Loading [MathJax]/extensions/Safe.js

Module 1.0 - Mini-ML

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Model

- Models: parameterized functions.
  - $m(x; \theta)$
  - x - input
  - m - model
- Initial Focus:
  - θ - parameters

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Specifying Parameters

- Datastructures to specify parameters
- Requirements
    - Independent of implementation
    - Compositional

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

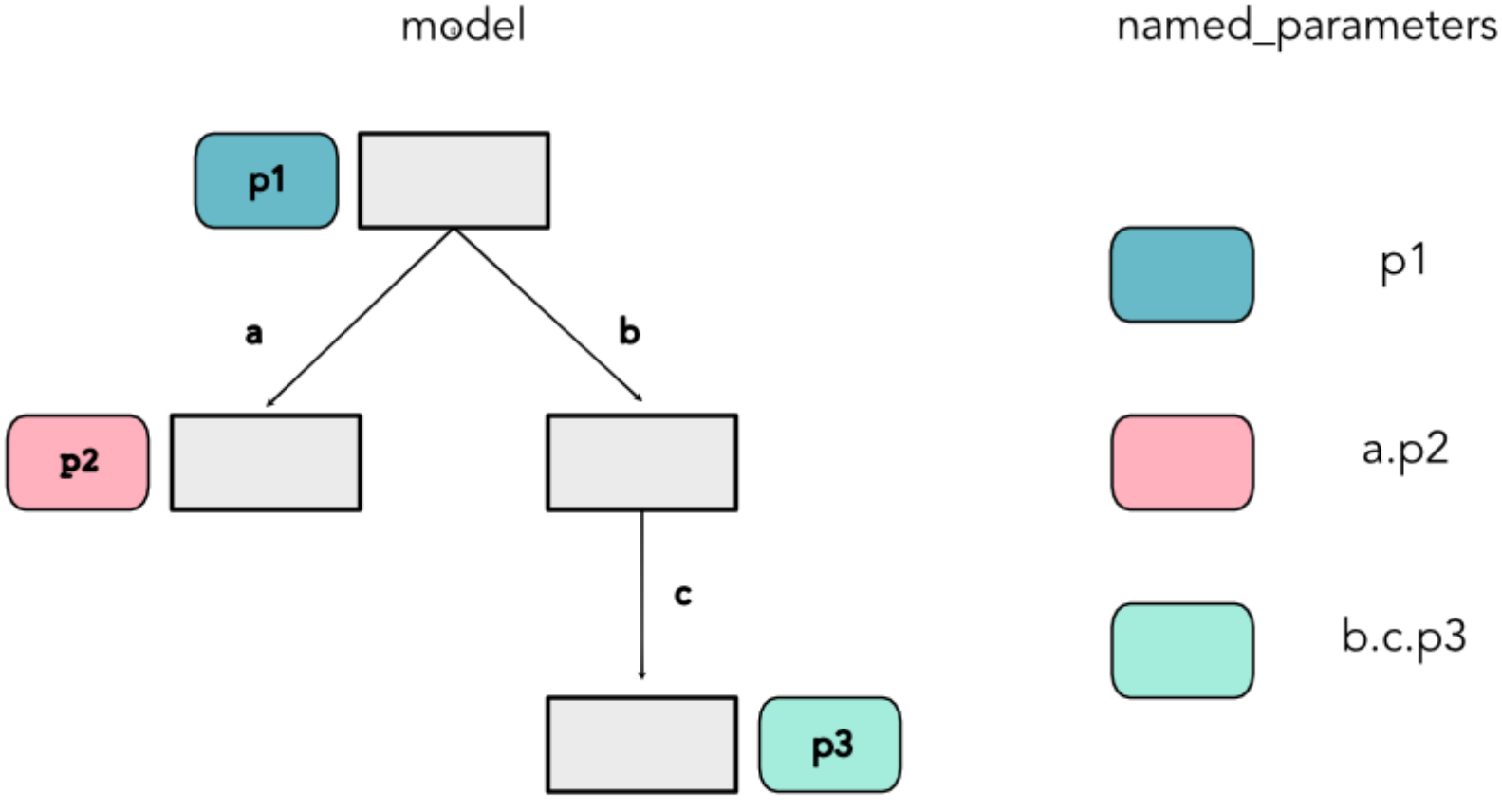
Module Example

```
from minitorch import Module, Parameter


class OtherModule(Module):
    pass


class MyModule(Module):
    def __init__(self):
        # Must initialize the super class!
        super().__init__()

        # Type 1, a parameter.
        self.parameter1 = Parameter(15)

        # Type 2, user data
        self.data = 25

        # Type 3. another Module
        self.sub_module = OtherModule()
```

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Module Naming



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Lecture Quiz

Quiz

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Outline

- Model
- Parameters
- Loss

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

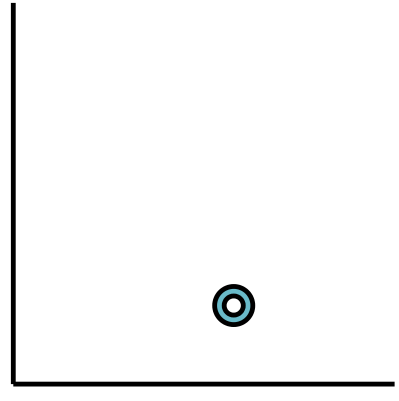
Datasets

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Points

- Convention x

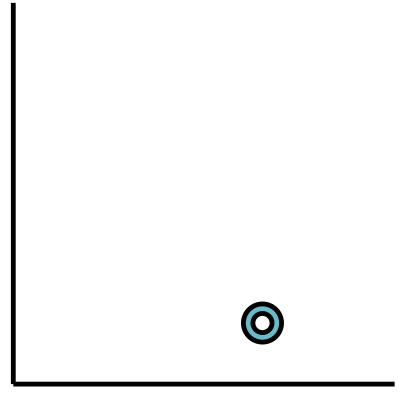


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Points

- Convention x

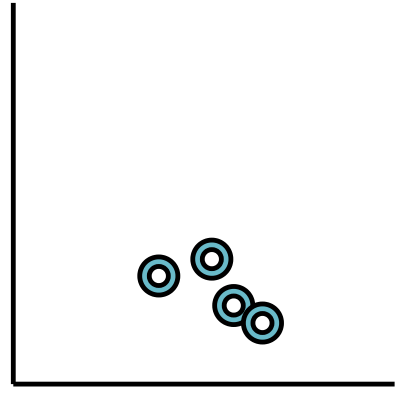


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Points

- Convention x

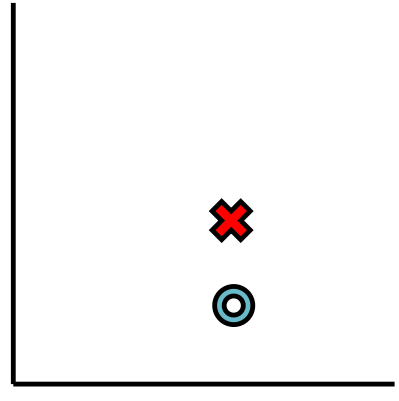


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Labels

- Convention y

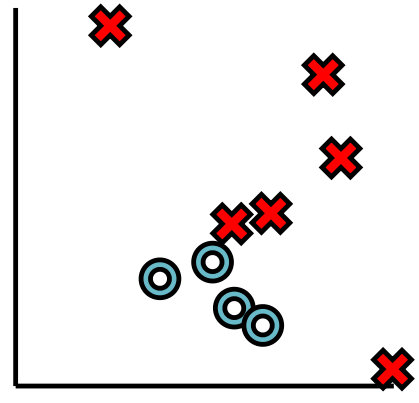


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Training Data

- Set of datapoints, each (x, y)

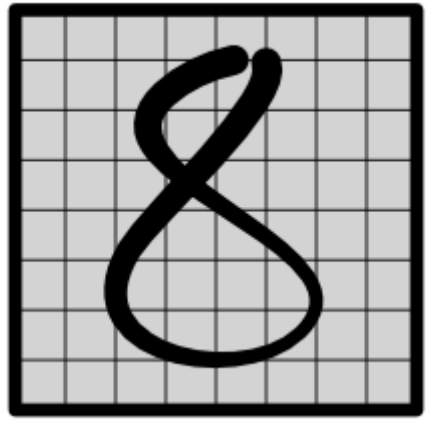


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Points

- Convention x



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Points

- Convention x

Loading [MathJax]/extensions/Safe.js

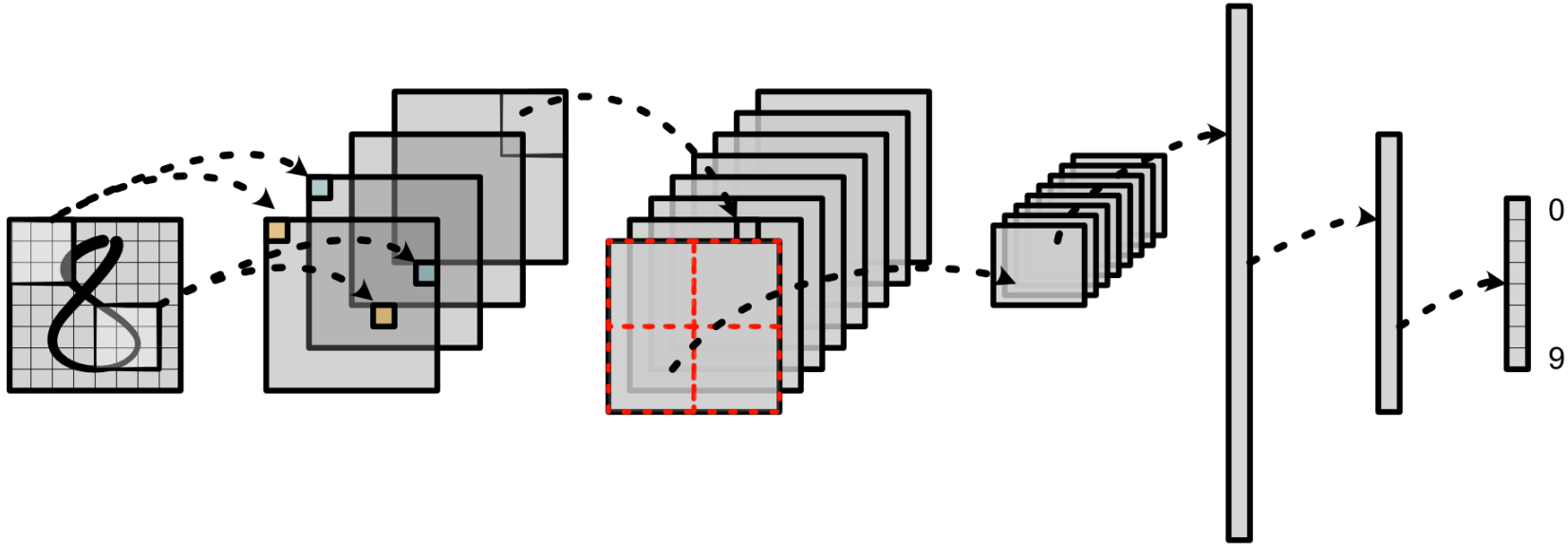
Loading [MathJax]/extensions/Safe.js

Data Set

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Network

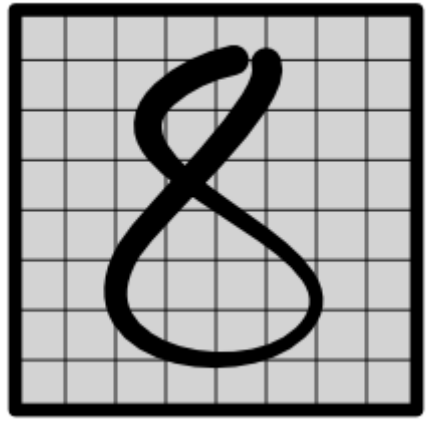


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Data Labels

- Convention y



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Model

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Models

- Functions from data points to labels
- Functions $m(x; \theta)$
- Any function is okay (e.g. Modules)

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Example Model

- Example of a simple model

  x = (0.5, 0.2)

```
class Model:
    def forward(self, x):
        return 0 if x[0] < 0.5 else 1
```

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Model 1

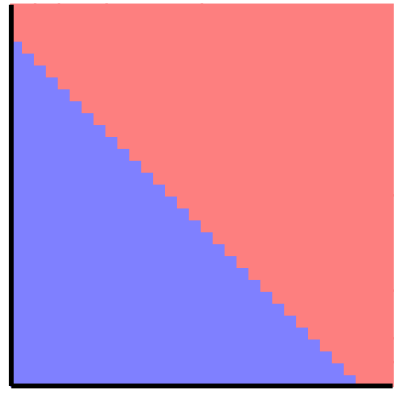
- Linear Model

```
@dataclass
class Linear:
    # Parameters
    w1: float
    w2: float
    b: float

    def forward(self, x1: float, x2: float) -> float:
        return self.w1 * x1 + self.w2 * x2 + self.b
```

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Model 1

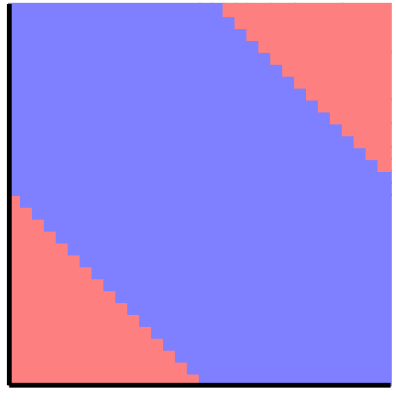


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Model 2

```python
@dataclass
class Split:
    # Submodules
    m1: Linear
    m2: Linear

    def forward(self, x1, x2):
        return self.m1.forward(x1, x2) * self.m2.forward(x1, x2)
```

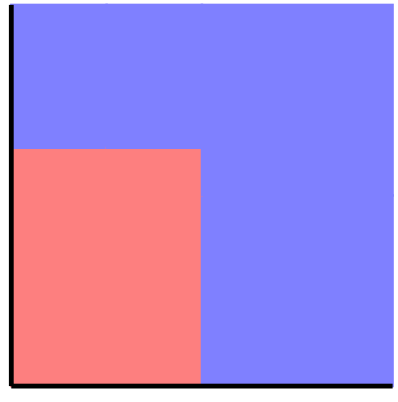


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Model 3

```
class Part:
    def forward(self, x1, x2):
        return 1 if (0.0 <= x1 < 0.5 and 0.0 <= x2 < 0.6) else 0
```



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Parameters

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Parameters

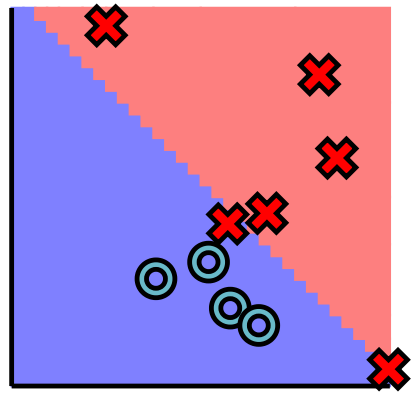
- Knobs that control the model
- Any information that controls the model shape

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Parameters

- Change θ



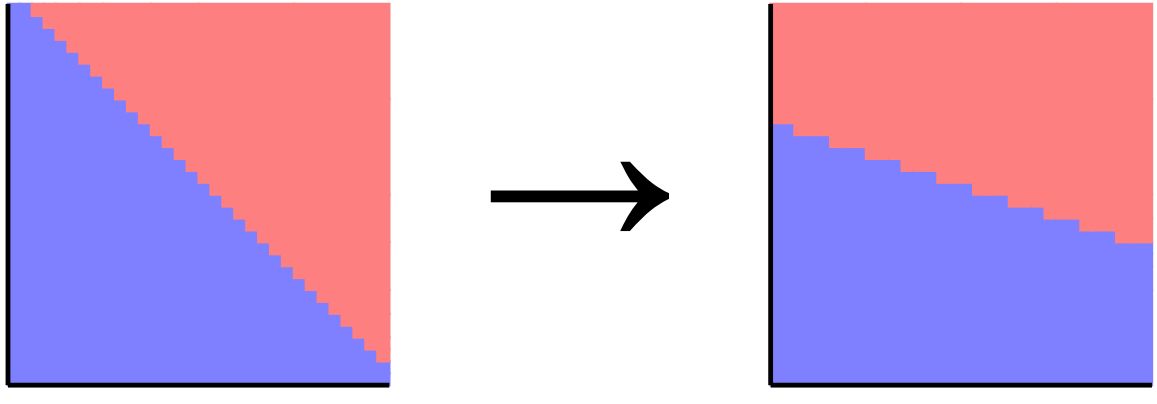
Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Linear Parameters

a. rotating the linear separator

```
model1 = Linear(1, 1, -1.0)
model2 = Linear(0.5, 1.5, -1.0)
```



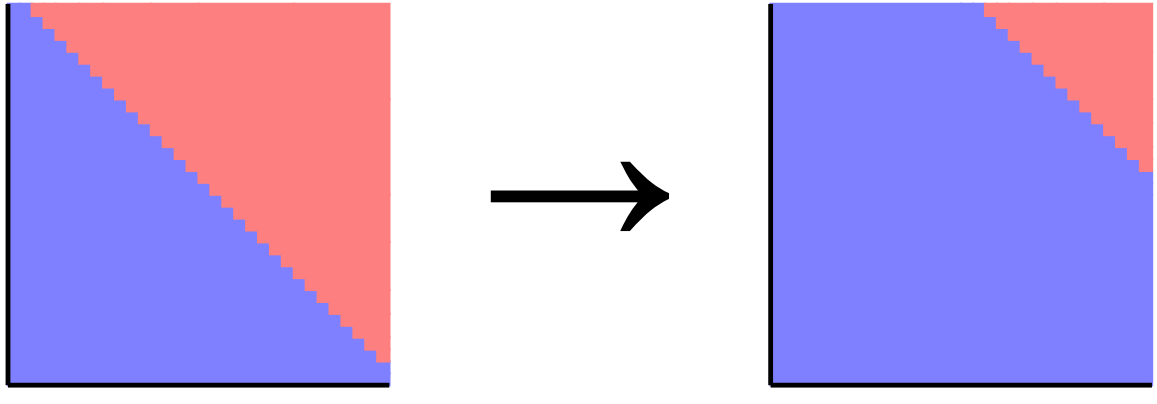
Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Linear Parameters

b. changing the separator cutoff

```
model1 = Linear(1, 1, -1.0)
model2 = Linear(1, 1, -1.5)
```



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Math

- Linear Model

$$m(x; w, b) = x_1 \times w_1 + x_2 \times w_2 + b$$

```
def forward(self, x1: float, x2: float) -> float:
    return self.w1 * x1 + self.w2 * x2 + self.b
```

Loading [MathJax]/extensions/Safe.js

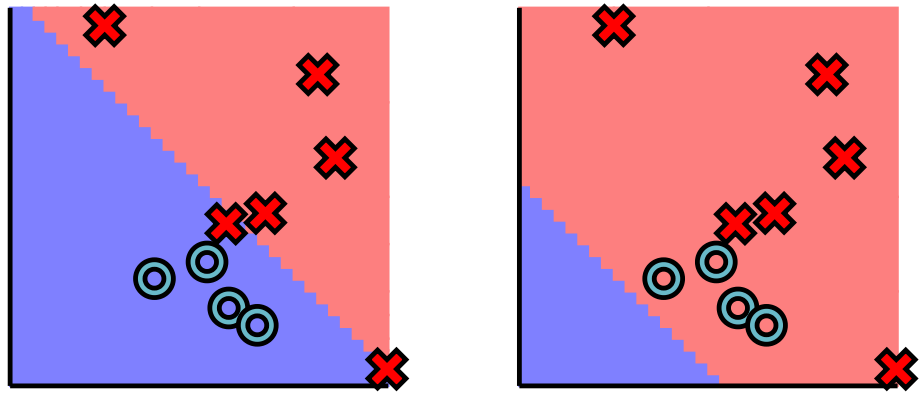
Loading [MathJax]/extensions/Safe.js

Loss

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

What is a good model?

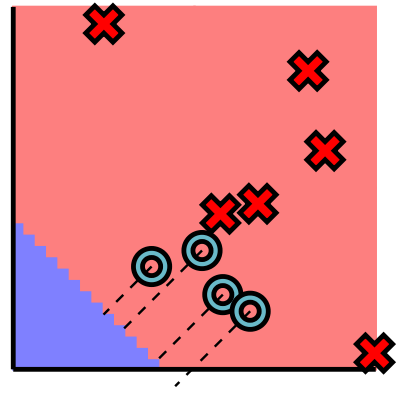


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Distance

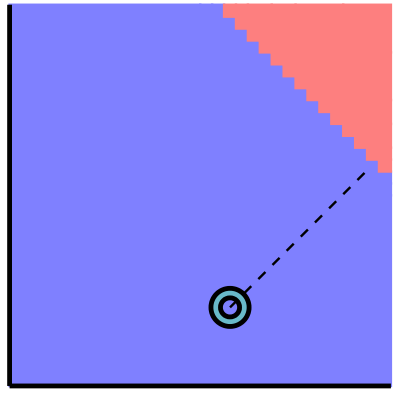
- $|m(x)|$ correct or incorrect

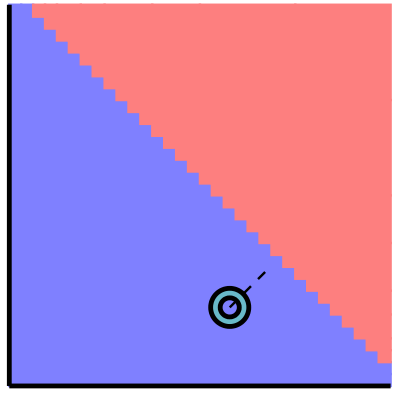


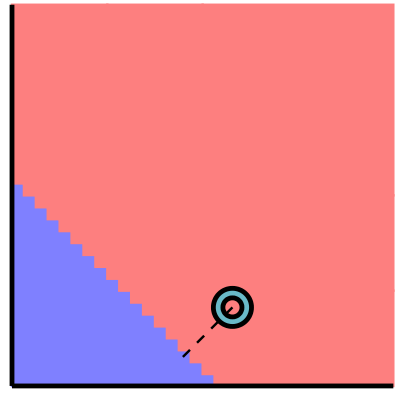
Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Points







Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Loss

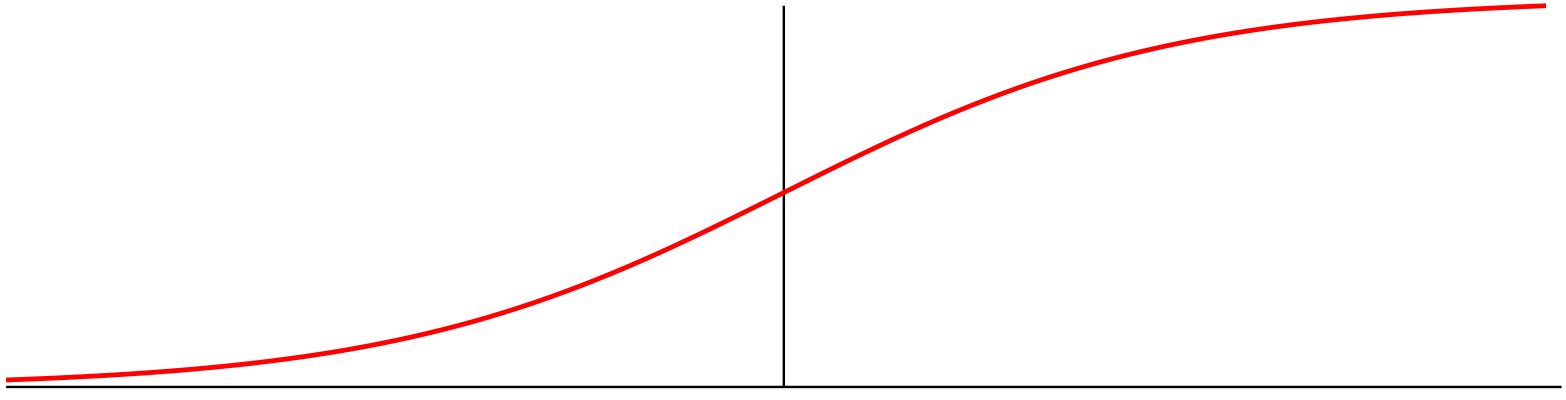
- Loss weights our incorrect points
- Uses distance from boundary

  $L(w_1, w_2, b)$ is loss, function of parameters.

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Sigmoid Function

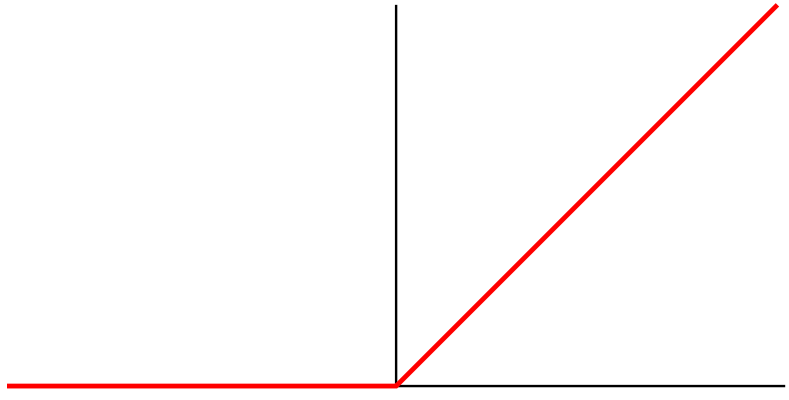


Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Log Sigmoid loss

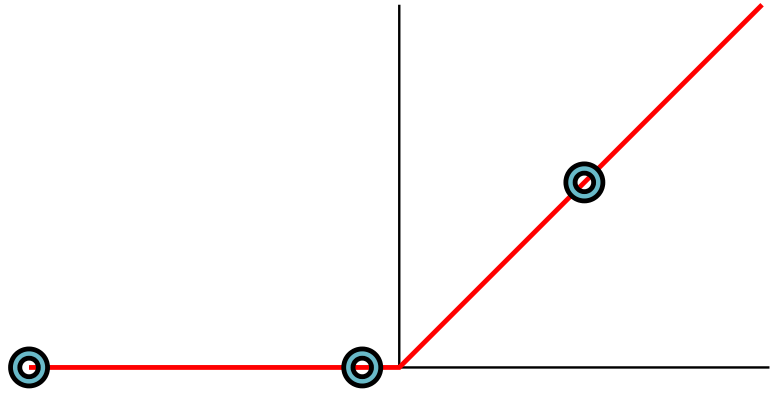
```
def point_loss(x):
    return minitorch.operators.relu(x)
```



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

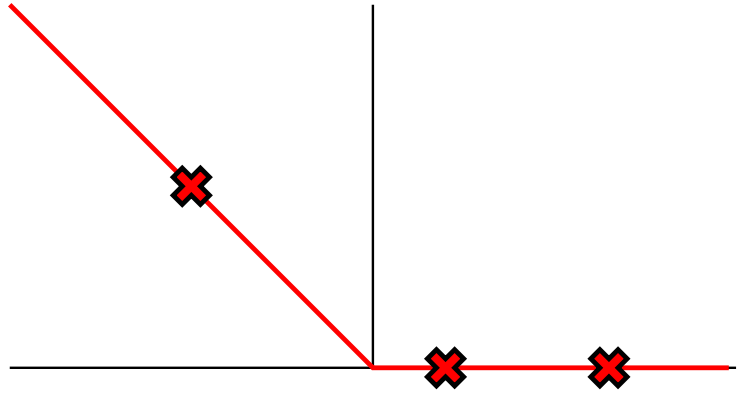
Loss of points



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Loss of points



Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Full Loss

```
def full_loss(m):
    l = 0
    for x, y in zip(s.X, s.y):
        l += point_loss(-y * m.forward(*x))
    return -l
```

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Playground

Playground

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Q&A

```
import torch
```

```
x = torch.tensor([0.0], requires_grad=True)
torch.relu(x).backward()
x.grad
```

```
tensor([0.])
```

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js