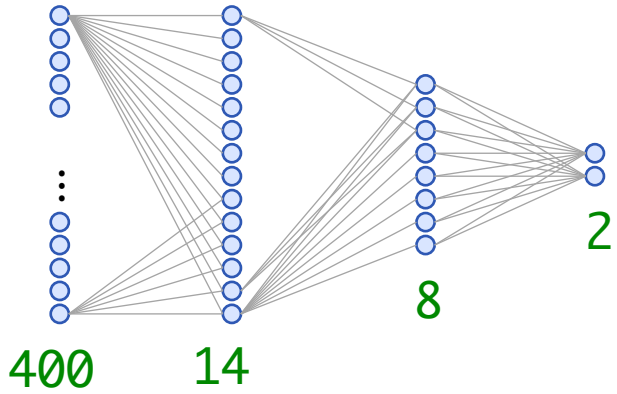
Practical Machine Learning

Backpropagation

Sven Mayer

Parameter

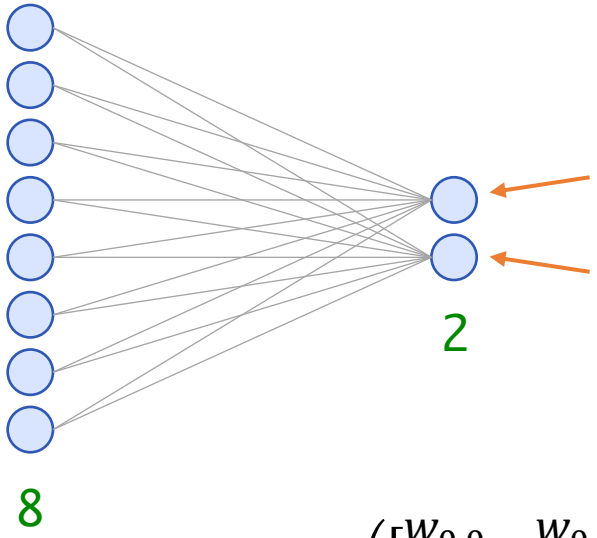


- Layers 400, 14, 8, and 2
  → weights: 400 * 14 + 14 * 8 + 8 * 2 = 5,728
  → biases: 14 + 8 + 2 = 24
  → trainable parameter: 5,728 + 24 = 5,752

- Trainable parameters can raise fast
  - Layers 400, 100, 40, 2 => Parameter: 44,222
  - (one model from the walkthough)

Combining Perceptions

Why is it all about fast matrix multiplication?



$$a(x \cdot w_0 + b_0) = y_0$$

$$a(x \cdot w_1 + b_1) = y_1$$

$$a\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,6} & w_{0,7} \\ w_{1,0} & w_{1,1} & \dots & w_{1,6} & w_{0,7} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

$$a\left(\begin{bmatrix} w_{0,0} & \cdots & w_{0,m} \\ \vdots & \ddots & \vdots \\ w_{n,0} & \cdots & w_{n,m} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

Training

Trainable Parameter

Weights
Trainable Parameter

Biases
Trainable Parameter

Input
Given by the previous layer

Output
Given by the label

$$a\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,6} & w_{0,7} \\ w_{1,0} & w_{1,1} & \dots & w_{1,6} & w_{0,7} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

$$a\left(\begin{bmatrix} w_{0,0} & \cdots & w_{0,m} \\ \vdots & \ddots & \vdots \\ w_{n,0} & \cdots & w_{n,m} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

Training

1. Initialize w and b randomly
2. Determine how good the model is using a ”cost function”
3. How can we adjust w and b to be better?

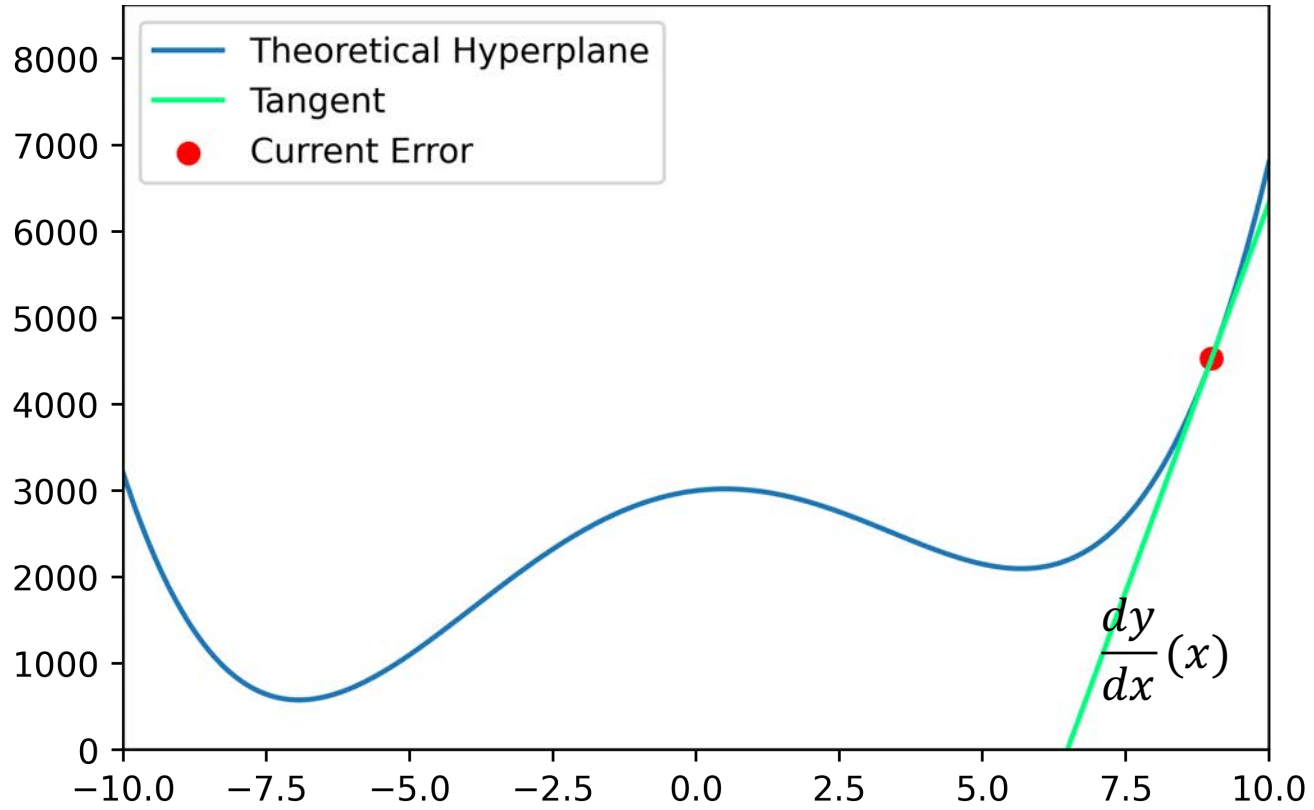
- Cost Functions
  - Squared error
  - RMSE
  - …

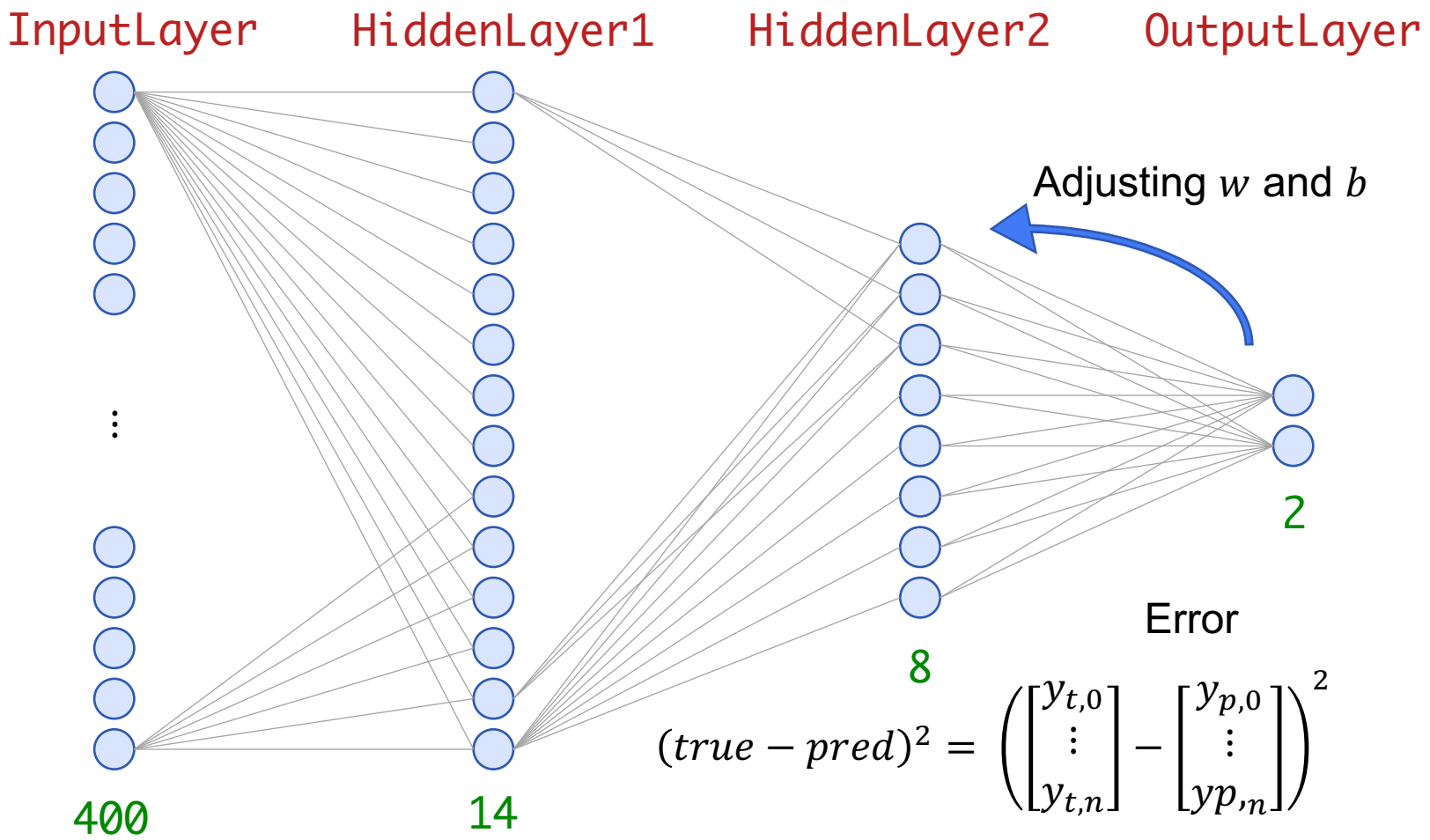
$$\left(\begin{bmatrix} y_{t,0} \\ \vdots \\ y_{t,n} \end{bmatrix} - \begin{bmatrix} y_{p,0} \\ \vdots \\ y_{p,n} \end{bmatrix}\right)^2$$

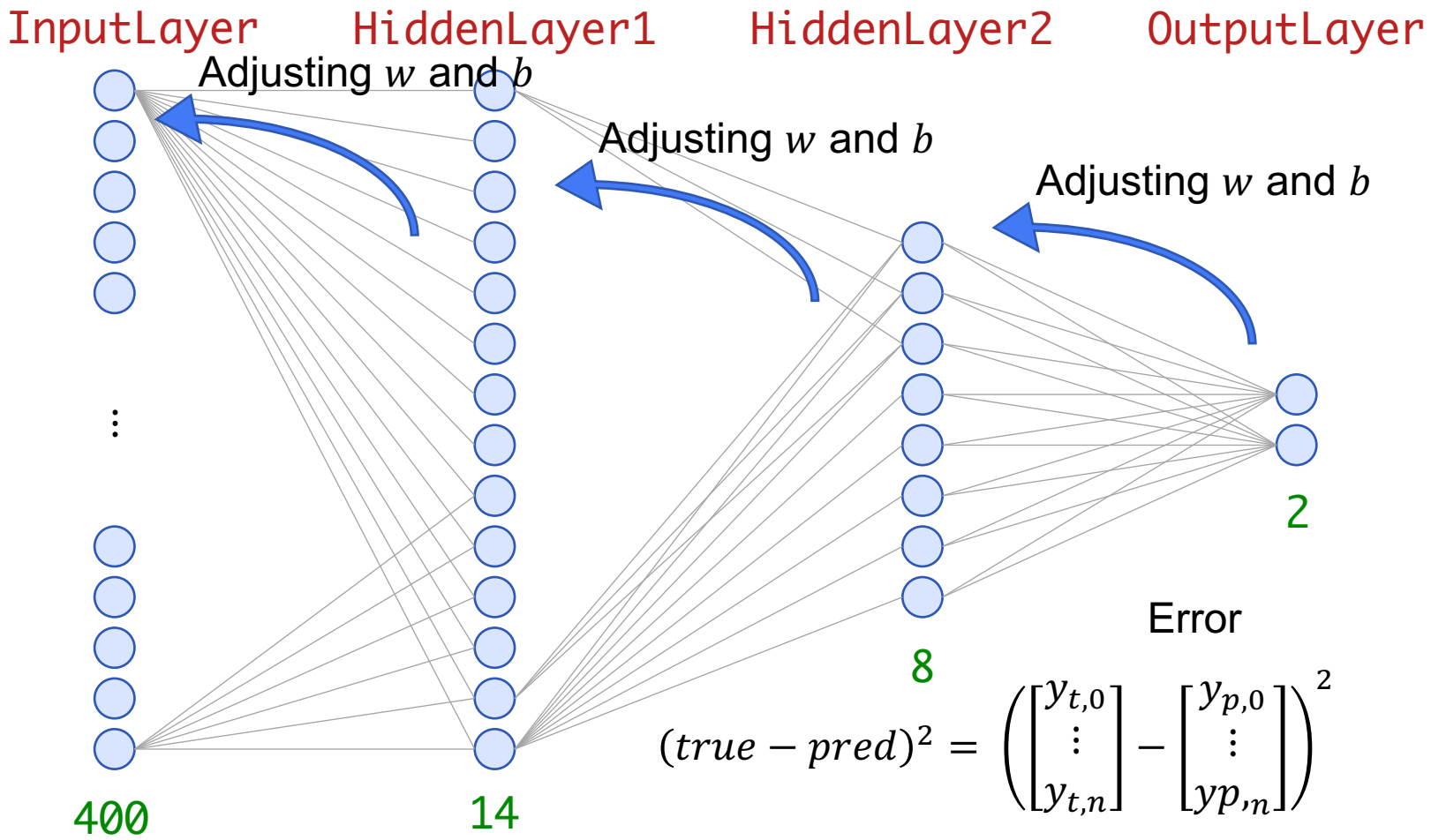
Optimization

“Stochastic Gradient Descent”

- In which direction do we need to go to minimize our cost?




InputLayer
HiddenLayer1
HiddenLayer2
OutputLayer
Adjusting w and b
400
14
8
2
Error
$(true - pred)^2 = \left(\begin{bmatrix} y_{t,0} \\ \vdots \\ y_{t,n} \end{bmatrix} - \begin{bmatrix} y_{p,0} \\ \vdots \\ yp_{,n} \end{bmatrix}\right)^2$


InputLayer
HiddenLayer1
HiddenLayer2
OutputLayer
Adjusting w and b
Adjusting w and b
Adjusting w and b
400
14
8
2
Error
$(true - pred)^2 = \left(\begin{bmatrix} y_{t,0} \\ \vdots \\ y_{t,n} \end{bmatrix} - \begin{bmatrix} y_{p,0} \\ \vdots \\ yp_{,n} \end{bmatrix}\right)^2$

Adjusting w and b

- With the slope, we know how to adjust w and b
- First fast large then small steps

→ “learning rate”

- The gradient can be calculated for each input x
- To not overfit to one x we take more inputs
- Taking all x from takes too long
- In each ”optimization step” we only look at a few x’s

→ ”batch size”

Conclusion

Backpropagation

- Backpropagation
- Trainable Parameter: Weights and biases
- Optimizing e.g. Stochastic Gradient Descent
- Loss function e.g. MSE, MAE, RMSE
- Learning rate
- Batch size

License