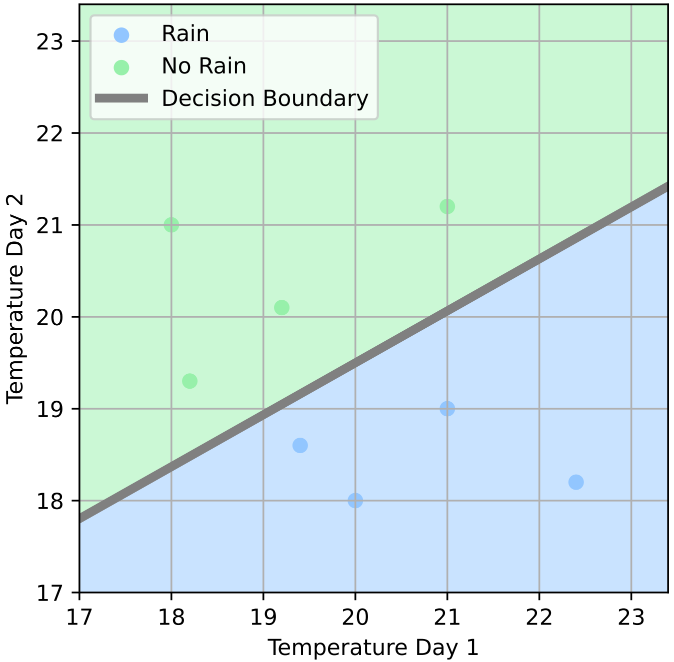
Linear Binary Classification: An Example

Input x: average daily temperature

Output $f(x)$: whether it will rain on Wednesday

Prediction:

$$P(\text{rain}) = f_\theta(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$



Binary Classification in PyTorch:

Notebook Demo

Binary Classification in PyTorch

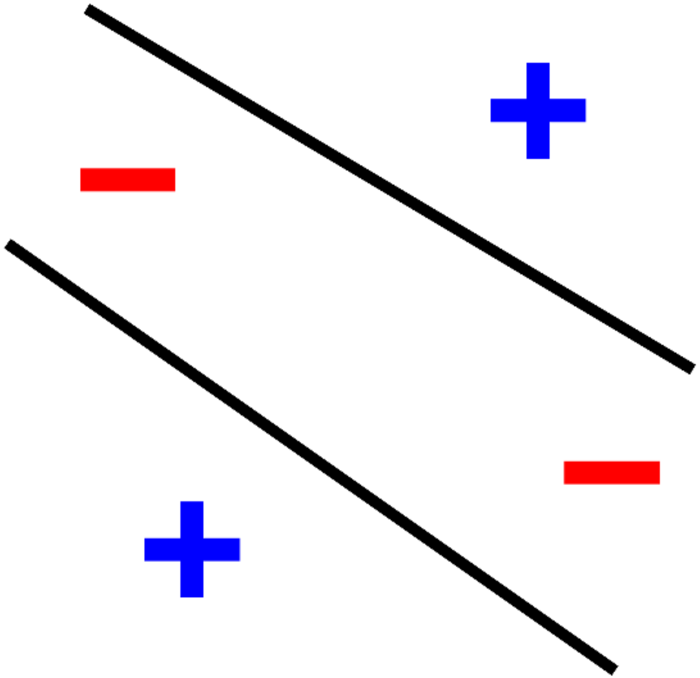
Define a binary classification model:

```
class BinaryClassifier(nn.Module):
    def __init__(self, input_size):
        super(BinaryClassifier, self).__init__()
        self.fc = nn.Linear(input_size, 1, bias=True)

    def forward(self, x):
        x = torch.sigmoid(self.fc(x))
        return x
```

Linear Binary Classification: Limitation

- Cannot deal with non-linear decision boundaries
- For deep learning, we will need to use non-linear models



Linear Multi-Class Classification

Multi-class classification model:

$$f_\theta : \mathbb{R}^n \to \mathbb{P}^c \quad \text{where} \quad \mathbb{P}^c \subset \mathbb{R}^c_+ \quad \forall_{\mathbf{y}\in\mathbb{P}^c} \mathbf{1}^\top \mathbf{y} = 1$$

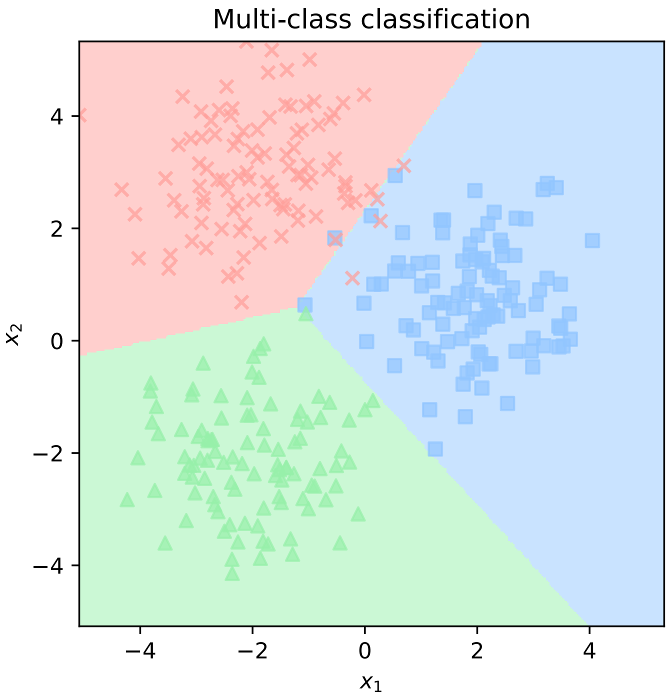
Linear multi-class classification:

$$f_\theta(\mathbf{x}) = \mathrm{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathrm{softmax}(\mathbf{v})_i = \frac{e^{v_i}}{\sum_{j=1}^n e^{v_j}}$$

Parameters:

$$\theta = (\mathbf{W}, \mathbf{b})$$



Softmax Function

For input $\mathbf{v} = \begin{bmatrix} v_1 \\ \dots \\ v_d \end{bmatrix} \in \mathbb{R}^d$, function $\mathrm{softmax} : \mathbb{R}^n \to \mathbb{P}^c$.

$$\mathbb{P}^c \subset \mathbb{R}^c_+ \quad \forall_{\mathbf{y} \in \mathbb{P}^c} \mathbf{1}^\top \mathbf{y} = 1$$

$$\mathrm{softmax}(\mathbf{v}) = \frac{1}{\sum_i e^{v_i}} \begin{bmatrix} e^{v_1} \\ \dots \\ e^{v_d} \end{bmatrix}$$

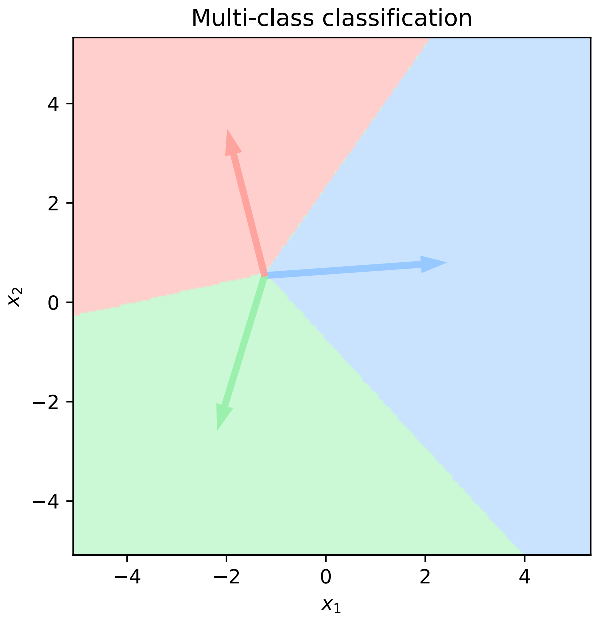
Linear Multi-Class Classification

Let $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \dots \\ \mathbf{w}_d^T \end{bmatrix}$ where $\mathbf{w}_j \in \mathbb{R}^n$

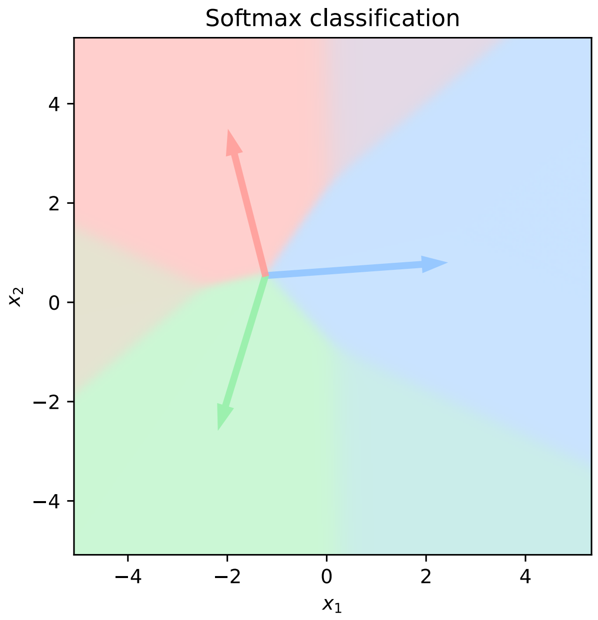
$$\begin{aligned} \mathrm{Classify}(\mathbf{x}) &= \arg\max_{j \in \{1,\dots,d\}} \mathrm{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_j && (1) \\ &= \arg\max_{j \in \{1,\dots,d\}} (\mathbf{W}\mathbf{x} + \mathbf{b})_j && (2) \\ &= \arg\max_{j \in \{1,\dots,d\}} \mathbf{w}_j^T \mathbf{x} + \mathbf{b}_j && (3) \end{aligned}$$

Linear Multi-Class Classification: Visualization

Hard decision boundary:



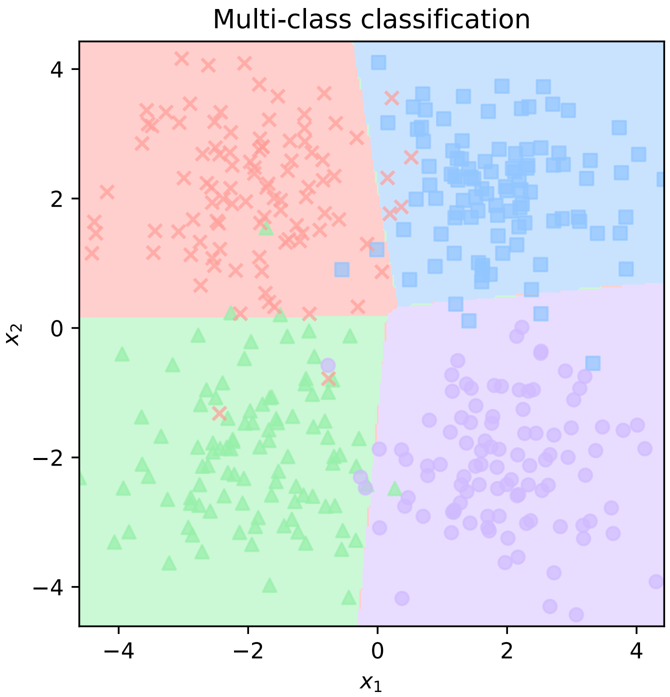
Soft decision boundary:



Linear Multi-Class Classification: Example

Input x: day of the week

Output $f(x)$: precipitation (rain, snow, hail, sun)



Prediction:

- $P(\text{rain}) = f_\theta(\mathbf{x})_1 = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_1$
- $P(\text{snow}) = f_\theta(\mathbf{x})_2 = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_2$
- $P(\text{hail}) = f_\theta(\mathbf{x})_3 = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_3$
- $P(\text{sun}) = f_\theta(\mathbf{x})_4 = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_4$

Linear Multi-Class Classification in PyTorch

Notebook Demo

Linear Multi-Class Classification in PyTorch

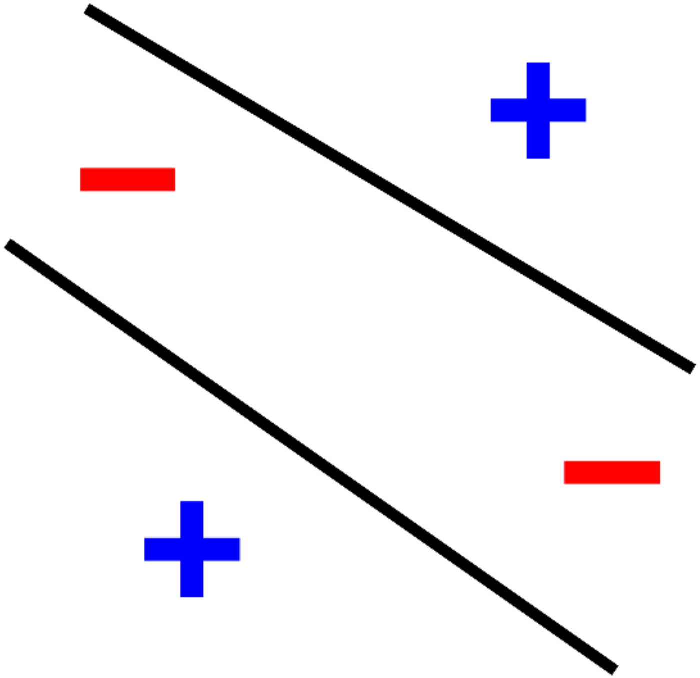
Define a multi-class classifier:

```
class MultiClassClassifier(nn.Module):
    def __init__(self, input_size, num_classes):
        super(MultiClassClassifier, self).__init__()
        self.fc = nn.Linear(input_size, num_classes, bias=True)

    def forward(self, x):
        x = self.fc(x)
        x = F.softmax(x, dim=1)
        return x
```

Linear Multi-Class Classification: Limitation

- Cannot deal with non-linear decision boundaries
- For deep learning, we will need to use non-linear models



Multi-Class vs Multiple Binary Classification: Demo

Notebook Demo

Multi-Class vs Multiple Binary Classification

Multi-class classification (Softmax):

- Describes exactly **one category**
- **no negative** examples
- calibrated probabilities
- used for **mutually exclusive** categories

Multiple binary classifier (Sigmoid):

- Allows for **multiple categories**
- **requires negative** examples
- **uncalibrated probabilities**
- used for **multi-label tagging**

Multi-Class vs Multiple Binary Classification: Examples

Multi-class classification (Softmax):

- each input has **exactly one** category
- no negative examples
- calibrated probabilities
- used for **mutually exclusive** categories

Examples:

- Predicting the weather (rain, cloudy, sunny)
- Predicting the scientific name of an animal
- Predicting the next word in a sentence

Multiple binary classifier (Sigmoid):

- each input may have **multiple** categories
- requires negative examples
- **uncalibrated** probabilities
- used for **multi-label** tagging

Examples:

- Predicting where in Texas it will rain
- Predicting attributes of an animal
- Predicting which books a sentence can be found in

Linear Models - TL;DR

Linear regression: $\mathbf{W}\mathbf{x} + \mathbf{b}$

Linear binary classification: $\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$

Linear multi-class classification: $\mathrm{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$