# CS 453X: Class 7

Jacob Whitehill

# Model complexity

# Model complexity

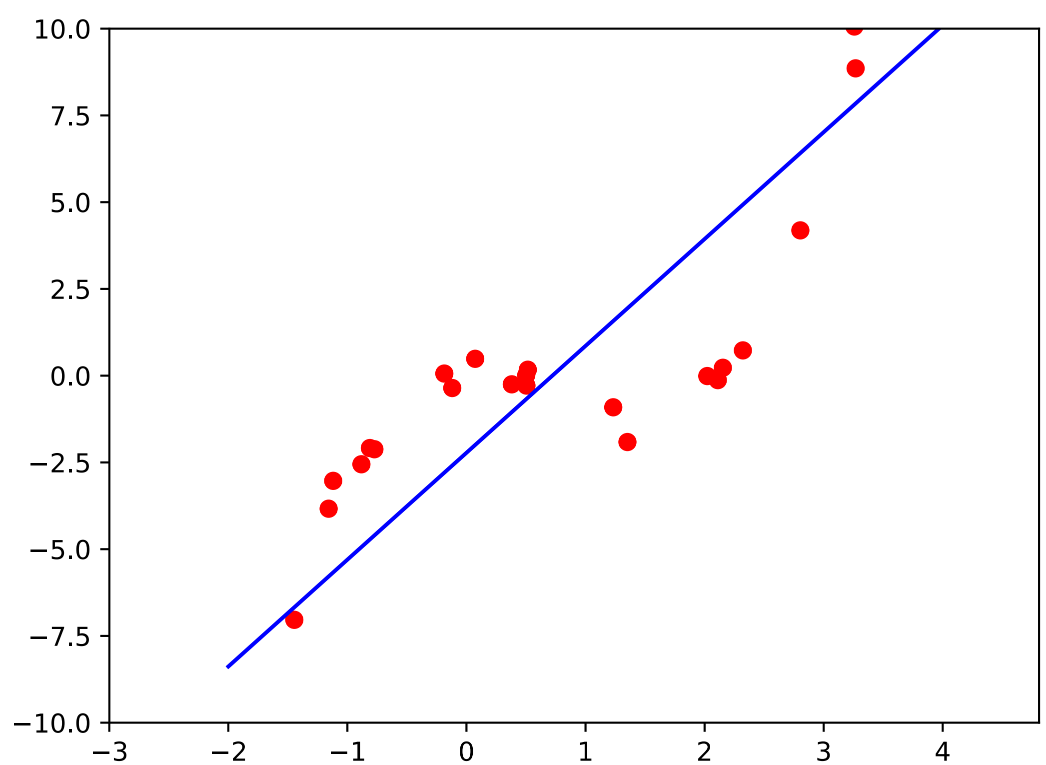
- With polynomial regression, we saw an easy way to increase the complexity of our ML model.
  - With higher degree *d*, our model becomes strictly more powerful.
  - With larger coefficients, the regression line becomes more flexible.

# Model complexity

- ML models can fail (i.e., exhibit poor accuracy) due to two reasons:

  1. **Bias**: the model is too simple to fit the data distribution ==> underfitting.

     - This can result in low *training* accuracy.



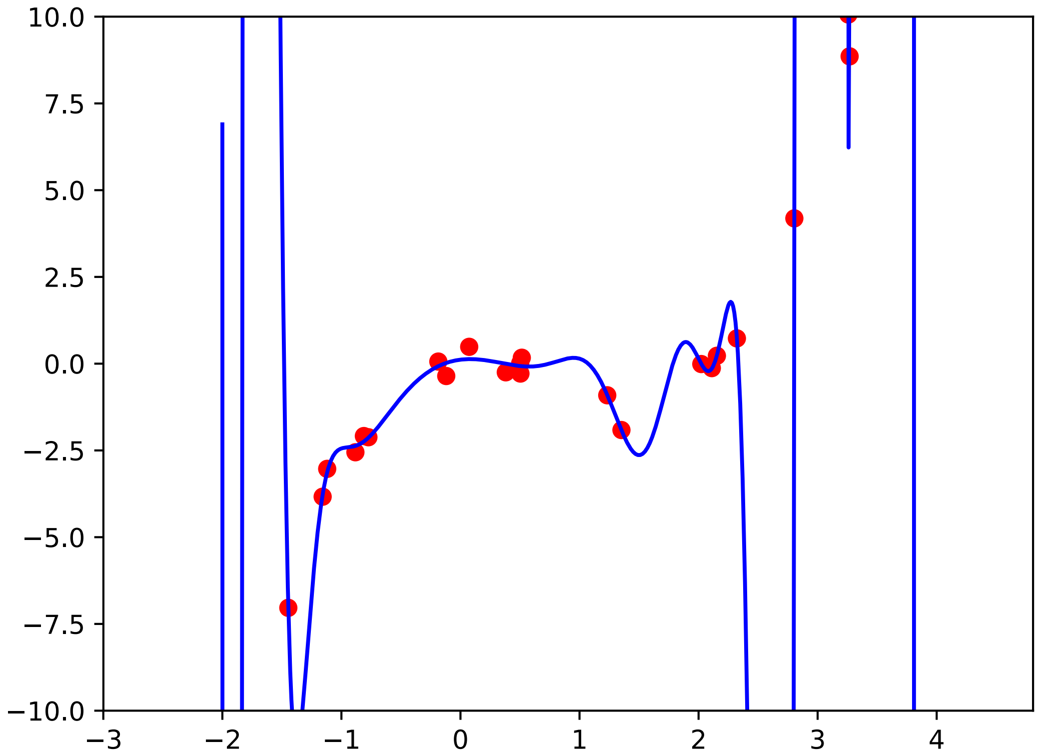

underfitting

# Model complexity

- ML models can fail (i.e., exhibit poor accuracy) due to two reasons:

  2. **Variance**: the model is too complex and is prone to overfitting. Re-training on different datasets will result in very different weight values.

  - This can result in low *testing* accuracy.




overfitting

# Model complexity

- Note that the degree of polynomial regression is just one kind of model complexity.
- Others:
  - Size of the input (24x24? 36x36?) to the machine.
  - Number of layers in a neural network (more later).

# Model complexity

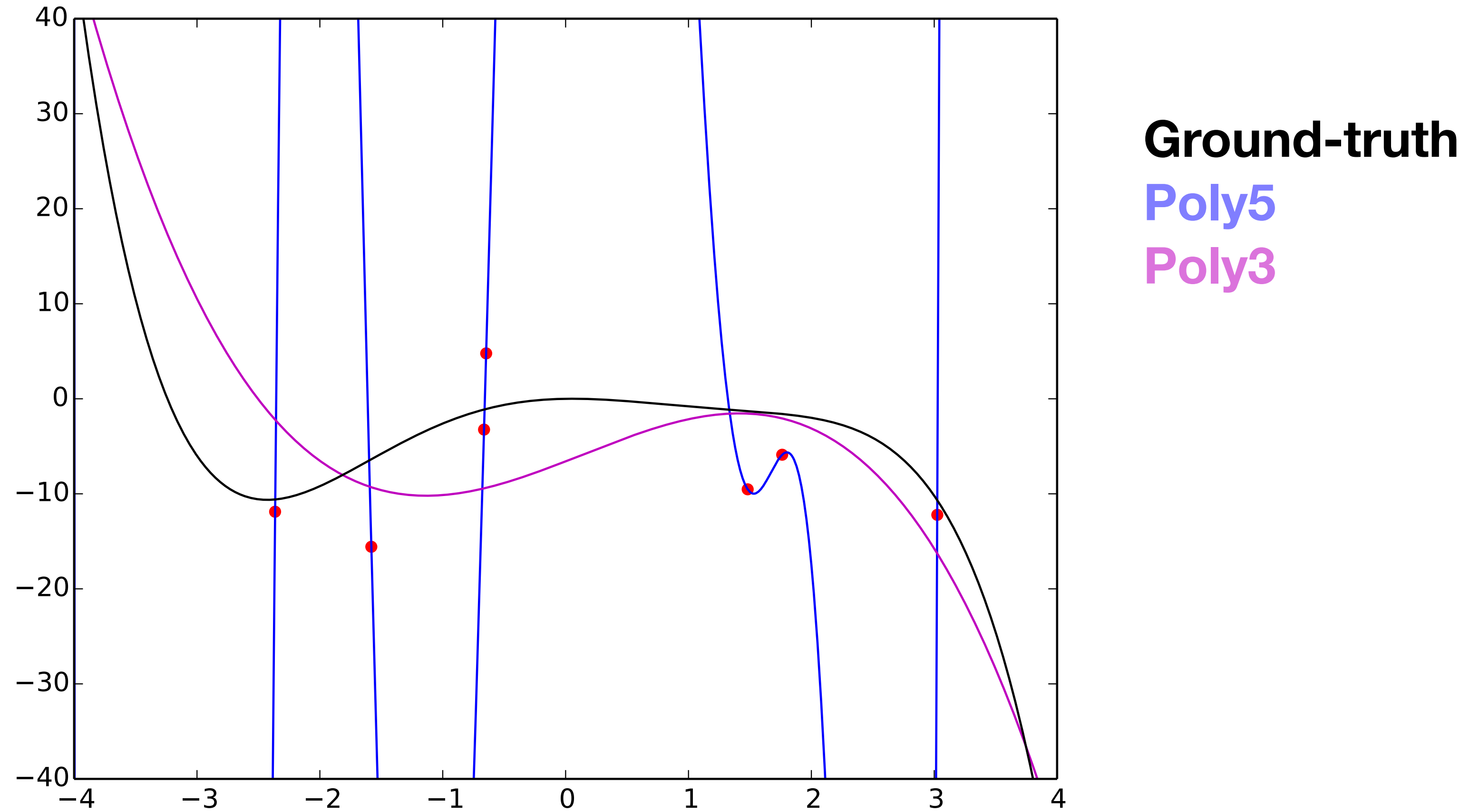
- In general: the more training data you have…
  - …the higher will be the testing accuracy of your trained machine.
  - …the more complex of a model you can use without overfitting.
  - … the less you need to regularize.

# Model complexity

- In general: the more training data you have…
  - …the higher will be the testing accuracy of your trained machine.
  - …the more complex of a model you can use without overfitting.
  - … the less you need to regularize.
- Therefore, as your training dataset grows, you might decide to switch to a more powerful architecture.

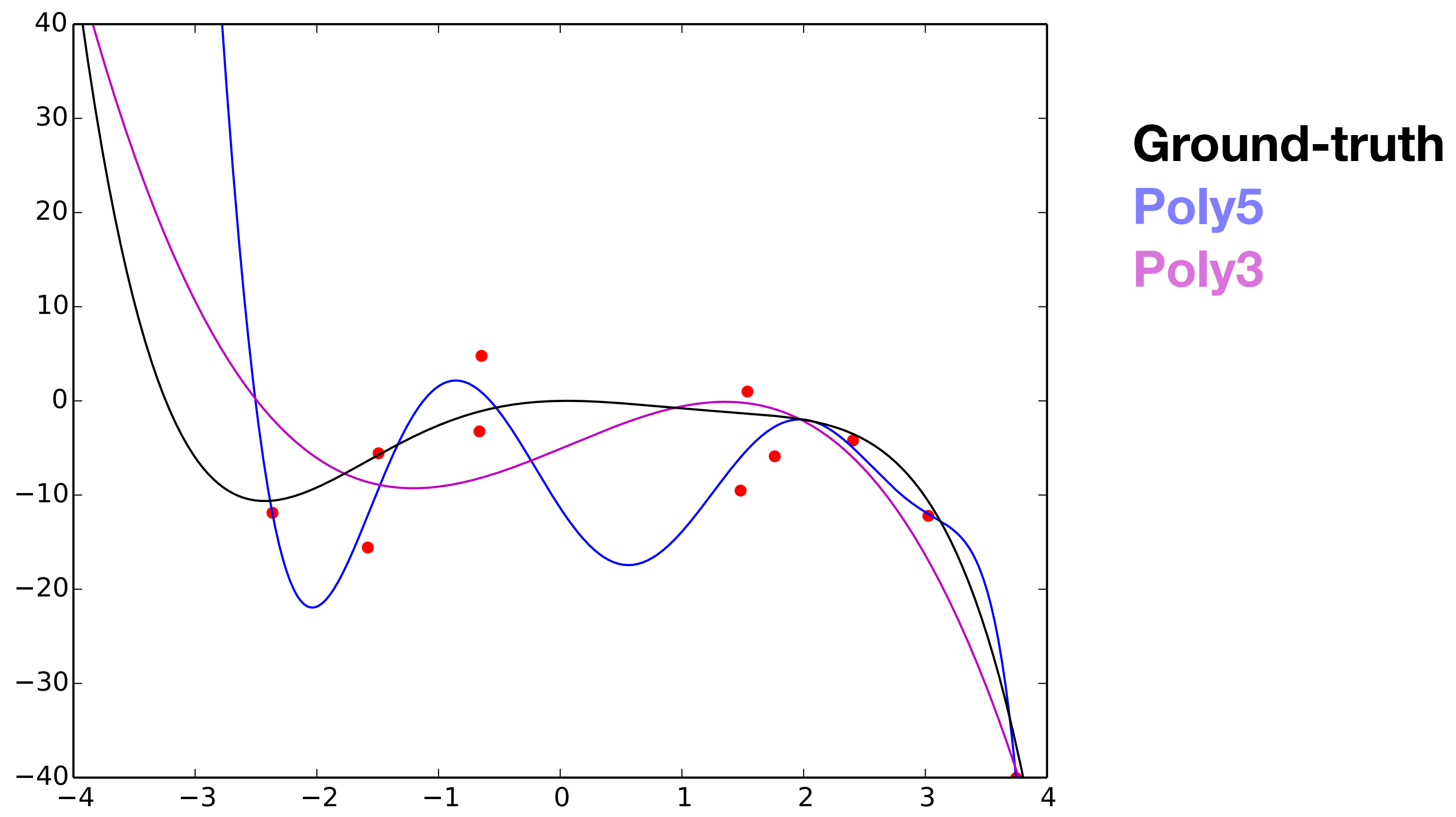
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



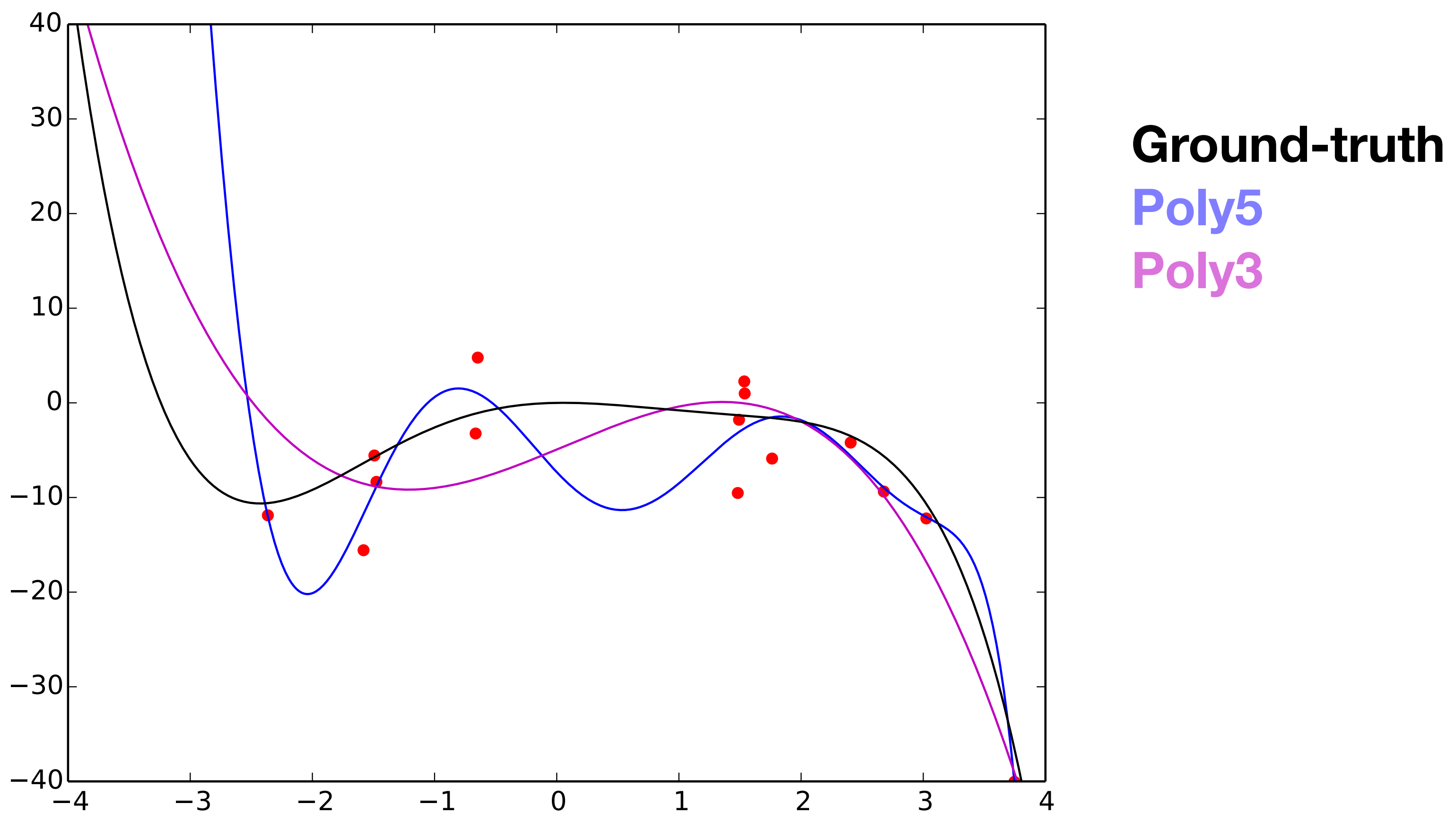
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



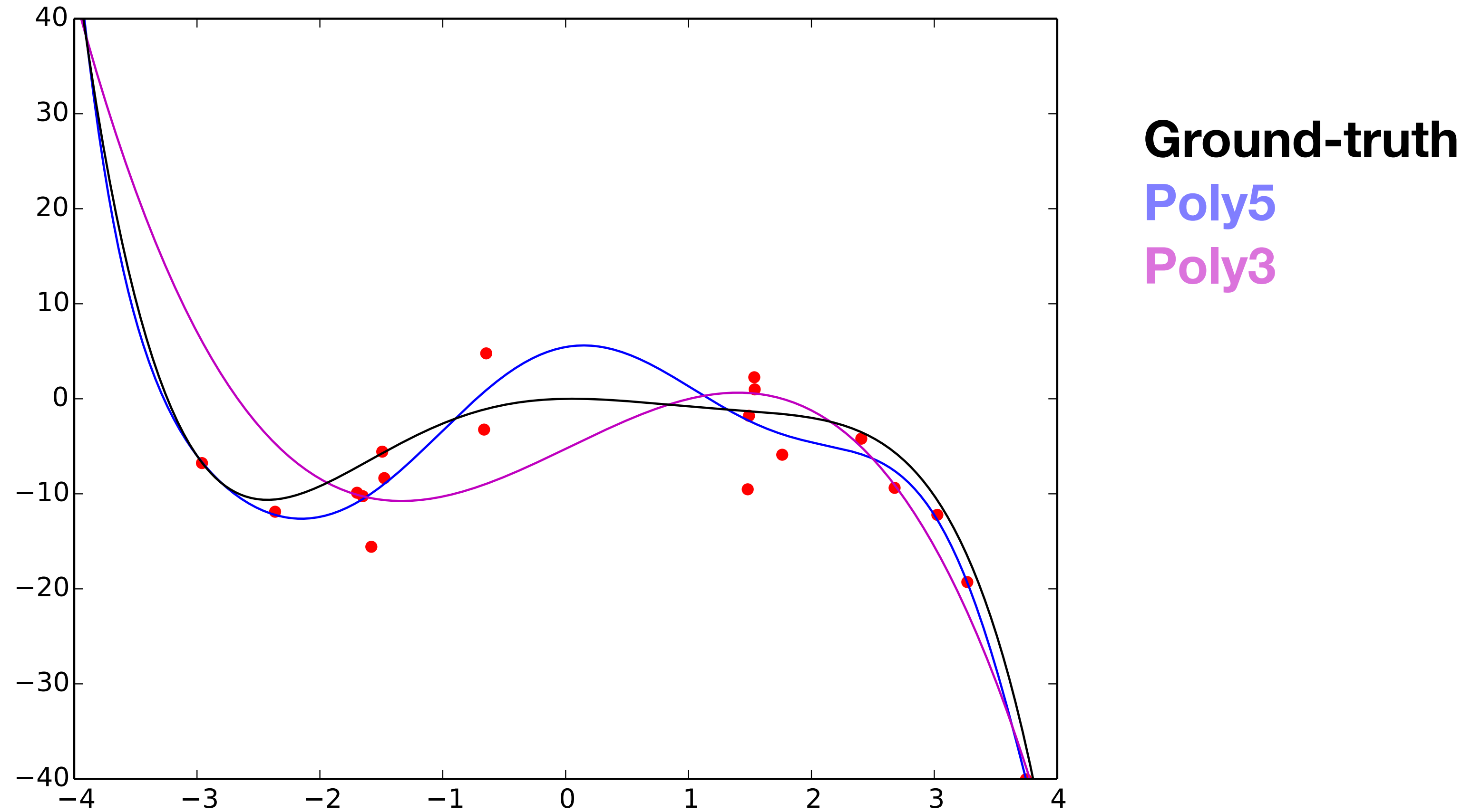
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



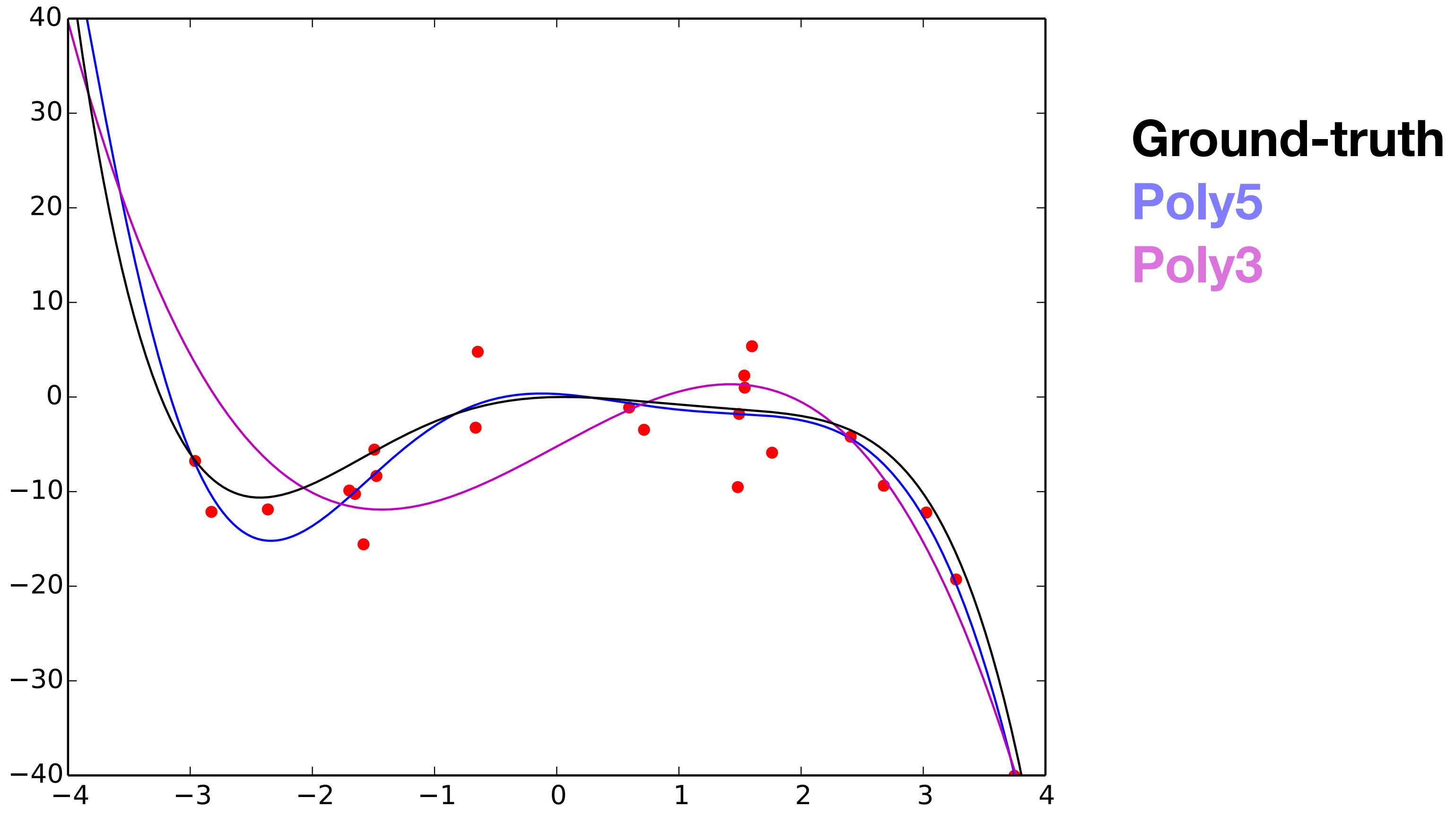
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



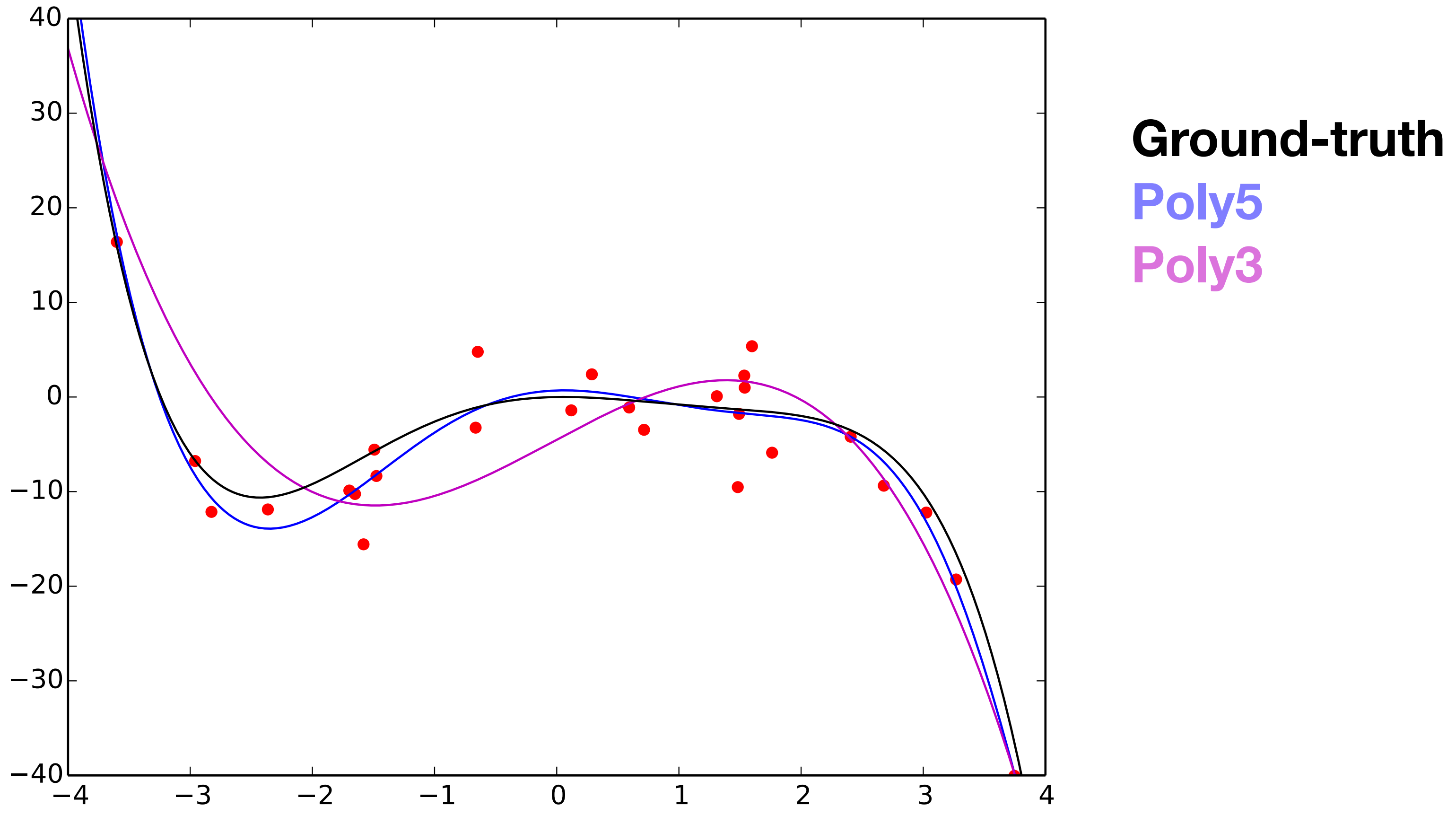
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



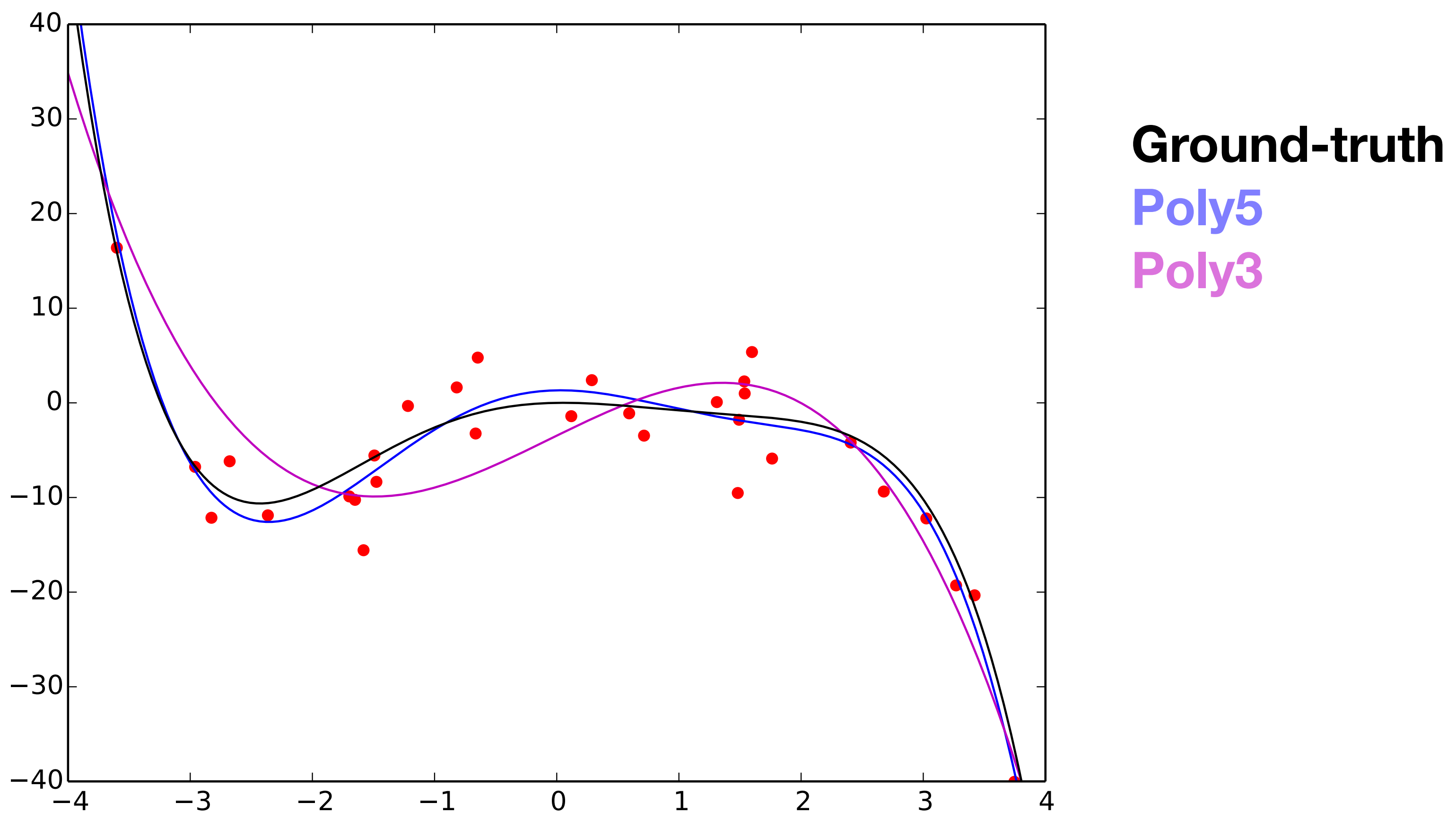
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



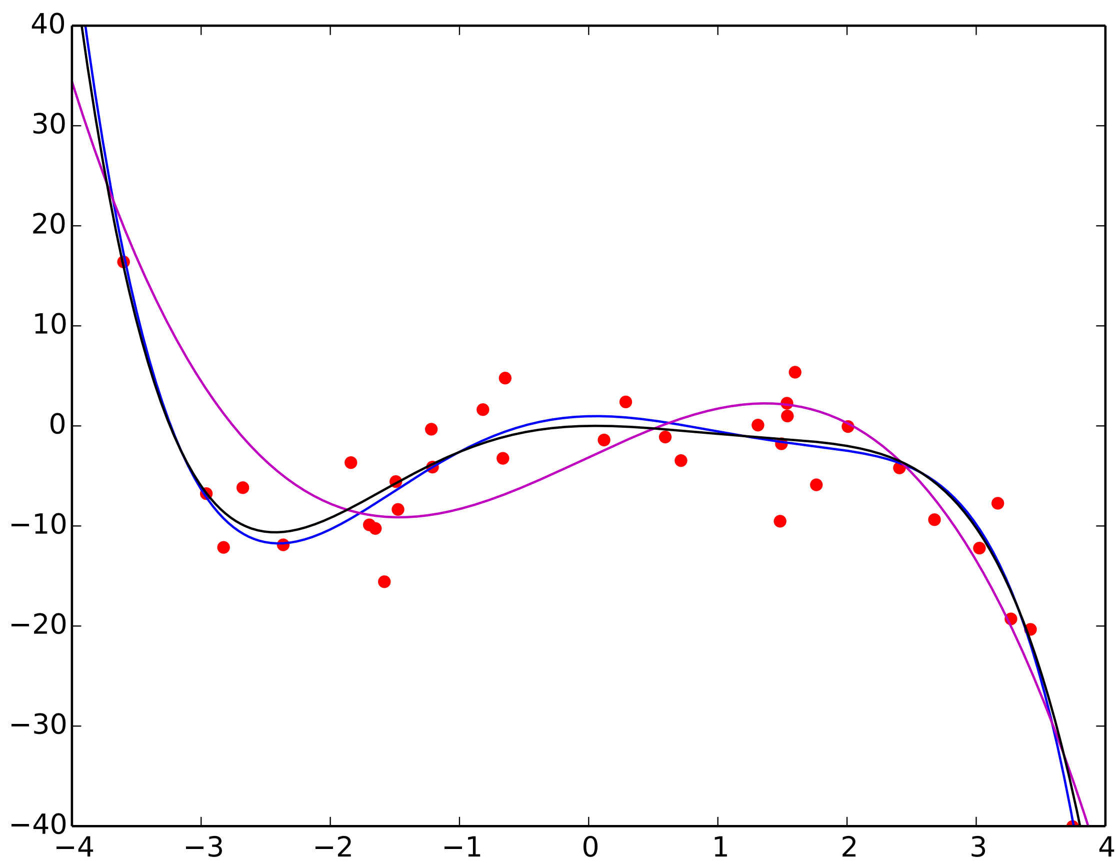
# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.



# Illustration

- Simulation:
  - Ground-truth: $y = -0.1x^5 + 0.1x^4 + 0.8x^3 - 1.8x^2 + 0.2x$
  - At each round, we add 4 more data points.
  - We compare polynomial regressors of degree 3 and 5.




**Ground-truth**
**Poly5**
**Poly3**

**By this point, the poly5 regressor is better than the poly3 regressor.**

# Logistic regression

# Using linear regression for classification

- In homework 2, you are using linear regression for classification.
    - **Regression**: predict any real number.
    - **Classification**: choose from a finite set (e.g., {0, 1}).
- While not incorrect, this is somewhat unnatural.

# Using linear regression for classification

- During training, we penalize the linear regression model based on the MSE:

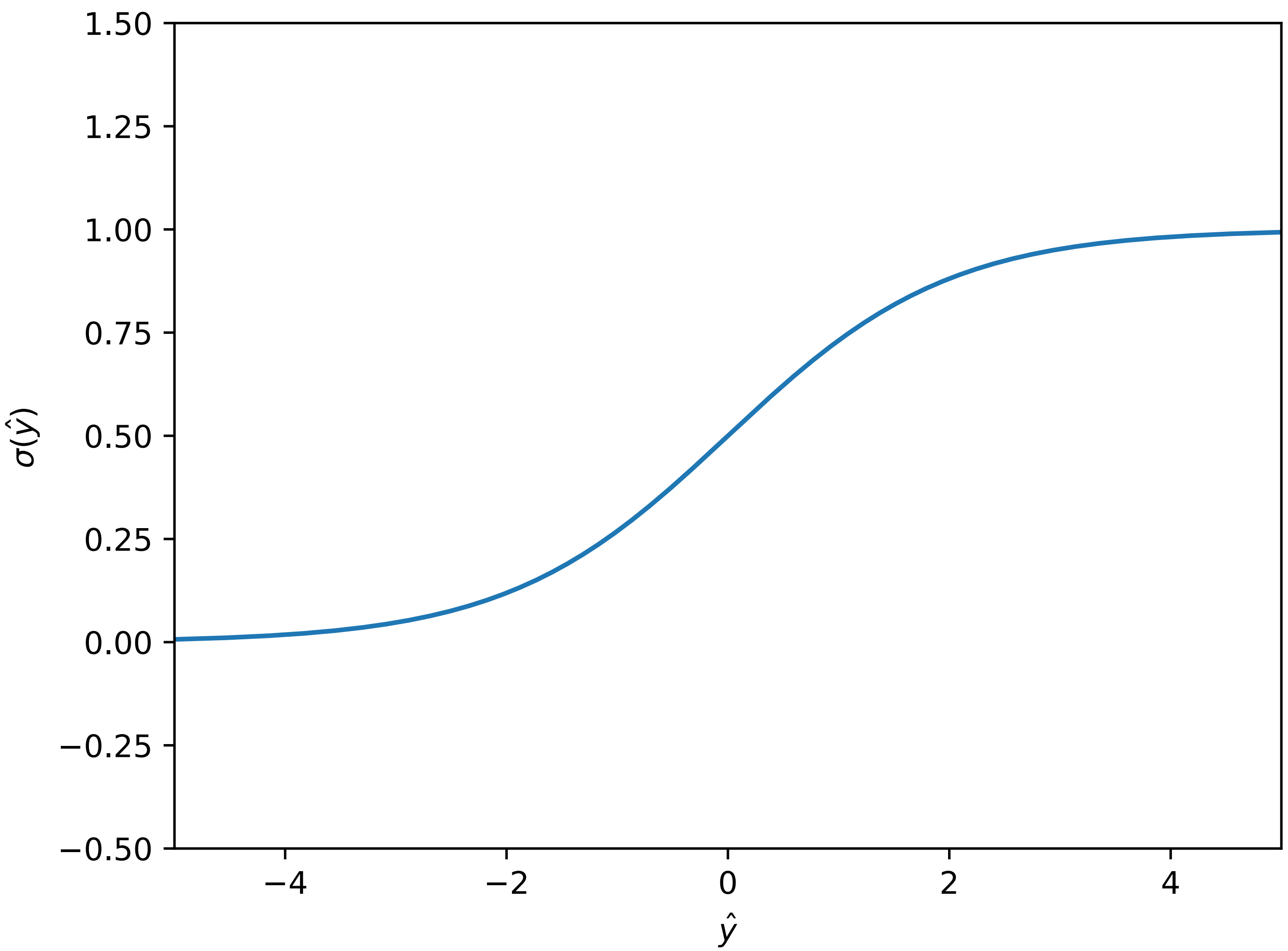
$$\frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

- Since every *y* is either 1 or 0, why let $\hat{y}$ ever be greater than 1 or less than 0?

- Why not "squash" the output to always lie in (0,1)?

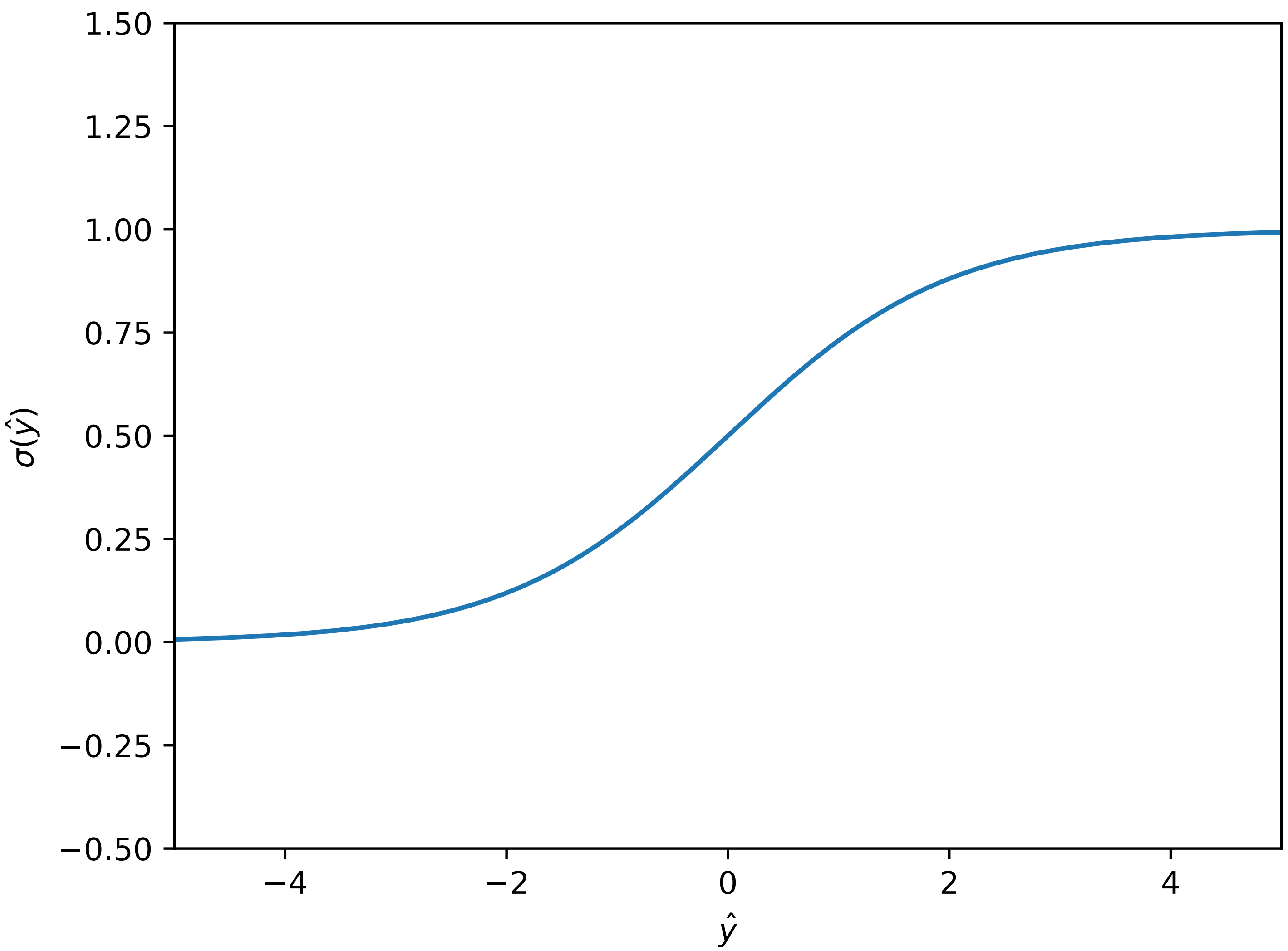
# Sigmoid: a "squashing" function

- A sigmoid function is an "s"-shaped, monotonically increasing and bounded function.
- Here is the **logistic sigmoid** function σ:



# Sigmoid: a "squashing" function

- A sigmoid function is an "s"-shaped, monotonically increasing and bounded function.
- Here is the **logistic sigmoid** function σ:



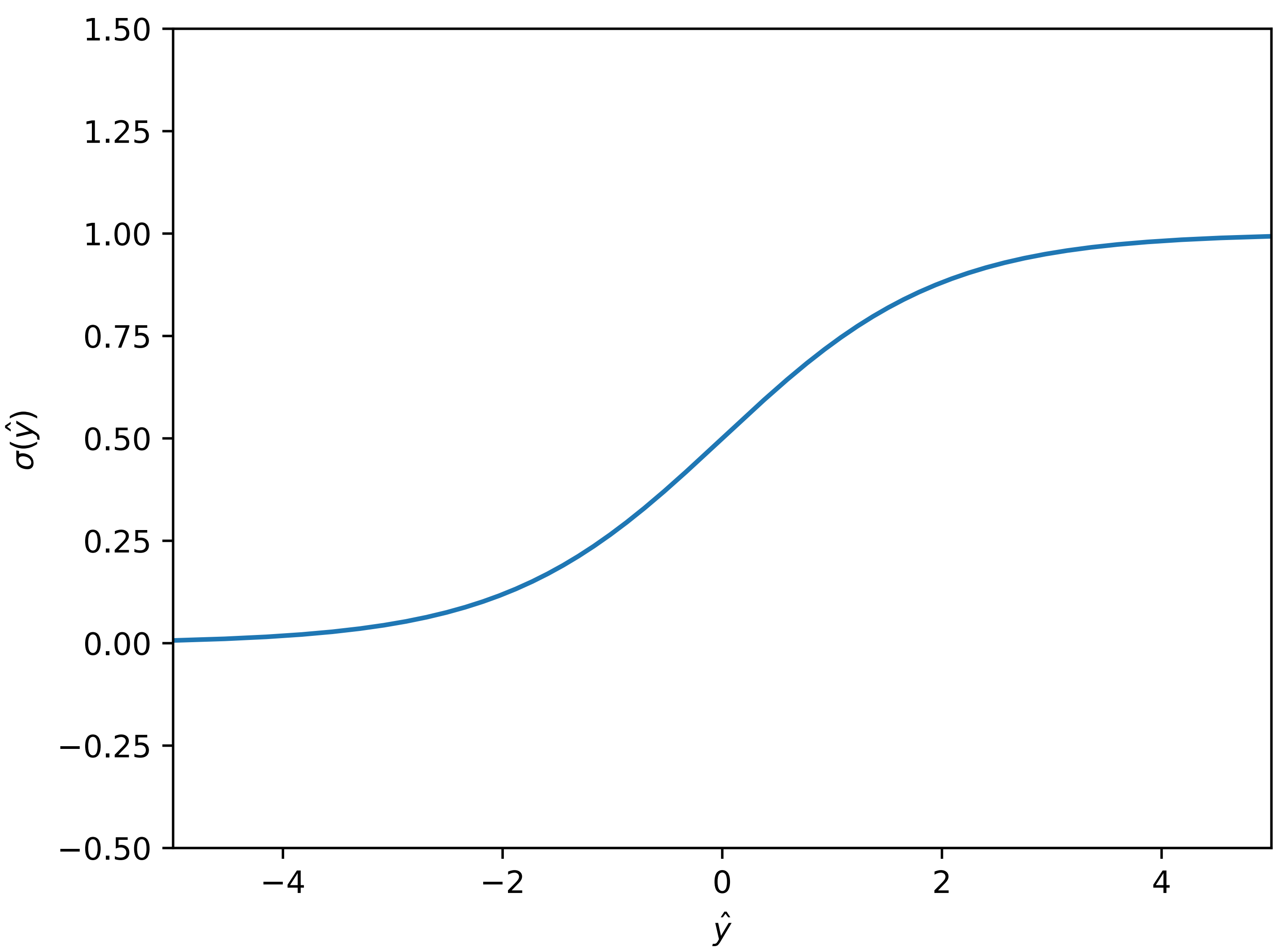

**Which function(s) describe(s) the curve?**

1. $$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$
2. $$\frac{e^x + e^{-x}}{e^x - e^{-x}}$$
3. $$\frac{1}{1 - e^{-x}}$$
4. $$\frac{1}{1 + e^{-x}}$$

# Sigmoid: a "squashing" function

- A sigmoid function is an "s"-shaped, monotonically increasing and bounded function.

- Here is the **logistic sigmoid** function σ:




**Which function(s) describe(s) the curve?**

**1.** $$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$ **tanh**

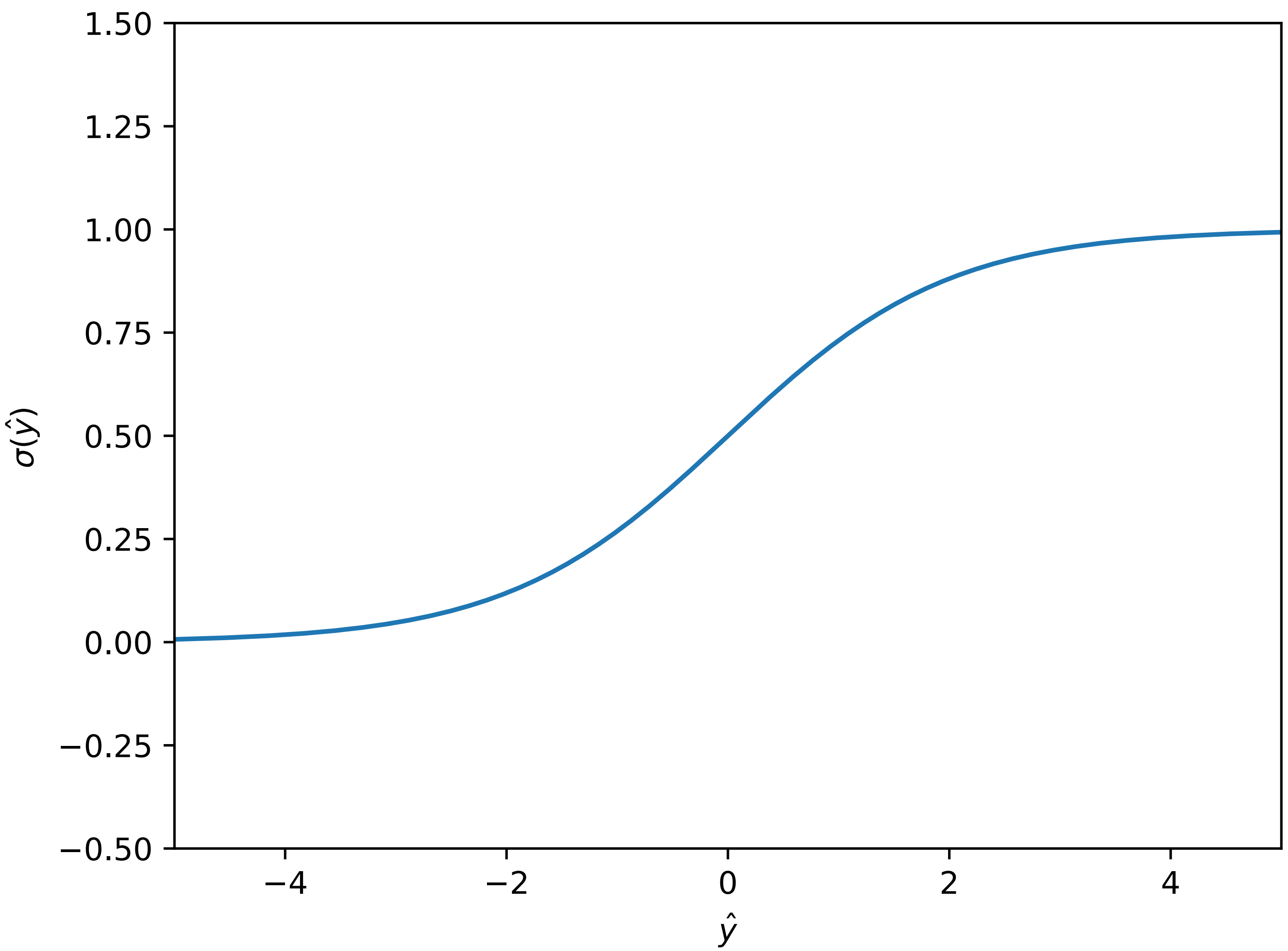
**Similar but not quite right.**

**4.** $$\frac{1}{1 + e^{-x}}$$

# Sigmoid: a “squashing” function

- A sigmoid function is an “s”-shaped, monotonically increasing and bounded function.

- Here is the **logistic sigmoid** function σ:




**Which function(s) describe(s) the curve?**

**4.** $$\frac{1}{1+e^{-x}}$$ **σ**

# Logistic sigmoid

- The logistic sigmoid function σ has some nice properties:
  - σ(-*z*) = 1 - σ(*z*)

$$
\begin{aligned}
\sigma(z) &= \frac{1}{1+e^{-z}} \\
1-\sigma(z) &= 1-\frac{1}{1+e^{-z}} \\
&= \frac{1+e^{-z}}{1+e^{-z}}-\frac{1}{1+e^{-z}} \\
&= \frac{e^{-z}}{1+e^{-z}} \\
&= \frac{1}{1/e^{-z}+1} \\
&= \frac{1}{1+e^{z}} \\
&= \sigma(-z)
\end{aligned}
$$

# Logistic sigmoid

- The logistic sigmoid function σ has some nice properties:
    - σ'(z) = σ(z)(1 - σ(z))

$$
\begin{aligned}
\sigma(z) &= \frac{1}{1+e^{-z}} \\
\frac{\partial \sigma}{\partial z} = \sigma'(z) &= -\frac{1}{(1+e^{-z})^2}(e^{-z} \times (-1)) \\
&= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{e^{-z}}{1+e^{-z}} \times \frac{1}{1+e^{-z}} \\
&= \frac{1}{1/e^{-z}+1} \times \frac{1}{1+e^{-z}} \\
&= \frac{1}{1+e^{z}} \times \frac{1}{1+e^{-z}} \\
&= \sigma(z)(1-\sigma(z))
\end{aligned}
$$

# Logistic regression

- With logistic regression, our predictions are defined as:

$$\hat{y} = \sigma\left(\mathbf{x}^\top \mathbf{w}\right)$$

- Hence, they are forced to be in (0,1).

- For classification, we can interpret the real-valued outputs as probabilities that express how confident we are in a prediction, e.g.:

  - $\hat{y}$=0.95: very confident that the class is a smile.

  - $\hat{y}$=0.45: not very confident that the class is a non-smile.

# Logistic regression

- How to train? Unlike linear regression, logistic regression has no analytical solution.
    - We can use gradient descent instead.
    - We have to apply the chain-rule of differentiation to handle the sigmoid function.

# Gradient descent for logistic regression

- Let's compute the gradient of $f_{\text{MSE}}$ for logistic regression.
- For simplicity, we'll consider just a single example:

$$
\begin{aligned}
f_{\text{MSE}}(\mathbf{w}) &= \frac{1}{2}(\hat{y} - y)^2 \\
&= \frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2 \\
\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2\right] \\
&= \left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)
\end{aligned}
$$

# Gradient descent for logistic regression

- Let's compute the gradient of $f_{\text{MSE}}$ for logistic regression.
- For simplicity, we'll consider just a single example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(\mathbf{w}) &= \frac{1}{2}(\hat{y} - y)^2 \\
&= \frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2 \\
\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2\right] \\
&= \left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right) \sigma(\mathbf{x}^\top \mathbf{w}) \left(1 - \sigma(\mathbf{x}^\top \mathbf{w})\right)
\end{aligned}
$$

# Gradient descent for logistic regression

- Let's compute the gradient of $f_{\mathrm{MSE}}$ for logistic regression.
- For simplicity, we'll consider just a single example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(\mathbf{w}) &= \frac{1}{2}(\hat{y} - y)^2 \\
&= \frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2 \\
\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2\right] \\
&= \mathbf{x}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)\sigma(\mathbf{x}^\top \mathbf{w})\left(1 - \sigma(\mathbf{x}^\top \mathbf{w})\right)
\end{aligned}
$$

# Gradient descent for logistic regression

- Let's compute the gradient of $f_{\text{MSE}}$ for logistic regression.

- For simplicity, we'll consider just a single example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(\mathbf{w}) &= \frac{1}{2}(\hat{y} - y)^2 \\
&= \frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2 \\
\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2\right] \\
&= \mathbf{x}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)\sigma(\mathbf{x}^\top \mathbf{w})\left(1 - \sigma(\mathbf{x}^\top \mathbf{w})\right) \\
&= \mathbf{x}\,(\hat{y} - y)\,\hat{y}\,(1 - \hat{y})
\end{aligned}
$$

# Gradient descent for logistic regression

- Let's compute the gradient of $f_{\text{MSE}}$ for logistic regression.
- For simplicity, we'll consider just a single example:

$$
\begin{aligned}
f_{\text{MSE}}(\mathbf{w}) &= \frac{1}{2}(\hat{y} - y)^2 \\
&= \frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2 \\
\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right)^2\right] \\
&= \mathbf{x}\left(\sigma(\mathbf{x}^\top \mathbf{w}) - y\right) \sigma(\mathbf{x}^\top \mathbf{w}) \left(1 - \sigma(\mathbf{x}^\top \mathbf{w})\right) \\
&= \mathbf{x}\,(\hat{y} - y)\, \hat{y}\,(1 - \hat{y})
\end{aligned}
$$

**Notice the extra multiplicative terms compared to the gradient for *linear* regression: x($\hat{y}$ - $y$)**

# Attenuated gradient

- What if the weights **w** are initially chosen badly, so that $\hat{y}$ is very close to 1, even though $y = 0$ (or vice-versa)?
  - Then $\hat{y}(1 - \hat{y})$ is close to 0.
- In this case, the gradient:

$$\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) = \mathbf{x}\,(\hat{y} - y)\,\hat{y}\,(1 - \hat{y})$$

  will be very close to 0.

- If the gradient is 0, then no learning will occur!

# Different cost function

- For this reason, logistic regression is typically trained using a different cost function from $f_{\text{MSE}}$.

- One particularly well-suited cost function uses logarithms.

- Logarithms and the logistic sigmoid interact well:

$$\frac{\partial}{\partial \mathbf{w}} \left[ \log \sigma(\mathbf{x}^\top \mathbf{w}) \right] \quad =$$

# Different cost function

- For this reason, logistic regression is typically trained using a different cost function from $f_{\text{MSE}}$.
- One particularly well-suited cost function uses logarithms.
- Logarithms and the logistic sigmoid interact well:

$$\frac{\partial}{\partial \mathbf{w}}\left[\log \sigma(\mathbf{x}^\top \mathbf{w})\right] \quad = \quad \frac{1}{\sigma(\mathbf{x}^\top \mathbf{w})}\sigma(\mathbf{x}^\top \mathbf{w})\left(1 - \sigma(\mathbf{x}^\top \mathbf{w})\right)$$

# Different cost function

- For this reason, logistic regression is typically trained using a different cost function from $f_{\text{MSE}}$.

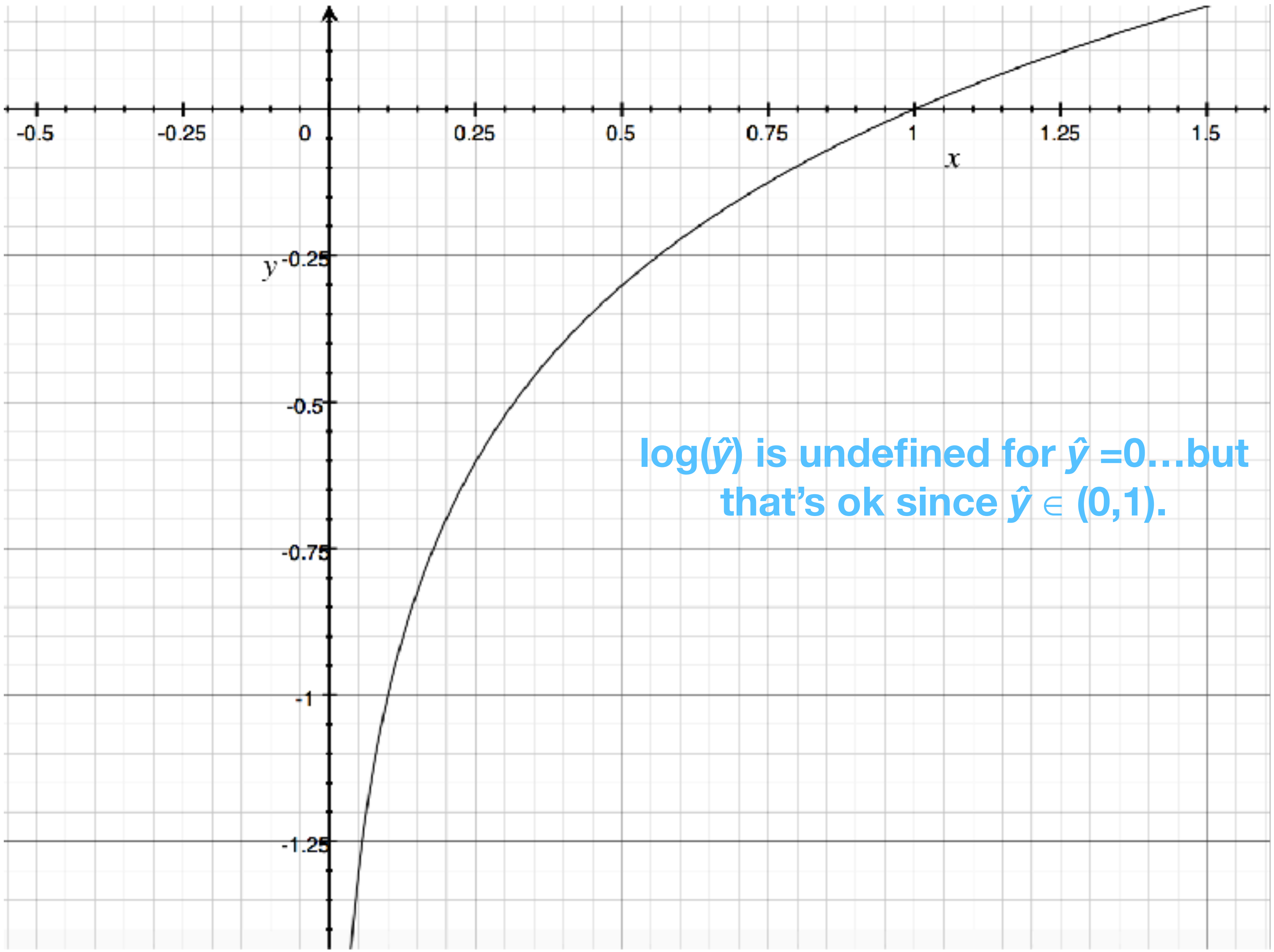
- One particularly well-suited cost function uses logarithms.

- Logarithms and the logistic sigmoid interact well:

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{w}}\left[\log \sigma(\mathbf{x}^\top \mathbf{w})\right] &= \frac{1}{\sigma(\mathbf{x}^\top \mathbf{w})}\sigma(\mathbf{x}^\top \mathbf{w})\left(1-\sigma(\mathbf{x}^\top \mathbf{w})\right) \\
&= 1-\sigma(\mathbf{x}^\top \mathbf{w})
\end{aligned}
$$

**The gradient of log(σ) is surprisingly simple.**

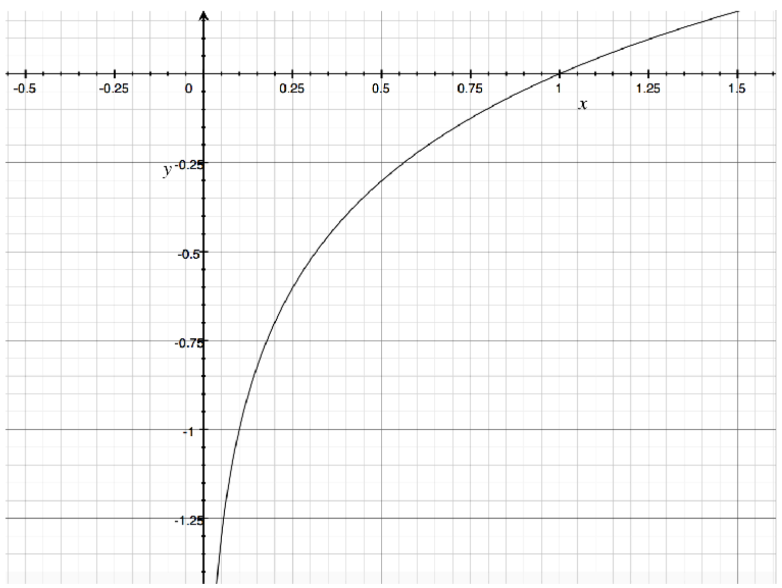
# Logarithm function



# Log loss

- How could we define a "log-loss" function $f_{\log}$ so that:
  - $f_{\log}(y, \hat{y})$ is small when $\hat{y} \approx y$ and large when they are far apart.

1. $-y \log \hat{y} - \hat{y} \log y$
2. $-y \log \hat{y} - (1-y) \log \hat{y}$
3. $-y \log \hat{y} - (1-y) \log (1-\hat{y})$
4. $-(1-y) \log \hat{y} - y \log (1-\hat{y})$



$\log(\hat{y})$

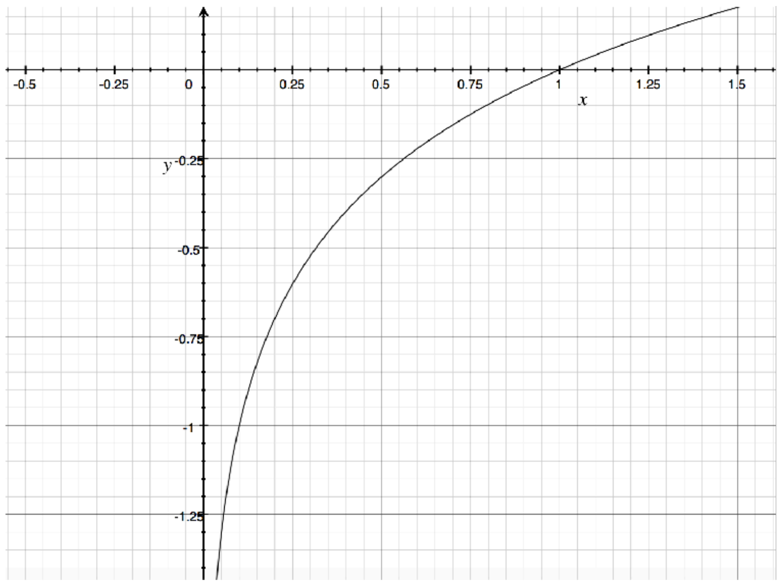
# Log loss

- How could we define a “log-loss” function $f_{\log}$ so that:
  - $f_{\log}(y, \hat{y})$ is small when $\hat{y} \approx y$ and large when they are far apart.

**This expression is known as the *log-loss*.**

**3.** $$-y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

**The *y* or (1-*y*) “selects” which term in the expression is active, based on the ground-truth label.**




**log($\hat{y}$)**

# Gradient descent for logistic regression with log-loss

$$\nabla_{\mathbf{w}} f_{\log}(\mathbf{w}) \quad = \quad \nabla_{\mathbf{w}} \left[ -\left( y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \right) \right]$$

# Gradient descent for logistic regression with log-loss

$$\begin{aligned}\nabla_{\mathbf{w}} f_{\log}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[- \left(y \log \hat{y} - (1-y)\log(1-\hat{y})\right)\right] \\ &= -\nabla_{\mathbf{w}} \left(y \log \sigma(\mathbf{x}^\top \mathbf{w}) + (1-y)\log(1-\sigma(\mathbf{x}^\top \mathbf{w}))\right)\end{aligned}$$

# Gradient descent for logistic regression with log-loss

$$
\begin{aligned}
\nabla_{\mathbf{w}} f_{\log}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[ -\left( y \log \hat{y} - (1-y) \log(1-\hat{y}) \right) \right] \\
&= -\nabla_{\mathbf{w}} \left( y \log \sigma(\mathbf{x}^\top \mathbf{w}) + (1-y) \log(1 - \sigma(\mathbf{x}^\top \mathbf{w})) \right) \\
&= -\left( y \frac{\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w})(1-\sigma(\mathbf{x}^\top \mathbf{w}))}{\sigma(\mathbf{x}^\top \mathbf{w})} - (1-y) \frac{\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w})(1-\sigma(\mathbf{x}^\top \mathbf{w}))}{1-\sigma(\mathbf{x}^\top \mathbf{w})} \right)
\end{aligned}
$$

# Gradient descent for logistic regression with log-loss

$$
\begin{aligned}
\nabla_{\mathbf{w}} f_{\log}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[ - \left( y \log \hat{y} - (1-y) \log(1-\hat{y}) \right) \right] \\
&= -\nabla_{\mathbf{w}} \left( y \log \sigma(\mathbf{x}^\top \mathbf{w}) + (1-y) \log(1 - \sigma(\mathbf{x}^\top \mathbf{w})) \right) \\
&= - \left( y \frac{\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w})(1 - \sigma(\mathbf{x}^\top \mathbf{w}))}{\sigma(\mathbf{x}^\top \mathbf{w})} - (1-y) \frac{\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w})(1 - \sigma(\mathbf{x}^\top \mathbf{w}))}{1 - \sigma(\mathbf{x}^\top \mathbf{w})} \right) \\
&= - \left( y\mathbf{x}(1 - \sigma(\mathbf{x}^\top \mathbf{w})) - (1-y)\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w}) \right)
\end{aligned}
$$

# Gradient descent for logistic regression with log-loss

$$
\begin{aligned}
\nabla_{\mathbf{w}} f_{\log}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[ -\left( y \log \hat{y} - (1-y) \log(1-\hat{y}) \right) \right] \\
&= -\nabla_{\mathbf{w}} \left( y \log \sigma(\mathbf{x}^\top \mathbf{w}) + (1-y) \log(1 - \sigma(\mathbf{x}^\top \mathbf{w})) \right) \\
&= -\left( y \frac{\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w})(1-\sigma(\mathbf{x}^\top \mathbf{w}))}{\sigma(\mathbf{x}^\top \mathbf{w})} - (1-y) \frac{\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w})(1-\sigma(\mathbf{x}^\top \mathbf{w}))}{1-\sigma(\mathbf{x}^\top \mathbf{w})} \right) \\
&= -\left( y\mathbf{x}(1-\sigma(\mathbf{x}^\top \mathbf{w})) - (1-y)\mathbf{x}\sigma(\mathbf{x}^\top \mathbf{w}) \right) \\
&= -\mathbf{x} \left( y - y\sigma(\mathbf{x}^\top \mathbf{w}) - \sigma(\mathbf{x}^\top \mathbf{w}) + y\sigma(\mathbf{x}^\top \mathbf{w}) \right) \\
&= -\mathbf{x} \left( y - \sigma(\mathbf{x}^\top \mathbf{w}) \right) \\
&= \mathbf{x}(\hat{y} - y)
\end{aligned}
$$

Same as for linear regression!

# Linear regression versus logistic regression

| | Linear regression | Logistic regression |
|---|---|---|
| **Primary use** | Regression | Classification |
| **Prediction ($\hat{y}$)** | $\hat{y} = \mathbf{x}^\mathsf{T}\mathbf{w}$ | $\hat{y} = \sigma(\mathbf{x}^\mathsf{T}\mathbf{w})$ |
| **Loss** | $f_{\text{MSE}}$ | $f_{\log}$ |
| **Gradient** | $\mathbf{x}(\hat{y} - y)$ | $\mathbf{x}(\hat{y} - y)$ |

# Linear regression versus logistic regression

| | Linear regression | Logistic regression |
|---|---|---|
| **Primary use** | Regression | Classification |
| **Prediction ($\hat{y}$)** | $\hat{y} = \mathbf{x}^\mathsf{T}\mathbf{w}$ | $\hat{y} = \sigma(\mathbf{x}^\mathsf{T}\mathbf{w})$ |
| **Loss** | $f_{\mathrm{MSE}}$ | $f_{\mathrm{log}}$ |
| **Gradient** | $\mathbf{x}(\hat{y} - y)$ | $\mathbf{x}(\hat{y} - y)$ |

- Logistic regression is used primarily for *classification* even though it's called "regression".
- Logistic regression is an instance of a **generalized linear model** — a linear model combined with a **link function** (e.g., logistic sigmoid).
  - In neural networks, link functions are typically called **activation functions**.

# Multi-class (“polychotomous”) classification

# Multi-class classification

- So far we have talked about classifying only 2 classes (e.g., smile versus non-smile).
    - This is sometimes called **binary classification**.
- But there are many settings in which multiple (>2) classes exist, e.g., emotion recognition, hand-written digit recognition:


6 classes (fear, anger, sadness, happiness, disgust, surprise)

10 classes (0-9)

# Classification versus regression

- Note that, even though the hand-written digit recognition ("MNIST") problem has classes called "0" , "1", …, "9", there is no sense of "distance" between the classes.

  - Misclassifying a 1 as a 2 is just as "bad" as misclassifying a 1 as a 9.

# Multi-class classification

- It turns out that logistic regression can easily be extended to support an arbitrary number (>2) of classes.
  - The multi-class case is called **softmax regression**.
- How to represent the ground-truth $y$ and prediction $\hat{y}$?
  - Instead of just a scalar $y$, we will use a vector $\mathbf{y}$.

# Example: 2 classes

- Suppose we have a dataset of 3 examples, where the ground-truth class labels are 0, 1, 0.

- Then we would define our ground-truth vectors as:

$$\mathbf{y}^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mathbf{y}^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{y}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- This is called **one-hot encoding** of the class label.