Artificial Neural Networks and Deep Learning

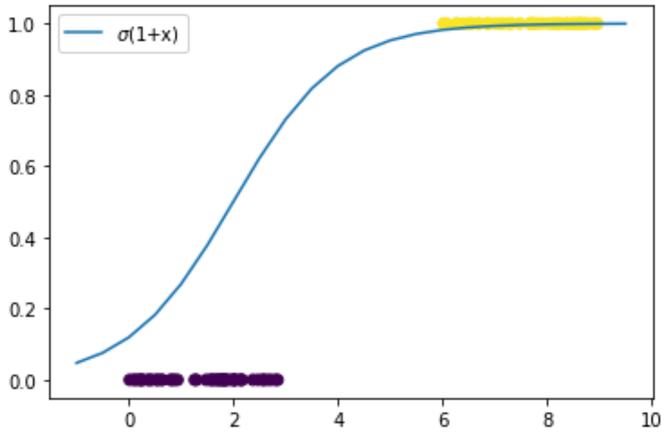
Week 3

Overfitting, regularization and Keras

25-26th of October:

link

Tutorial for gradient descent to fit logistic regression model:



The model weights of $w_0 = 1$ and $w_1 = 1$ don't fit the data quite well.

Let's denote the output of the linear part by f and the output of the sigm
incurred by the model on the training data. The cost function is defined as

$$C(\sigma) = (\sigma - y)^2.$$

In other words, the squared difference between the output (prediction) an
function with respect to the weights

$$\frac{\partial C}{\partial w_i}$$

where i = 0,1. Using the chain rule we can write:

$$\frac{\partial C}{\partial w_i} = \frac{\partial C(\sigma)}{\partial \sigma} \frac{\partial \sigma(f)}{\partial f} \frac{\partial f(w_i)}{\partial w_i}$$

One way to startup

Ento Labs

One way to startup

Ento Labs

"There are plenty of unused data sources that you can build a business from"

One way to startup

Ento Labs

"There are plenty of unused data sources that you can build a business from"

Bekendtgørelse om individuel måling af el, gas, vand, varme og køling[1)]

I medfør af § 4 A, § 28, stk. 4, § 30, stk. 2, og § 31, stk. 2, i byggeloven, jf. lovbekendtgørelse nr. 1185 af 14. oktober 2010, som ændret ved lov nr. 640 af 12. juni 2013, fastsættes:

Anvendelsesområde og definitioner m.v.

§ 1. Bekendtgørelsen omfatter målere, der installeres eller er installeret i eller uden for en bygning for at måle bygningens forbrug af el, gas, koldt vand, varmt vand, varme eller køling.

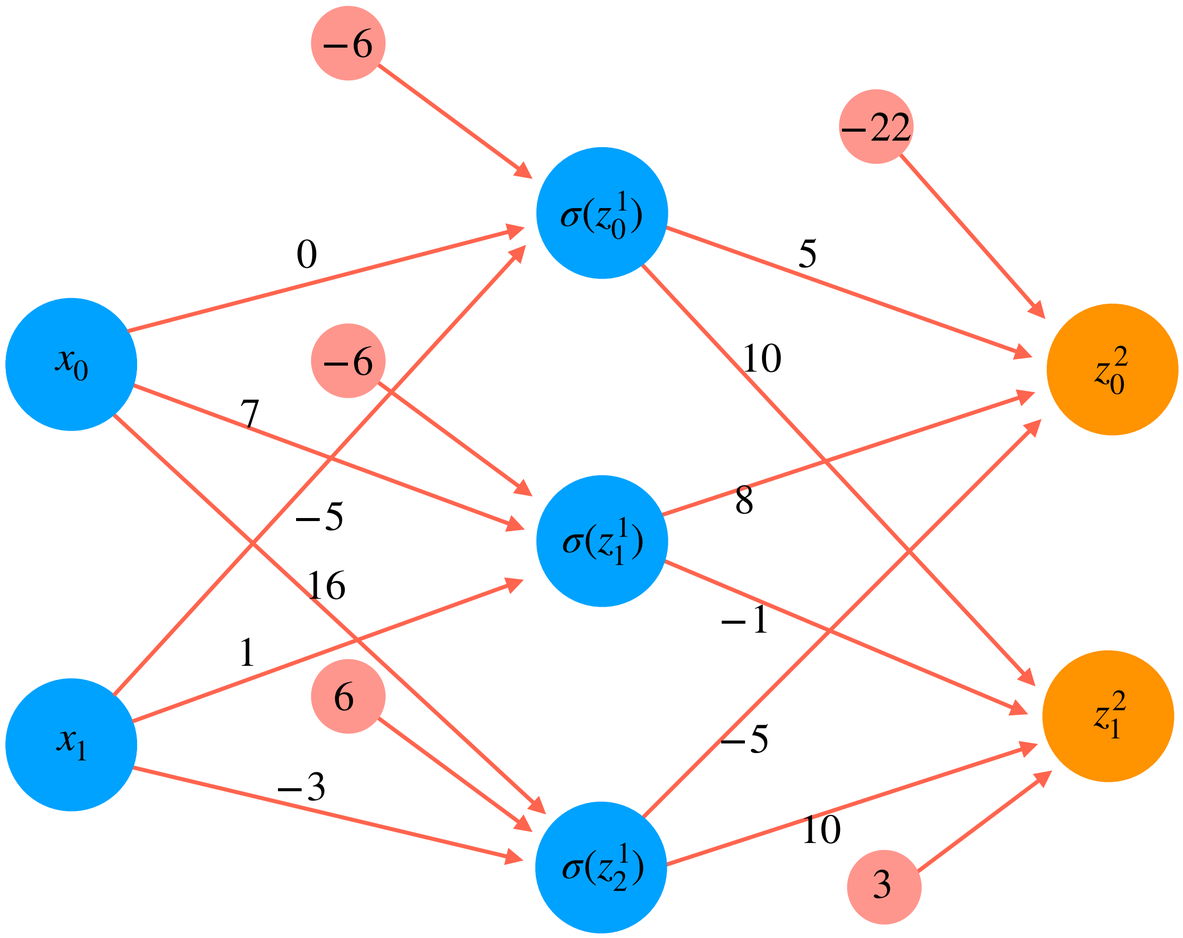
Stk. 2. Bekendtgørelsen omfatter også udskiftning af eksisterende målere.

Stk. 3. Målerne skal anvendes til måling af forbruget. Betaling til forsyningsvirksomheden skal ske efter det målte forbrug, for så vidt angår den del af betalingen, der er forbrugsafhængig.

Stk. 4. Ved fordelingsmåling forstås i denne bekendtgørelse, at for ejendomme, der består af flere bolig- eller erhvervsenheder, hvor betaling til forsyningsvirksomheden sker fælles for ejendommen, fordeles forbruget mellem de enkelte bolig- eller erhvervsenheder efter det på fordelingsmålere registrerede forbrug. Ved afregningsmåling forstås, at

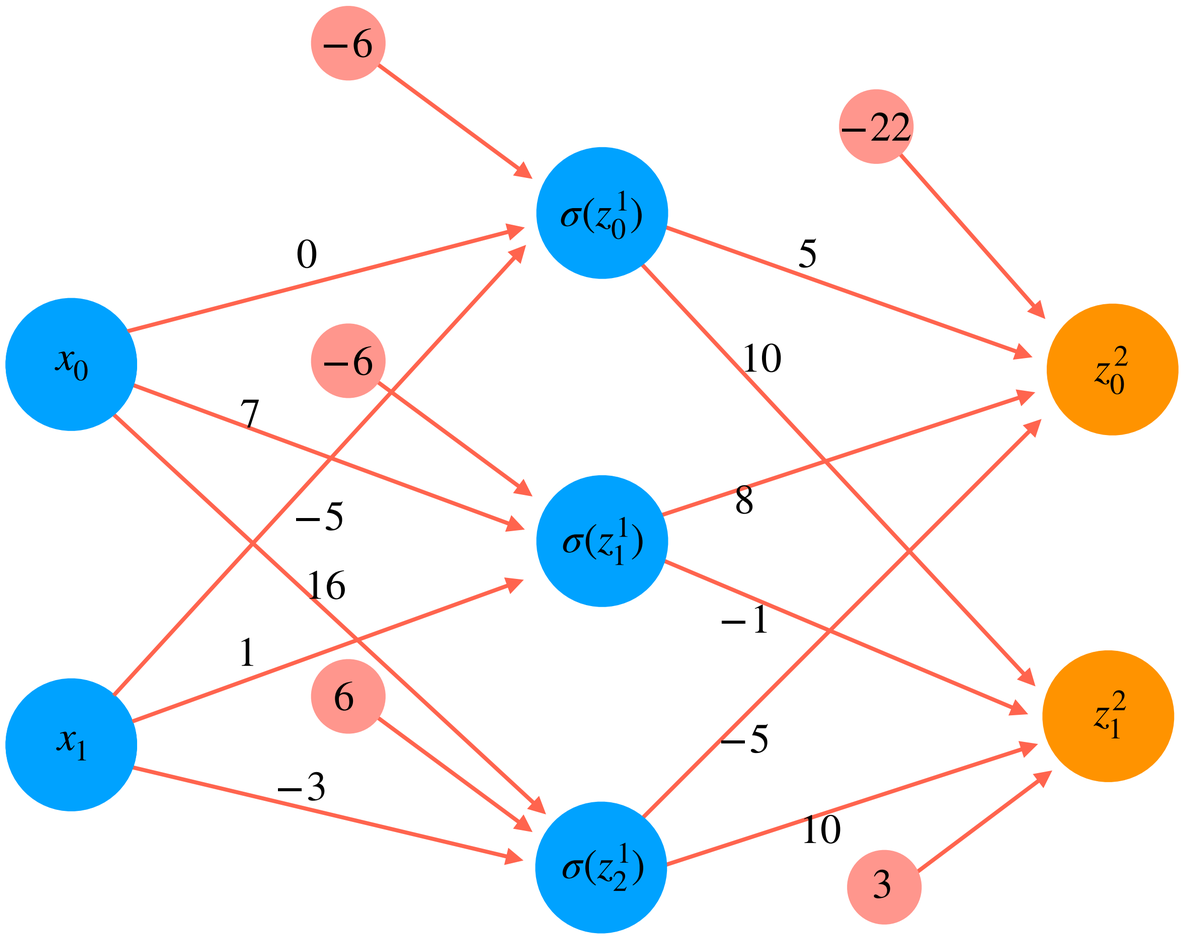
$$y = f(\text{area}, \text{covid}, \text{building}, \ldots))$$

Backprop example - multioutput regression



$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix}$$

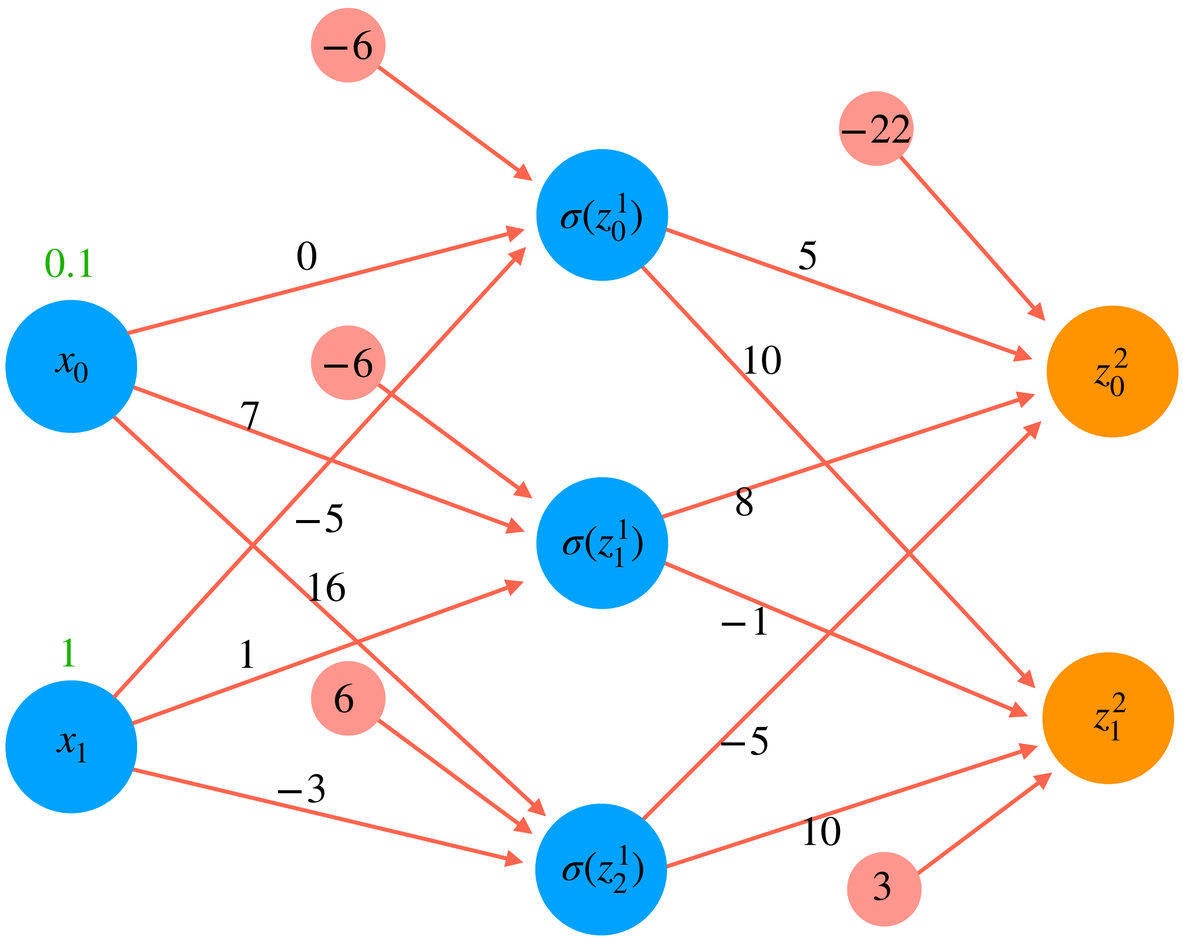
Backprop example - multioutput regression



$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$

Backprop example - multioutput regression

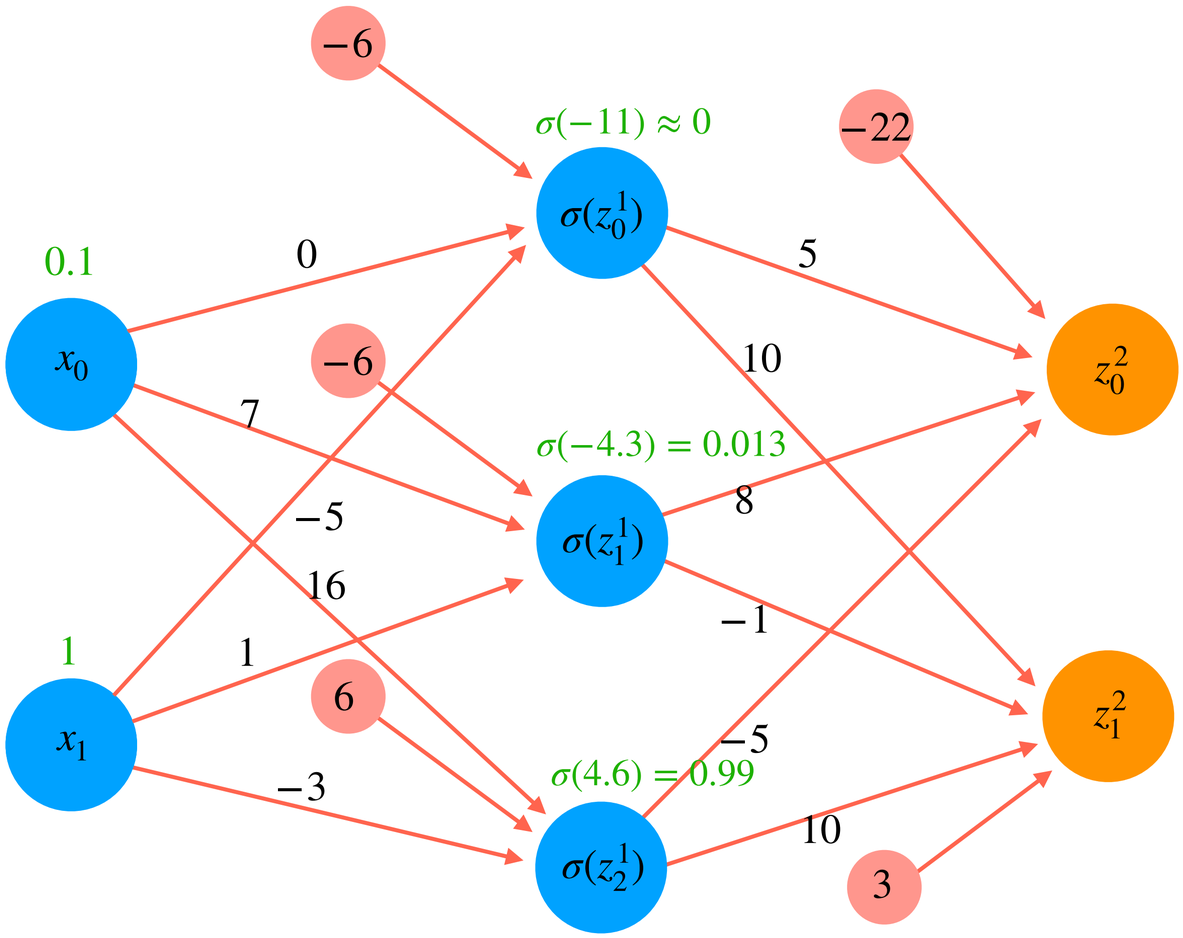


Data (x,y)

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -14 \\ 13 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$

Backprop example - multioutput regression

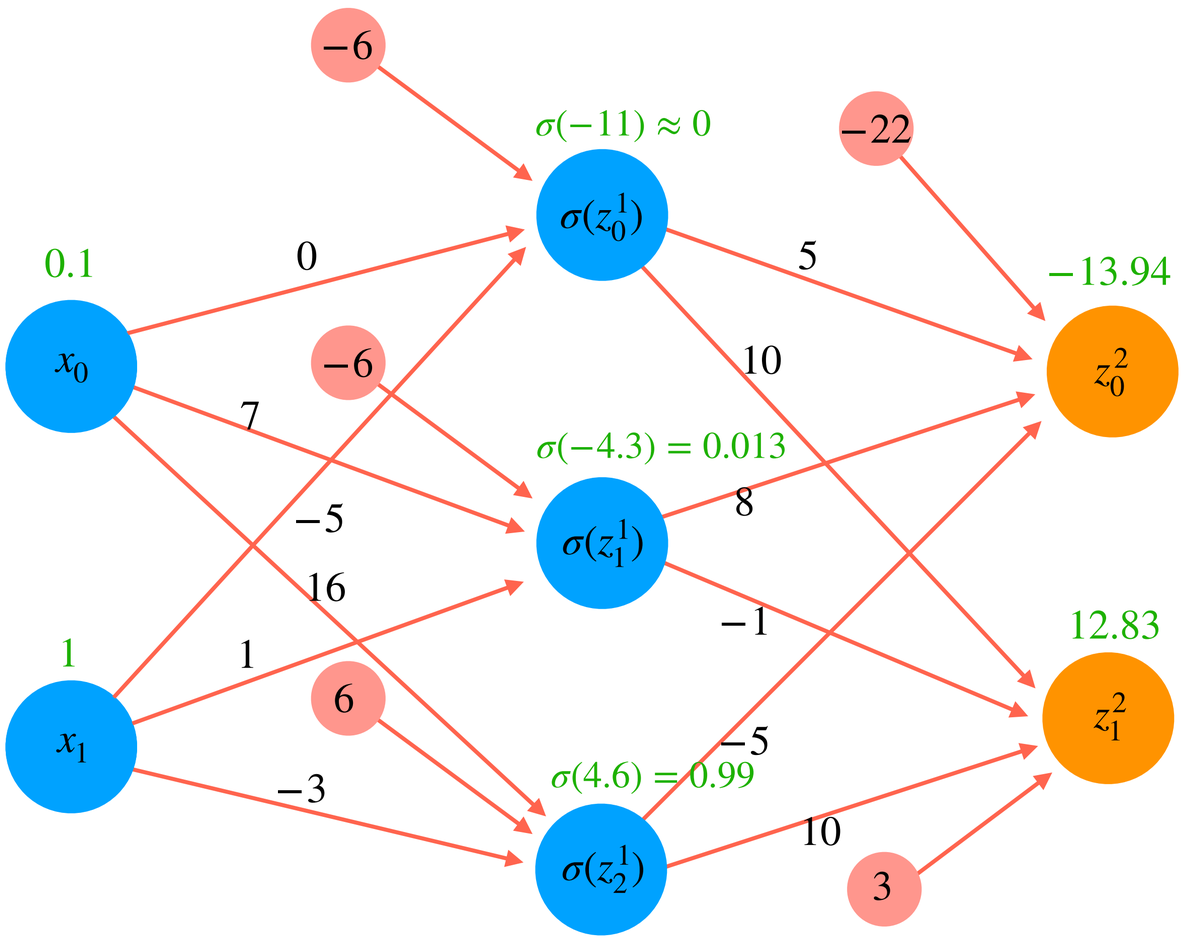


Data (x,y)

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -14 \\ 13 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$

Backprop example - multioutput regression

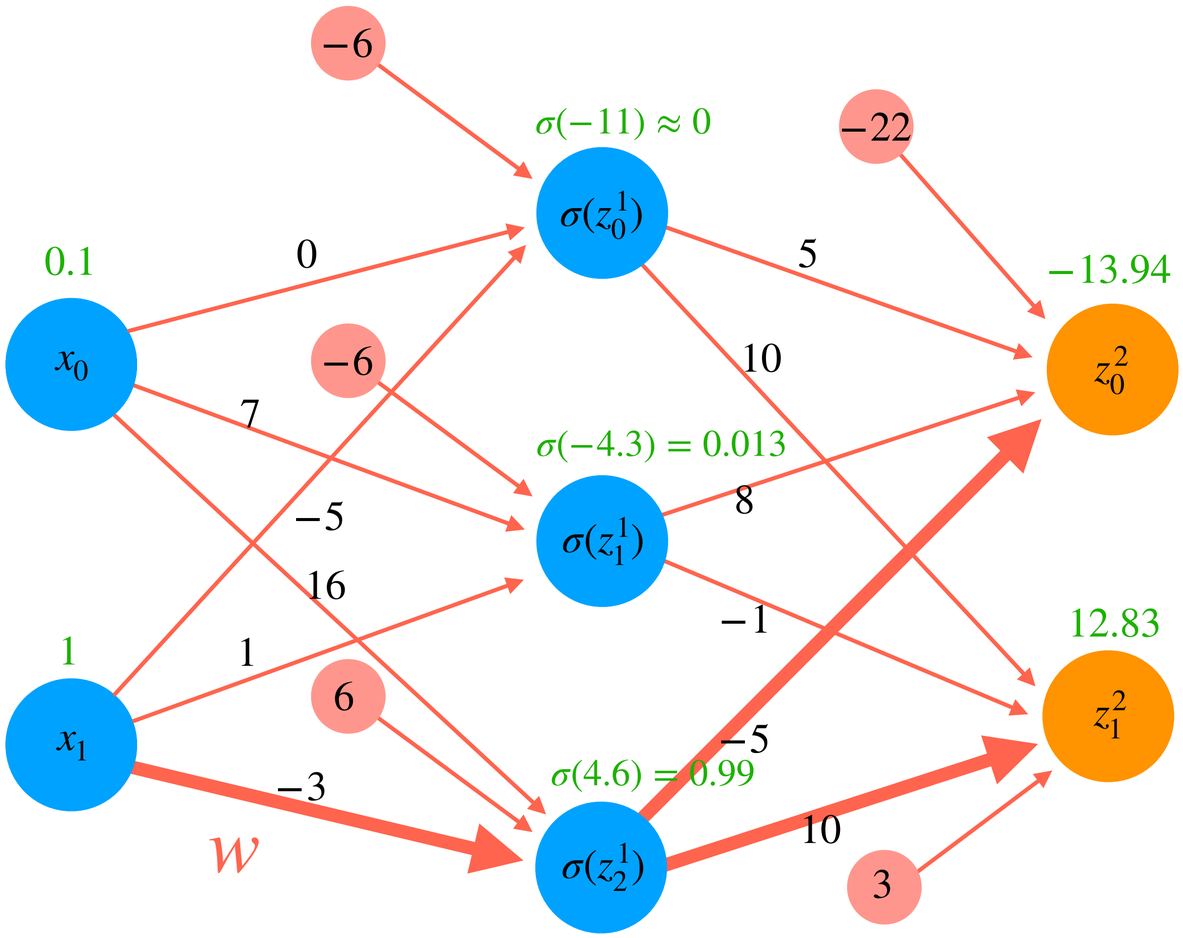


Data (x,y)

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -14 \\ 13 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$
$$= (0.06)^2 + (0.17)^2 = 0.0325$$

Backprop example - multioutput regression



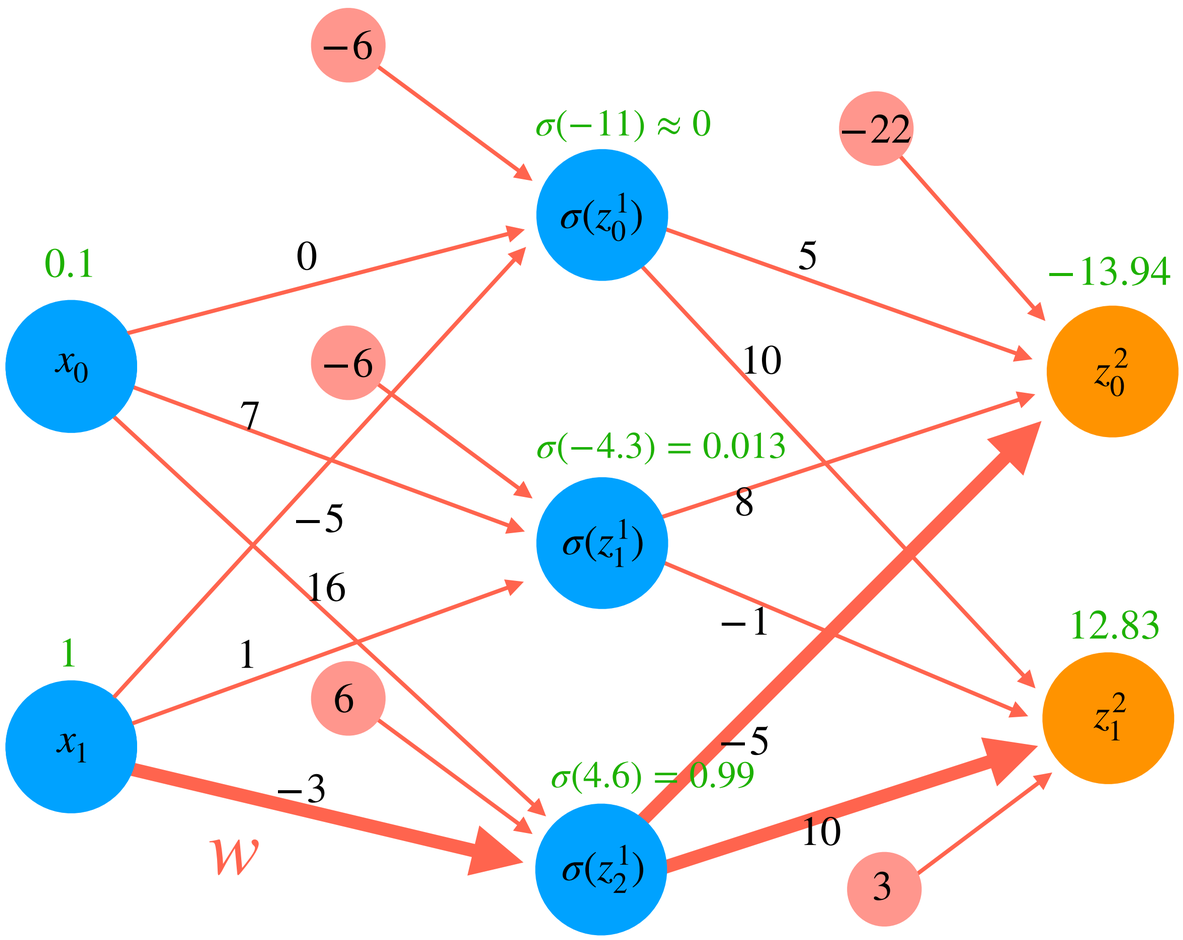
Data (x,y)

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -14 \\ 13 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$
$$= (0.06)^2 + (0.17)^2 = 0.0325$$

- w affects C through more than one path

Backprop example - multioutput regression



Data (x,y)

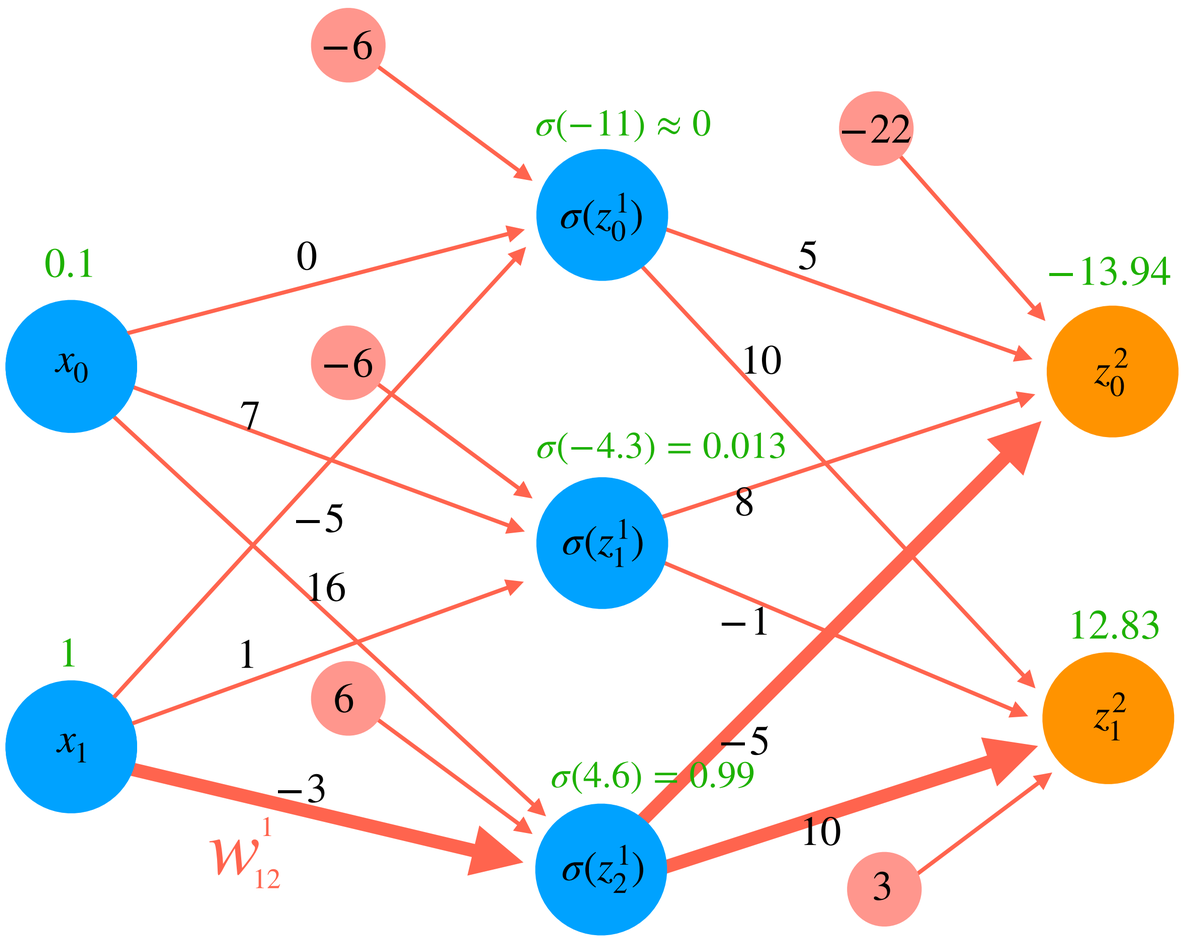
$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -14 \\ 13 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$
$$= (0.06)^2 + (0.17)^2 = 0.0325$$

- w affects C through more than one path
- Not a problem if C is a sum over the outputs (differentiation is a *linear operator*)

$$\frac{\partial}{\partial w} C = \frac{\partial}{\partial w} \left[\left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2 \right]$$
$$= \frac{\partial}{\partial w} \left(\hat{y}_0 - y_0\right)^2 + \frac{\partial}{\partial w} \left(\hat{y}_1 - y_1\right)^2$$

Backprop example - multioutput regression



Data (x,y)

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} z_0^2 \\ z_1^2 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -14 \\ 13 \end{bmatrix}$$

$$C(\hat{\mathbf{y}}, \mathbf{y}) = \sum_d \left(\hat{y}_d - y_d\right)^2 = \left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2$$
$$= (0.06)^2 + (0.17)^2 = 0.0325$$

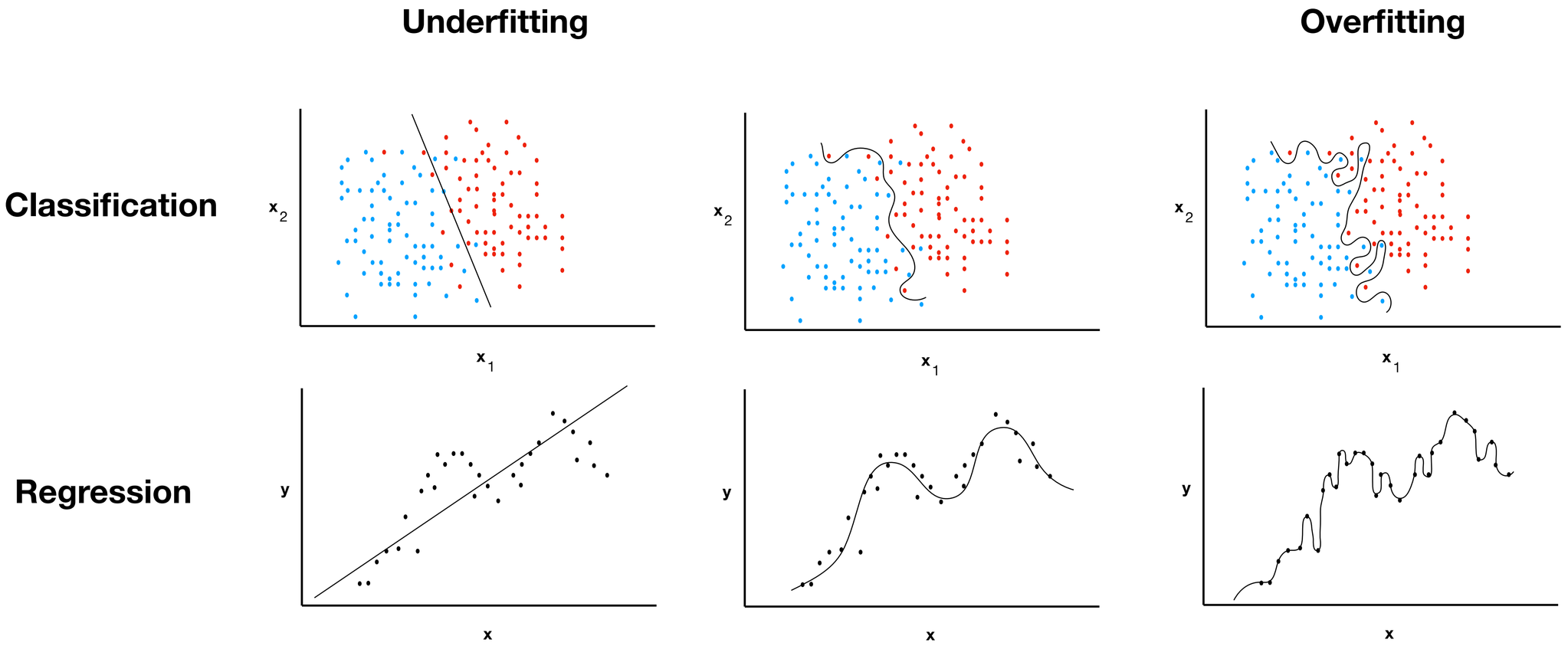
- w affects C through more than one path
- Not a problem if C is a sum over the outputs (differentiation is a *linear operator*)

$$\frac{\partial}{\partial w} C = \frac{\partial}{\partial w} \left[\left(\hat{y}_0 - y_0\right)^2 + \left(\hat{y}_1 - y_1\right)^2 \right]$$
$$= \frac{\partial}{\partial w} \left(\hat{y}_0 - y_0\right)^2 + \frac{\partial}{\partial w} \left(\hat{y}_1 - y_1\right)^2$$

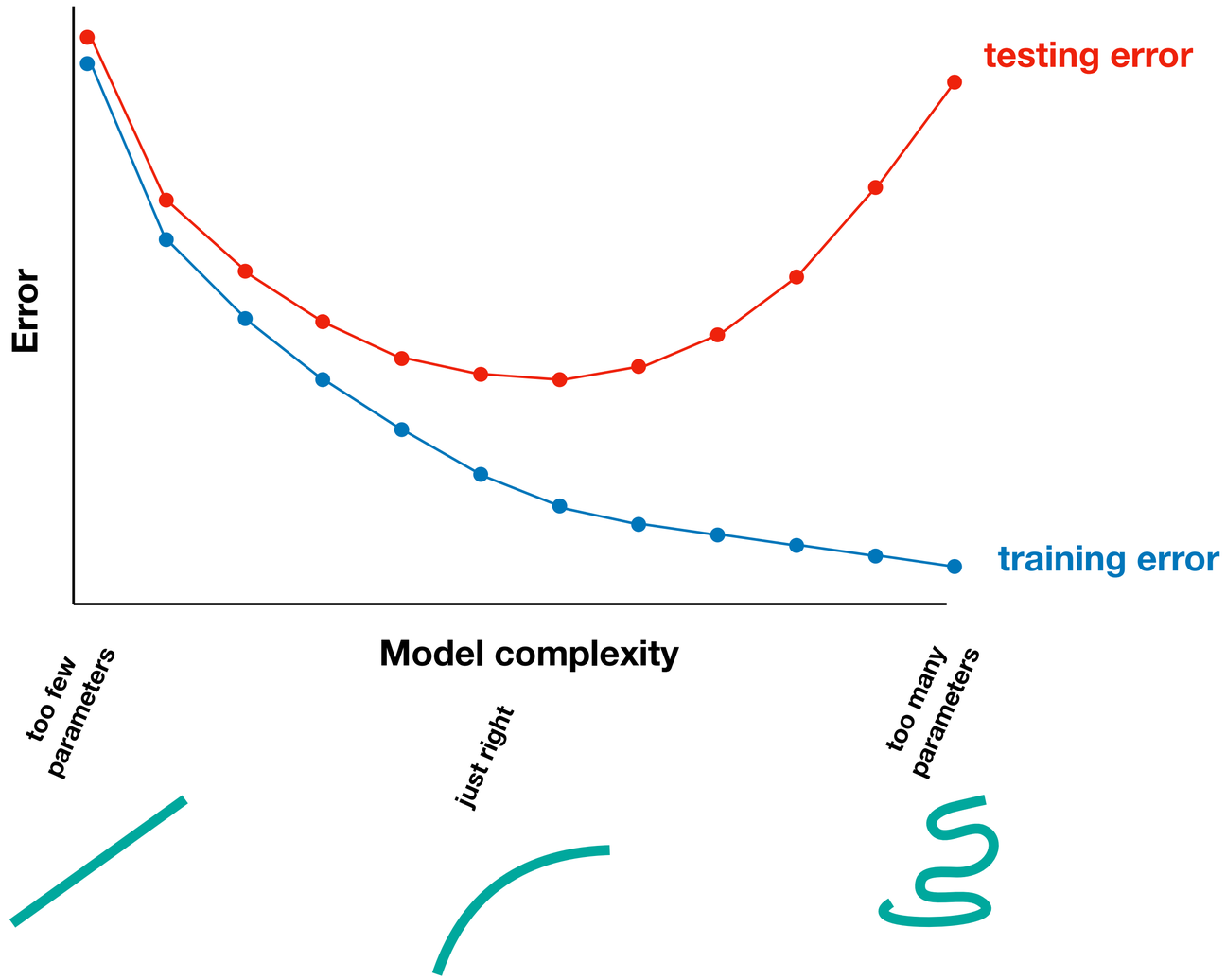
Regularization

Tricks to avoid overfitting

Regularization – underfitting and overfitting



Regularization – underfitting and overfitting



How does regularization reduce overfitting?

Regularization – Different techniques

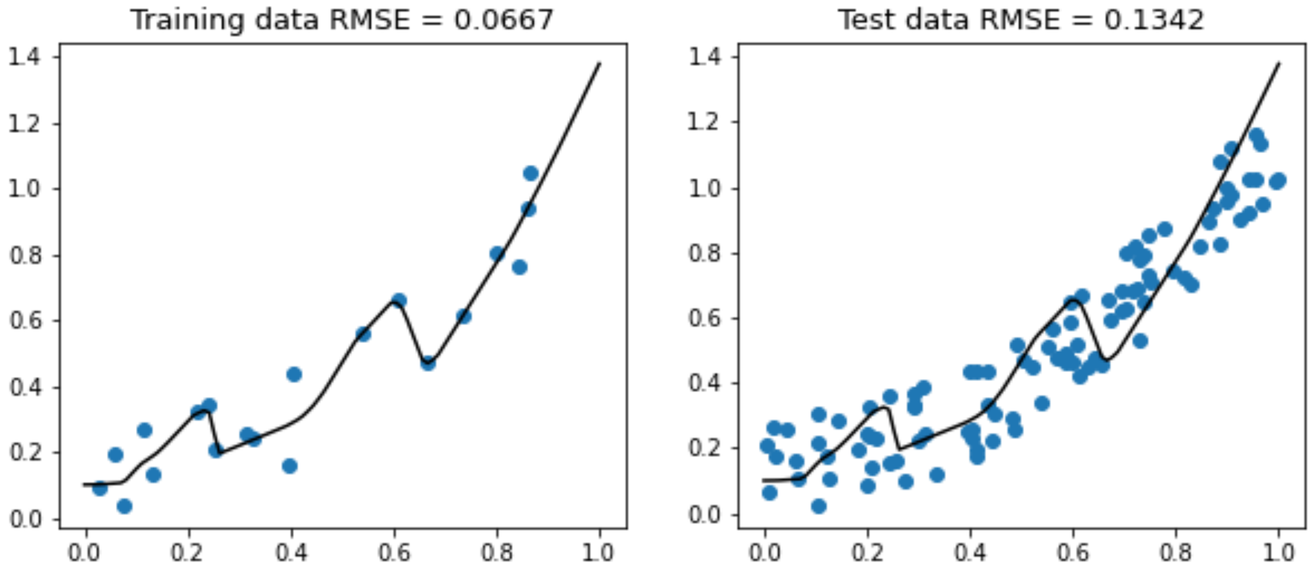
Early stopping

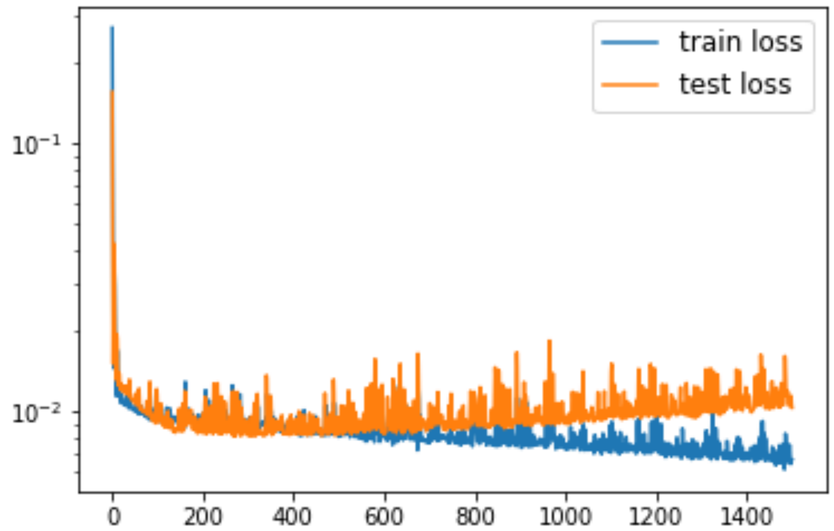
"Stop training when performance on validation dataset starts worsening"

Regularization – Different techniques

Early stopping

"Stop training when performance on validation dataset starts worsening"



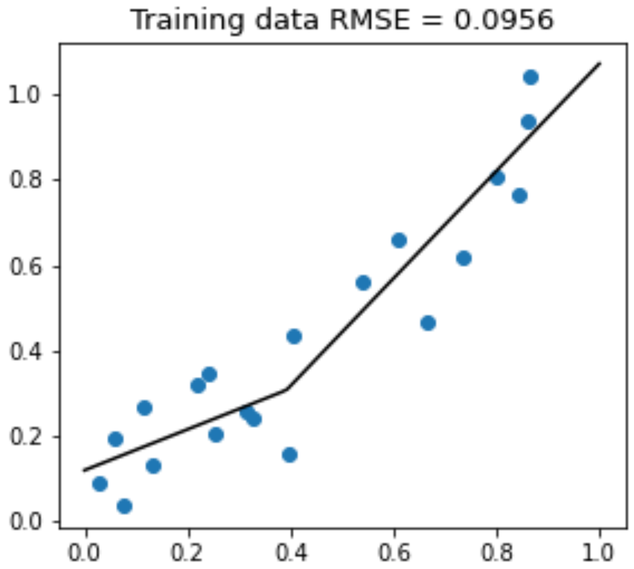


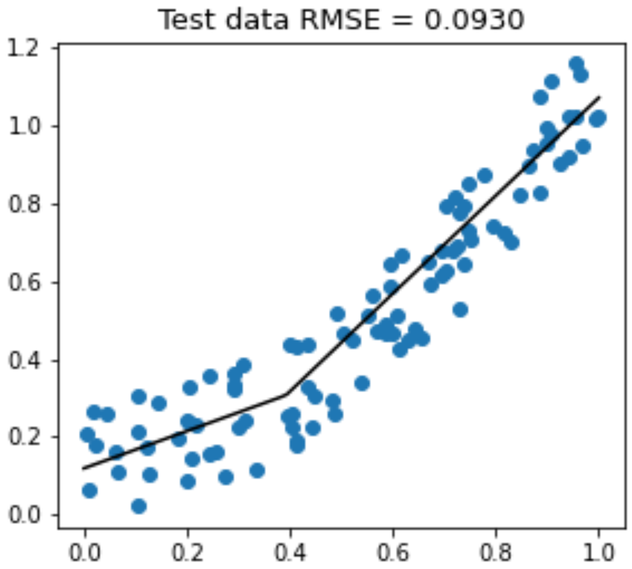

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(500, input_dim=1, activation='relu'))
model.add(Dense(500, activation='relu'))
model.add(Dense(1))
```

Regularization – Different techniques

Early stopping

"Stop training when performance on validation dataset starts worsening"






```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(10, input_dim=1, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1))
```

Regularization – Different techniques

Early stopping

"Stop training when performance on validation dataset starts worsening"

L-norm regularization: "Introduce a cost for large weights"

$$C = Loss + Regularization\ term$$

Regularization – Different techniques

Early stopping

"Stop training when performance on validation dataset starts worsening"

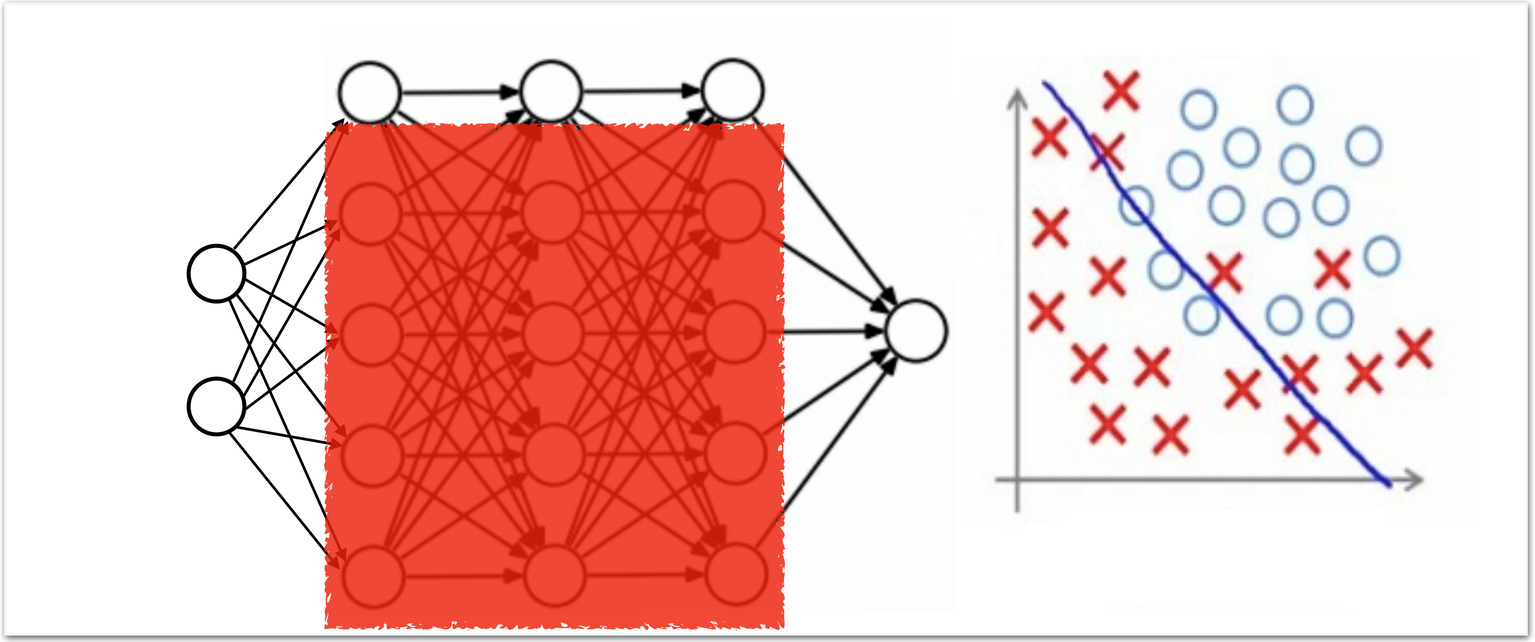
L-norm regularization: "Introduce a cost for large weights"

$$C = Loss + Regularization\ term$$

L1: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_1$

L2: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_2$

Regularization – Different techniques



https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Regularization – Different techniques

Early stopping

"Stop training when performance on validation dataset starts worsening"

L-norm regularization: "Introduce a cost for large weights"

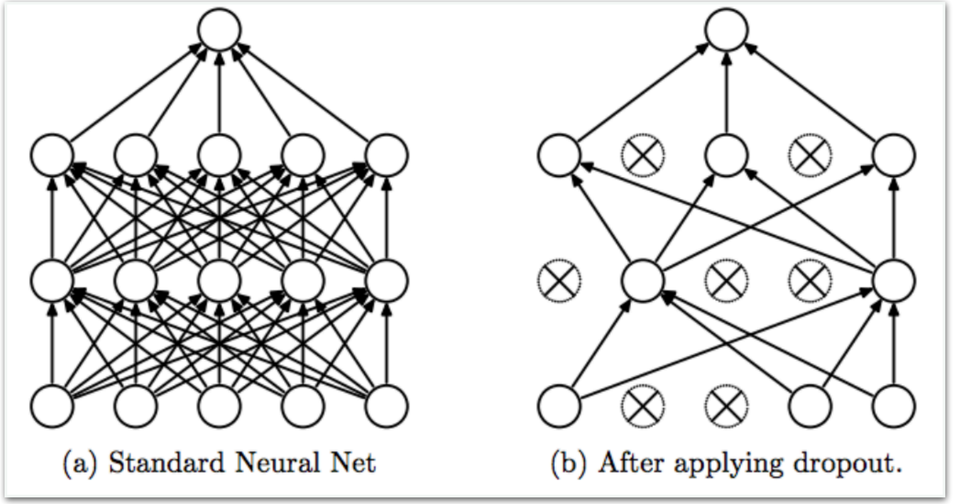
$$C = Loss + Regularization\ term$$

L1: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_1$

L2: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_2$

Dropout:

"In each SGD step, randomly ignore a fraction *p* of neurons"



Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

- Can select *p* in wide range. Typical is 0.2 – 0.8, dependent on size of ANN
- Typical to apply dropout in layers close to the output.

Regularization – Different techniques

Early stopping

"Stop training when performance on validation dataset starts worsening"

L-norm regularization: "Introduce a cost for large weights"

$$C = Loss + Regularization\ term$$

L1: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_1$

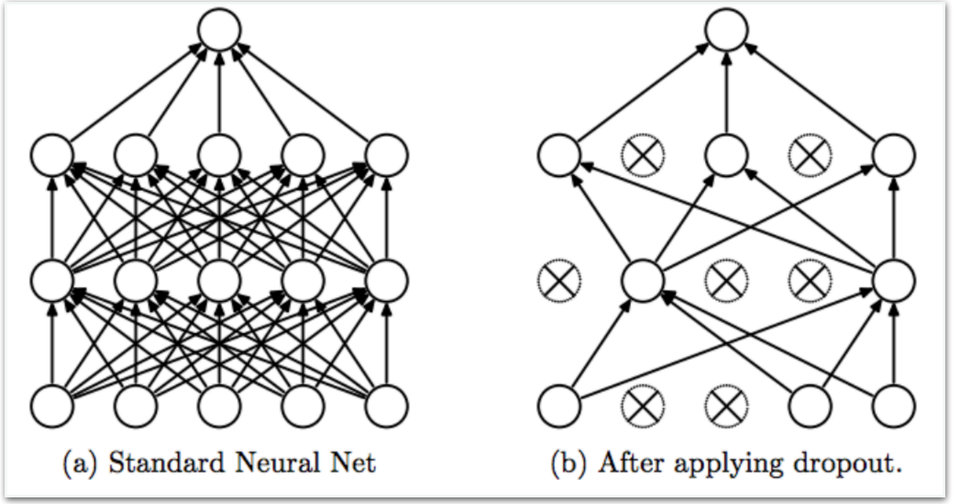
L2: $C = Loss + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_2$

Data augmentation (/more data)

"Shear, shift, scale and/or rotate input data"

Dropout:

"In each SGD step, randomly ignore a fraction *p* of neurons"



Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

- Can select *p* in wide range. Typical is 0.2 – 0.8, dependent on size of ANN
- Typical to apply dropout in layers close to the output.

A word on:

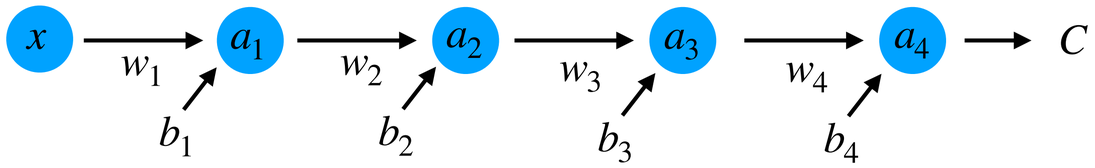
The Vanishing Gradient Problem

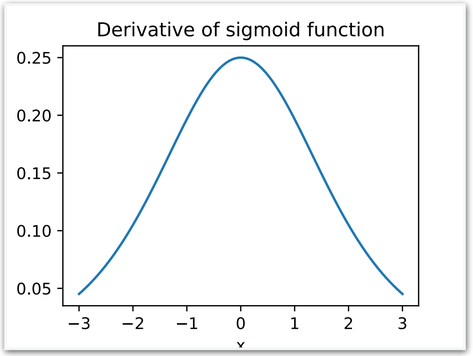
Vanishing gradients – A problem in *deep* neural nets

Problem:

- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing “simple” patterns, which are important to learning

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \frac{\partial C}{\partial a_4}$$



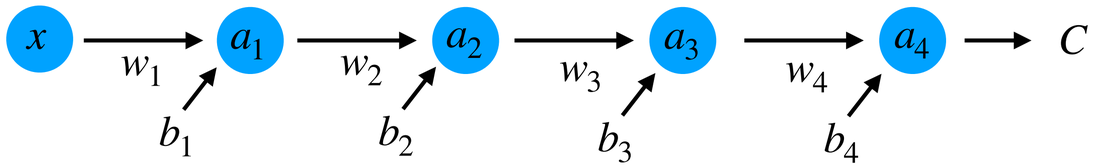


Vanishing gradients – A problem in *deep* neural nets

Problem:

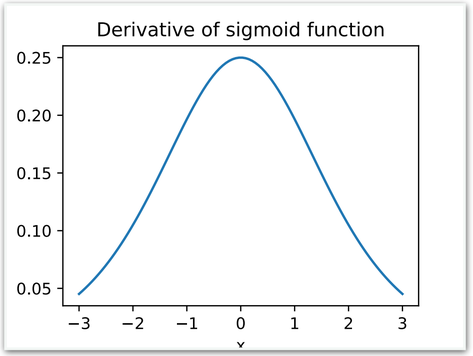
- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing "simple" patterns, which are important to learning

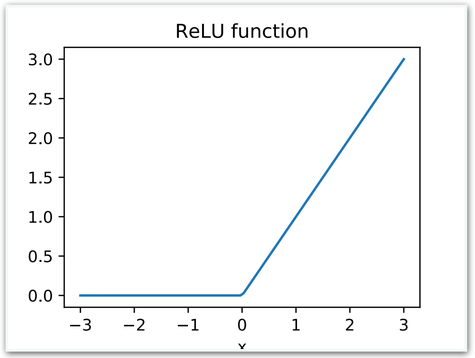
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \frac{\partial C}{\partial a_4}$$

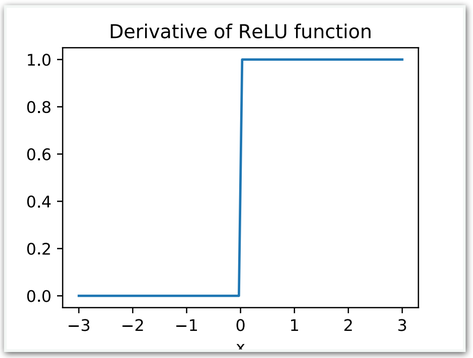


Solution:

- Use an activation function without small gradient for high values
- Candidate activation function: ReLU





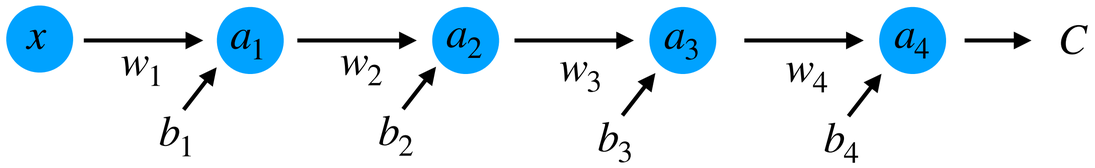


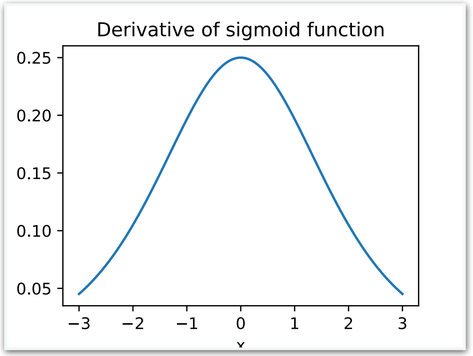
Vanishing gradients – A problem in *deep* neural nets

Problem:

- Gradients closer and closer to the input tend to get smaller and smaller
- Leads to smaller weight updates near input and larger weight updates near output
- Bad because layers near input take part in recognizing "simple" patterns, which are important to learning

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \frac{\partial C}{\partial a_4}$$



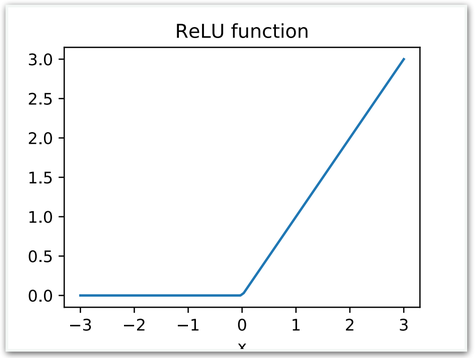


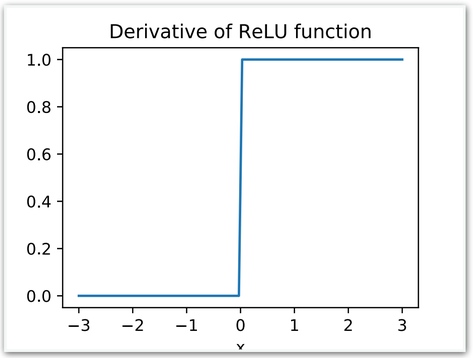
Solution:

- Use an activation function without small gradient for high values
- Candidate activation function: ReLU

Problems with ReLU:

- Exploding gradients!





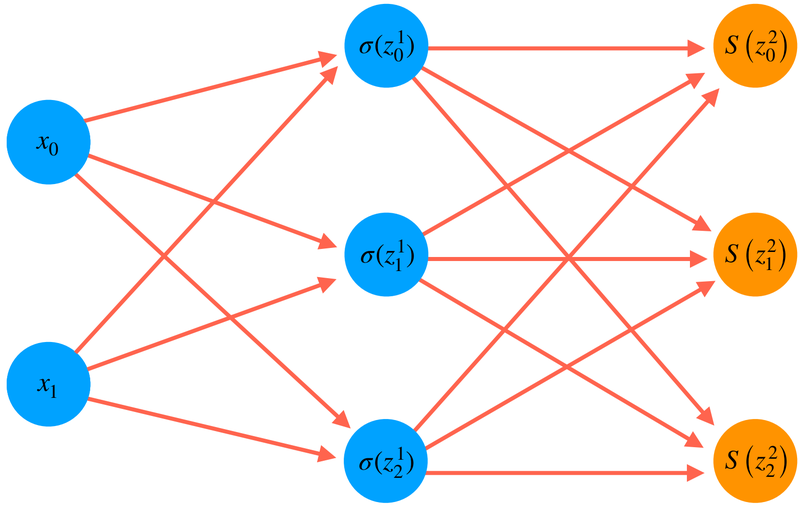
Solution:

- Batch normalization, gradient clipping, weight regularization

A new activation function for the output layer – The Softmax function

$$S(z_i^L) = \frac{e^{z_i^L}}{\sum_j e^{z_j^L}}$$

- Used in the last layer for ***multi-class classification***
- Ensures output neurons sum to 1
- Output is interpreted as a discrete probability distribution



A new (superior) loss function for classification – *Cross-entropy*

Binary classification problem

Ground truth: y

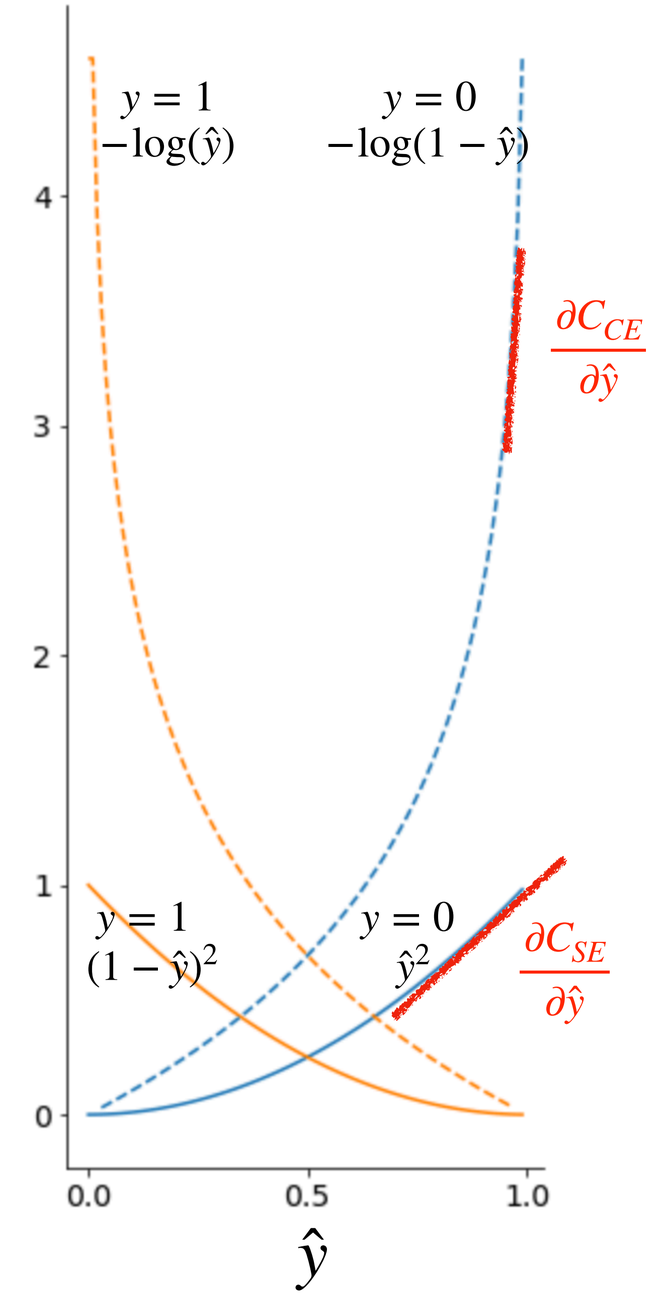
Prediction: $\hat{y}$

Squared error: $C = (y - \hat{y})^2$

Cross-entropy: $C = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$

Cross-entropy penalizes classification mistakes more!

$$\frac{\partial C_{SE}}{\partial w} = \frac{\partial C_{SE}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w}, \qquad \frac{\partial C_{CE}}{\partial w} = \frac{\partial C_{CE}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w}$$



Binary classification problem

Ground truth: y

Prediction: $\hat{y}$

Squared error: $C = (y - \hat{y})^2$

Cross-entropy: $C = -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$

Binary classification problem

Output layer activation: *sigmoid*

Cost function: *cross entropy*

Multiclass classification problem

Output layer activation: *softmax*

Cost function: *categorical cross entropy*