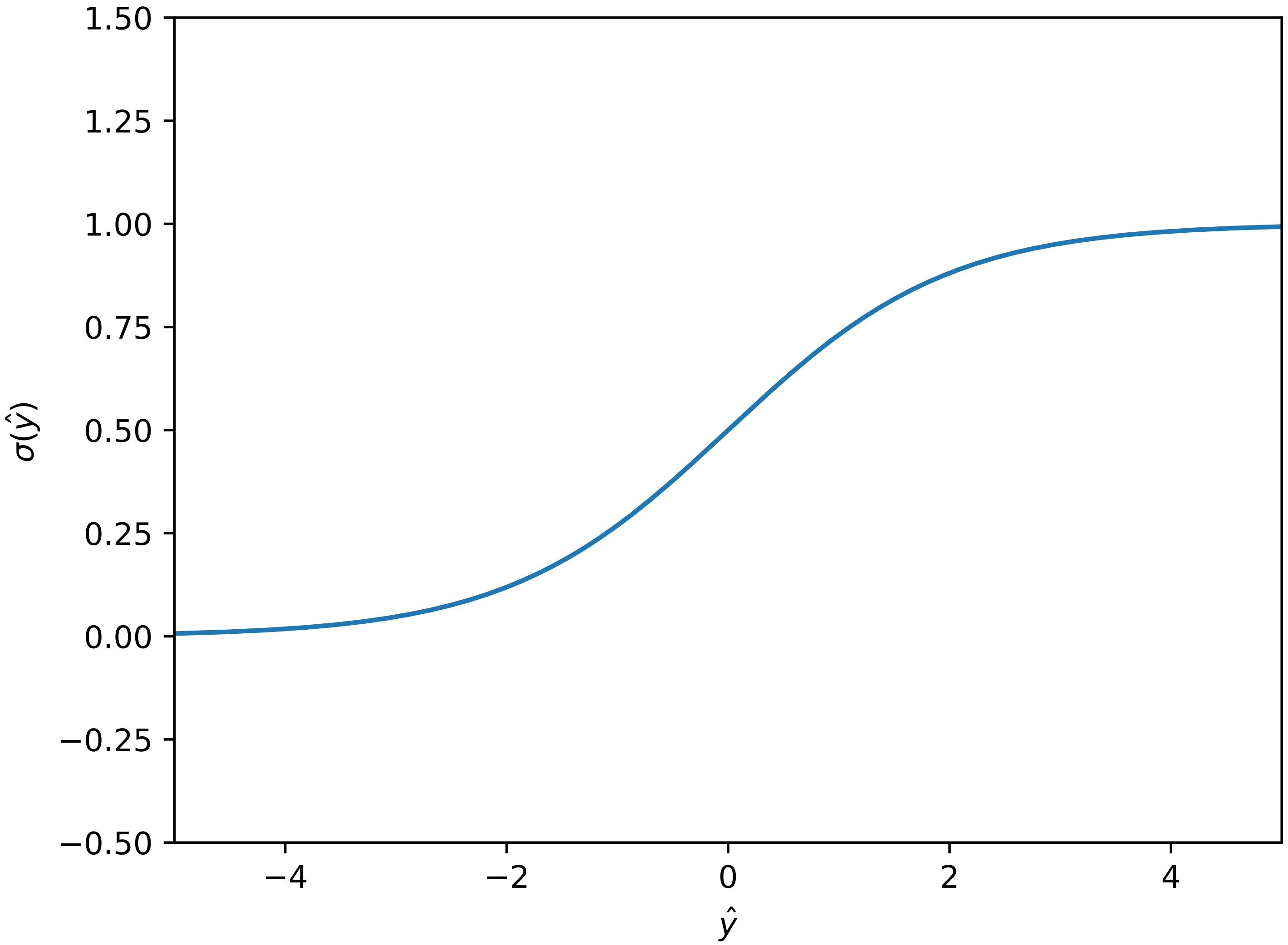
CS 4342: Class 7

Jacob Whitehill

Logistic regression

Sigmoid: a "squashing" function

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



$$\frac{1}{1+e^{-x}}$$

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.

Exercise

- Suppose we want to predict lung cancer from a person's exposure to radon r and asbestos a:
  - $y = 1$ if person develops lung cancer; $y = 0$ otherwise.
  - $\mathbf{x} = [a, r]^\top$, where:
    - a = kilograms of asbestos inhaled
    - r = average microCuries of radiation at home
  - Machine (with parameters $\mathbf{w}$): $\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w})$

Exercise

- Suppose we want to predict lung cancer from a person's exposure to radon *r* and asbestos *a*:
  - $y = 1$ if person develops lung cancer; $y = 0$ otherwise.
  - $\mathbf{x} = [a, r]^\mathsf{T}$, where:
    - a = kilograms of asbestos inhaled
    - r = average microCuries of radiation at home
  - Machine (with parameters $\mathbf{w}$): $\hat{y} = \sigma(\mathbf{x}^\mathsf{T}\mathbf{w})$
- Suppose we train the machine so that $\mathbf{w}=[1.5\ .22]^\mathsf{T}$.
- What is the machine's prediction for a person who inhales 2 grams of asbestos and whose home has an average of 4 microCuries?

Solution

- Just plug in values for **w** and **x** (making sure to convert from grams to kilograms) and then pass through σ:
    - $\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w}) = \sigma(.002*1.5 + 4*.22) = \sigma(0.883) = 0.707$.
- In other words, the person is predicted to have a 70.7% probability of getting lung cancer.

Logistic regression

- How to train? Unlike linear regression, logistic regression has no analytical ("one-shot") solution.
    - Could we use grid search (like in homework 1)?

Logistic regression

- How to train? Unlike linear regression, logistic regression has no analytical ("one-shot") solution.
  - Could we use grid search (like in homework 1)? No — intractable and unclear how to determine the grid.

Logistic regression

- How to train? Unlike linear regression, logistic regression has no analytical ("one-shot") 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_{MSE} for logistic regression.
- For simplicity, we'll consider just a single example:

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

Notice the extra multiplicative terms compared to the gradient for *linear* regression: x(*ŷ* - *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:

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

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_{MSE}.

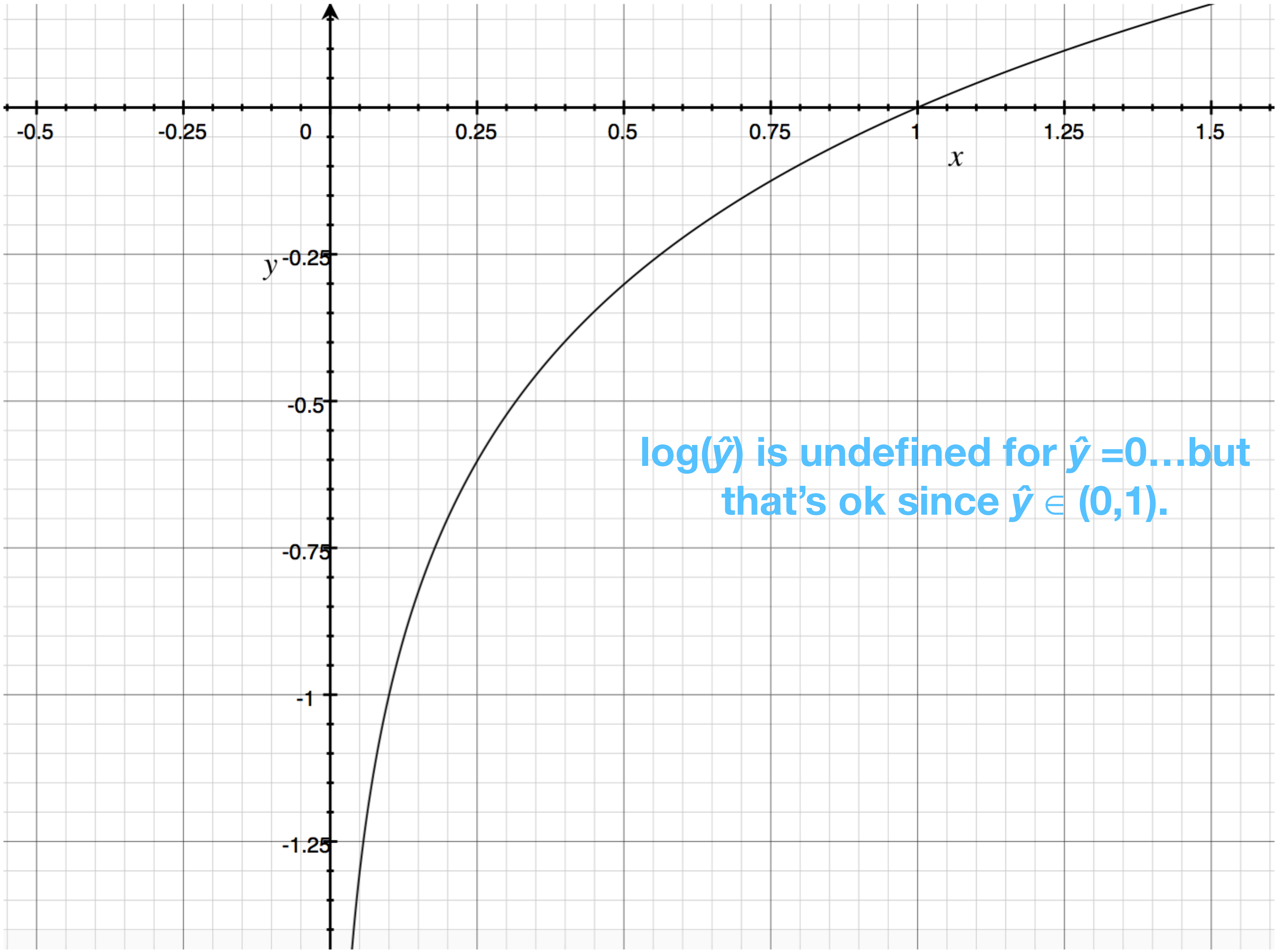
- One particularly well-suited cost function uses logarithms.

- Logarithms and the logistic sigmoid interact well:

$$\begin{aligned}\frac{\partial}{\partial z}\left[\log \sigma(z)\right] &= \frac{1}{\sigma(z)}\sigma'(z) \\ &= \frac{1}{\sigma(z)}\sigma(z)(1-\sigma(z)) \\ &= 1-\sigma(z)\end{aligned}$$

The gradient of log(σ) is surprisingly simple.

Logarithm function

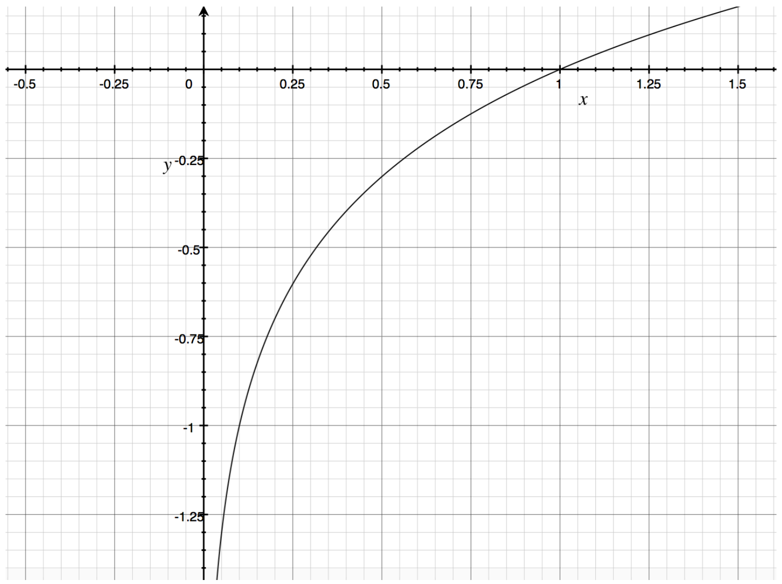


log($\hat{y}$) is undefined for $\hat{y}$ =0…but that's ok since $\hat{y} \in (0,1)$.

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.

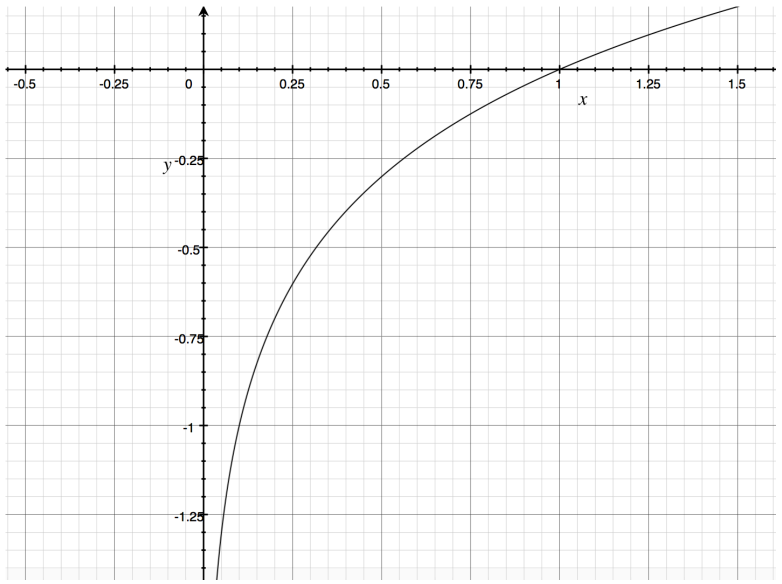
1. $-y \log \hat{y} - \hat{y} \log y$
2. $-y \log \hat{y} - (1-y) \log \hat{y}$
3. $-y \log \hat{y} - (1-y) \log(1-\hat{y})$
4. $-(1-y) \log \hat{y} - y \log(1-\hat{y})$



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.



log($\hat{y}$)

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.

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

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

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 ($\hat{y}$)	$\hat{y} = \mathbf{x}^\top \mathbf{w}$	$\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w})$
Loss	f_{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 ($\hat{y}$)	$\hat{y} = \mathbf{x}^\top \mathbf{w}$	$\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w})$
Loss	f_{MSE}	f_{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).

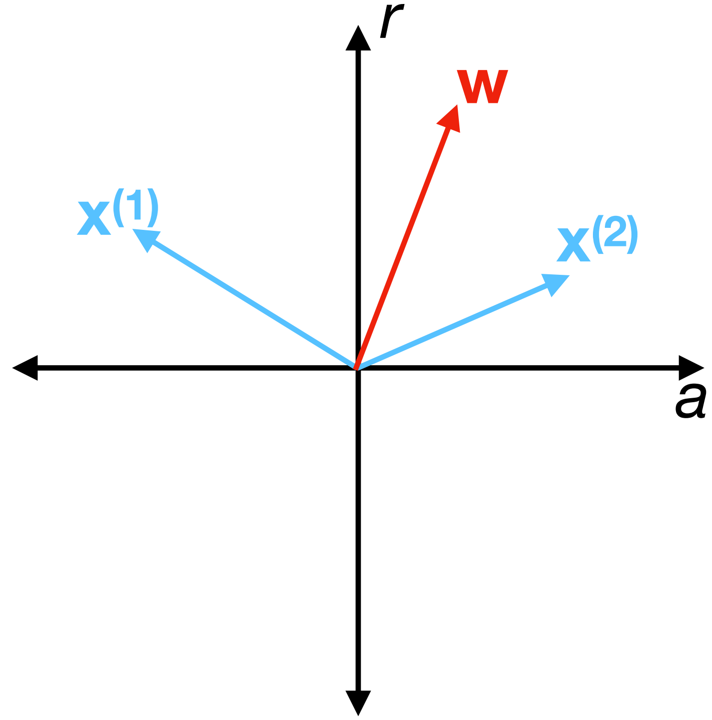
    - In neural networks, link functions are typically called **activation functions.**

Exercise

- Suppose we train a logistic regressor using f_{log}, and our training set contains only **positive** examples.
- As before, we let $\hat{y} = \sigma(\mathbf{x}^\top\mathbf{w})$ and $\mathbf{x} = [a, r]^\top$.
- What will/could happen during training? Explain your reasoning based on a specific dataset that you create (2 training examples should suffice).
- f_{log}: $-y \log \hat{y} - (1-y)\log(1-\hat{y})$

Solution

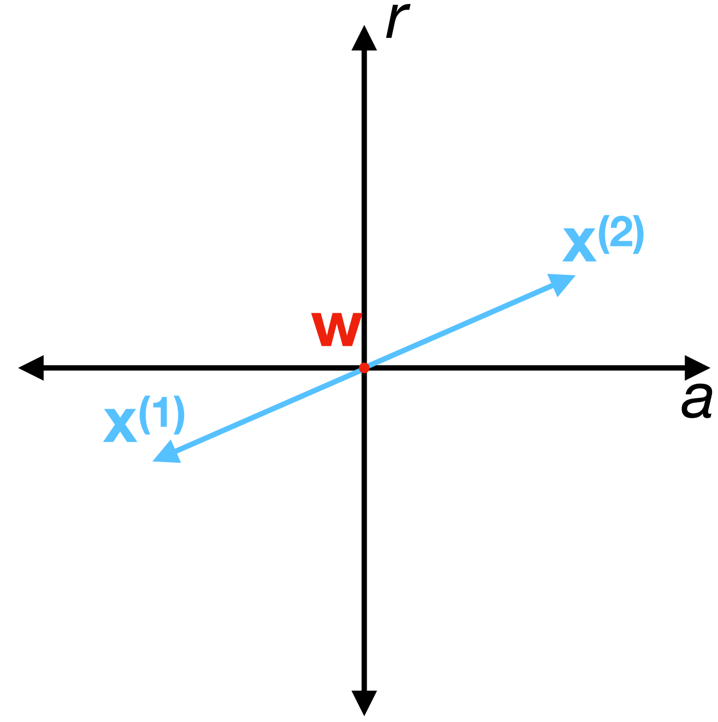
- Possibility 1: there exists a vector **w** with positive inner-product with *every* $\mathbf{x}^{(i)}$ in the training set, e.g.:



- In this case, $f_{\log}$ can be made arbitrarily small by making **w** be any vector with positive inner-product with the training examples.

Solution

- Possibility 2: there exists *no* vector **w** with positive inner-product with every $\mathbf{x}^{(i)}$ in the training set, e.g.:



- In this case, a best **w** may exist. For a dataset with 2 examples where $\mathbf{x}^{(1}=\mathbf{x}^{(2)}$, then the best **w** is **0**.

Exercise

- Now let's change our prediction model to be $\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w} + \mathrm{b})$ and $\mathbf{x} = [a, r]^\top$.

- What will/could happen now during training if all the training examples are positive?

Solution

- We can make f_{log} arbitrarily small by setting **w**=**0** and making b a large positive number.

- f_{log}: $-y \log \hat{y} - (1-y)\log(1-\hat{y})$

Softmax regression (aka multinomial logistic regression)

Multi-class classification

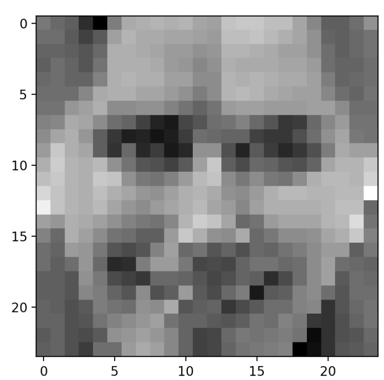
- So far we have talked about classifying only 2 classes (e.g., smile versus non-smile).
  - This is sometimes called **binary classification**.
- But there are many settings in which multiple (>2) classes exist, e.g., emotion recognition, hand-written digit recognition:

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

10 classes (0-9)

Multi-class classification

- In one form of multi-classification (over *c* classes), we map every input **x** into exactly 1 class:



x

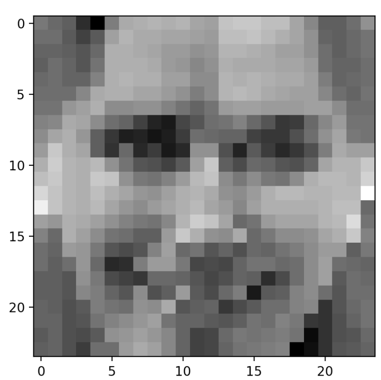
Multi-class machine

"joy"

$\hat{y} \in \{ 1, \ldots, c \}$

Multi-class classification

- In another, we map **x** into a **probability distribution** over the c classes:



x

Multi-class machine

"joy": 0.7
"anger": 0.05
"sadness": 0.1
…

$\hat{y} \in \{ 1, \ldots, c \}$

This is the approach we will use.

Classification versus regression

- Note that, in contrast to regression problems (e.g., age estimation), there is no sense of “distance” between classes:
    - Misclassifying a “joyful” face as “sad” is just as bad as misclassifying a “joyful” face as “angry”.

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** or sometimes **multinomial logistic 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 and 2 classes, where the ground-truth class labels are 0, 1, 0.
- Then we would define our ground-truth vectors as:

$$\mathbf{y}^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$\mathbf{y}^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\mathbf{y}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Exactly 1 coordinate of each **y** is 1; the others are 0.

Example: 2 classes

- Suppose we have a dataset of 3 examples and 2 classes, where the ground-truth class labels are 0, 1, 0.

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

$$\mathbf{y}^{(1)} = \begin{bmatrix} \boxed{1} \\ 0 \end{bmatrix} \longleftarrow \text{This “slot” is for class 0.}$$

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

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

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

Example: 2 classes

- Suppose we have a dataset of 3 examples and 2 classes, where the ground-truth class labels are 0, 1, 0.

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

$$\mathbf{y}^{(1)} = \begin{bmatrix} 1 \\ \boxed{0} \end{bmatrix} \longleftarrow \textbf{This "slot" is for class 1.}$$

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

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

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

Example: 2 classes

- The machine's predictions $\hat{\mathbf{y}}$ about each example's label are also **probabilistic**.

- They could consist of:

$$\hat{\mathbf{y}}^{(1)} = \begin{bmatrix} \boxed{0.93} \\ 0.07 \end{bmatrix} \longleftarrow \text{Machine's "belief" that the label is 0.}$$

$$\hat{\mathbf{y}}^{(2)} = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}$$

$$\hat{\mathbf{y}}^{(3)} = \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix}$$

- Each coordinate of $\hat{\boldsymbol{y}}$ is a probability.

Example: 2 classes

- The machine's predictions $\hat{\mathbf{y}}$ about each example's label are also **probabilistic**.

- They could consist of:

$$\hat{\mathbf{y}}^{(1)} = \begin{bmatrix} 0.93 \\ \boxed{0.07} \end{bmatrix}$$

Machine's "belief" that the label is 1.

$$\hat{\mathbf{y}}^{(2)} = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}$$

$$\hat{\mathbf{y}}^{(3)} = \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix}$$

- The sum of the coordinates in each $\hat{\mathbf{y}}$ is 1.

Cross-entropy loss

- We need a loss function that can support $c \geq 2$ classes.
- We will use the **cross-entropy** loss (aka **negative log-likelihood**):

$$f_{\mathrm{CE}} = -\sum_{i=1}^{n} \sum_{k=1}^{c} \mathbf{y}_k^{(i)} \log \hat{\mathbf{y}}_k^{(i)}$$

Cross-entropy loss

- Note that the f_{log} (for logistic regression) is a special case of f_{CE} (for softmax regression) for c=2.

- To see how, consider just a simple example:

$$f_{\text{CE}} \quad = \quad -\sum_{k=0}^{1} \mathbf{y}_k \log \hat{\mathbf{y}}_k$$

Cross-entropy loss

- Note that the $f_{\log}$ (for logistic regression) is a special case of f_{CE} (for softmax regression) for c=2.

- To see how, consider just a simple example:

$$f_{\mathrm{CE}} = -\sum_{k=0}^{1} \mathbf{y}_k \log \hat{\mathbf{y}}_k$$

Note: the sum from *k*=1 to *c* is renumbered from 0 to c-1.

Cross-entropy loss

- Note that the $f_{\log}$ (for logistic regression) is a special case of f_{CE} (for softmax regression) for c=2.

- To see how, consider just a simple example:

$$
\begin{aligned}
f_{\mathrm{CE}} &= -\sum_{k=0}^{1} \mathbf{y}_k \log \hat{\mathbf{y}}_k \\
&= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - \mathbf{y}_0 \log \hat{\mathbf{y}}_0
\end{aligned}
$$

Cross-entropy loss

- Note that the $f_{\log}$ (for logistic regression) is a special case of f_{CE} (for softmax regression) for c=2.

- To see how, consider just a simple example:

$$
\begin{aligned}
f_{\text{CE}} &= -\sum_{k=0}^{1} \mathbf{y}_k \log \hat{\mathbf{y}}_k \\
&= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - \mathbf{y}_0 \log \hat{\mathbf{y}}_0 \\
&= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - (\boxed{1 - \mathbf{y}_1}) \log(\boxed{1 - \hat{\mathbf{y}}_1})
\end{aligned}
$$

$$\hat{\mathbf{y}}^{(1)} = \begin{bmatrix} 0.93 \\ 0.07 \end{bmatrix}$$

Recall that the sum over all coordinates of each ŷ (and each y) must equal 1. Since there are only 2 classes, then $\hat{y}_0$ =1 - $\hat{y}_1$ (and y_0 = 1 - y_1).

Jacob Whitehill, WPI

Cross-entropy loss

- Note that the $f_{\log}$ (for logistic regression) is a special case of f_{CE} (for softmax regression) for c=2.

- To see how, consider just a simple example:

$$\begin{aligned} f_{\text{CE}} &= -\sum_{k=0}^{1} \mathbf{y}_k \log \hat{\mathbf{y}}_k \\ &= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - \mathbf{y}_0 \log \hat{\mathbf{y}}_0 \\ &= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - (1 - \mathbf{y}_1) \log(1 - \hat{\mathbf{y}}_1) \\ &= -\mathbf{y} \log \hat{\mathbf{y}} - (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}) \end{aligned}$$

For *c*=2 classes, we can define ŷ (and y) simply as probability that the example is class 1.

Cross-entropy loss

- Note that the $f_{\log}$ (for logistic regression) is a special case of f_{CE} (for softmax regression) for c=2.

- To see how, consider just a simple example:

$$
\begin{aligned}
f_{\mathrm{CE}} &= -\sum_{k=0}^{1} \mathbf{y}_k \log \hat{\mathbf{y}}_k \\
&= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - \mathbf{y}_0 \log \hat{\mathbf{y}}_0 \\
&= -\mathbf{y}_1 \log \hat{\mathbf{y}}_1 - (1 - \mathbf{y}_1) \log(1 - \hat{\mathbf{y}}_1) \\
&= -\mathbf{y} \log \hat{\mathbf{y}} - (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}) \\
&= f_{\log}
\end{aligned}
$$

Softmax activation function

- Softmax regression outputs a *vector* of probabilistic class labels $\hat{\mathbf{y}}$ containing c components.
  - We need c different vectors of weights $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(c)}$.
  - Each weight vector $\mathbf{w}^{(i)}$ measures how “compatible” $\mathbf{x}$ is with class i.

Softmax activation function

- With softmax regression, we first compute:

$$\mathbf{z}_1 = \mathbf{x}^\top \mathbf{w}^{(1)}$$
$$\mathbf{z}_2 = \mathbf{x}^\top \mathbf{w}^{(2)}$$

...

$$\mathbf{z}_c = \mathbf{x}^\top \mathbf{w}^{(c)}$$

I will refer to the z's as "pre-activation scores".

Softmax activation function

- With softmax regression, we first compute:

$$\mathbf{z}_1 = \mathbf{x}^\top \mathbf{w}^{(1)}$$
$$\mathbf{z}_2 = \mathbf{x}^\top \mathbf{w}^{(2)}$$
$$\dots$$
$$\mathbf{z}_c = \mathbf{x}^\top \mathbf{w}^{(c)}$$

- Since we want to output probabilities, we then **normalize** across all *c* classes so that:

1. Each output $\hat{\mathbf{y}}_k$ is non-negative.
2. The sum of $\hat{\mathbf{y}}_k$ over all *c* classes is 1.

Normalization of the $\hat{y}_k$

1. To enforce non-negativity, we can exponentiate each $\mathbf{z}_k$:

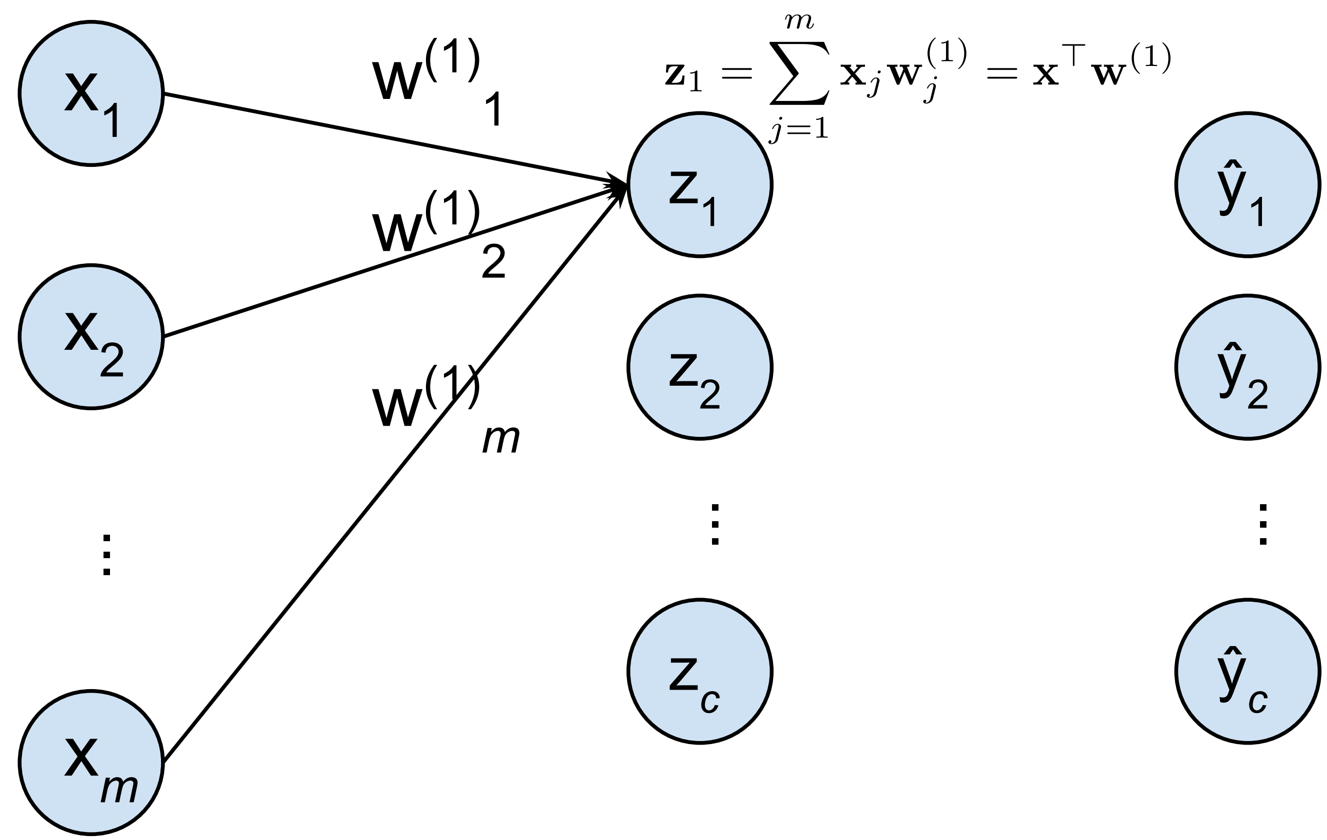
$$\hat{\mathbf{y}}_k = \quad \exp(\mathbf{z}_k)$$

Normalization of the $\hat{y}_k$

2. To enforce that the $\hat{\mathbf{y}}_k$ sum to 1, we can divide each entry by the sum:

$$\hat{\mathbf{y}}_k = \frac{\exp(\mathbf{z}_k)}{\sum_{k'=1}^{c} \exp(\mathbf{z}_{k'})}$$

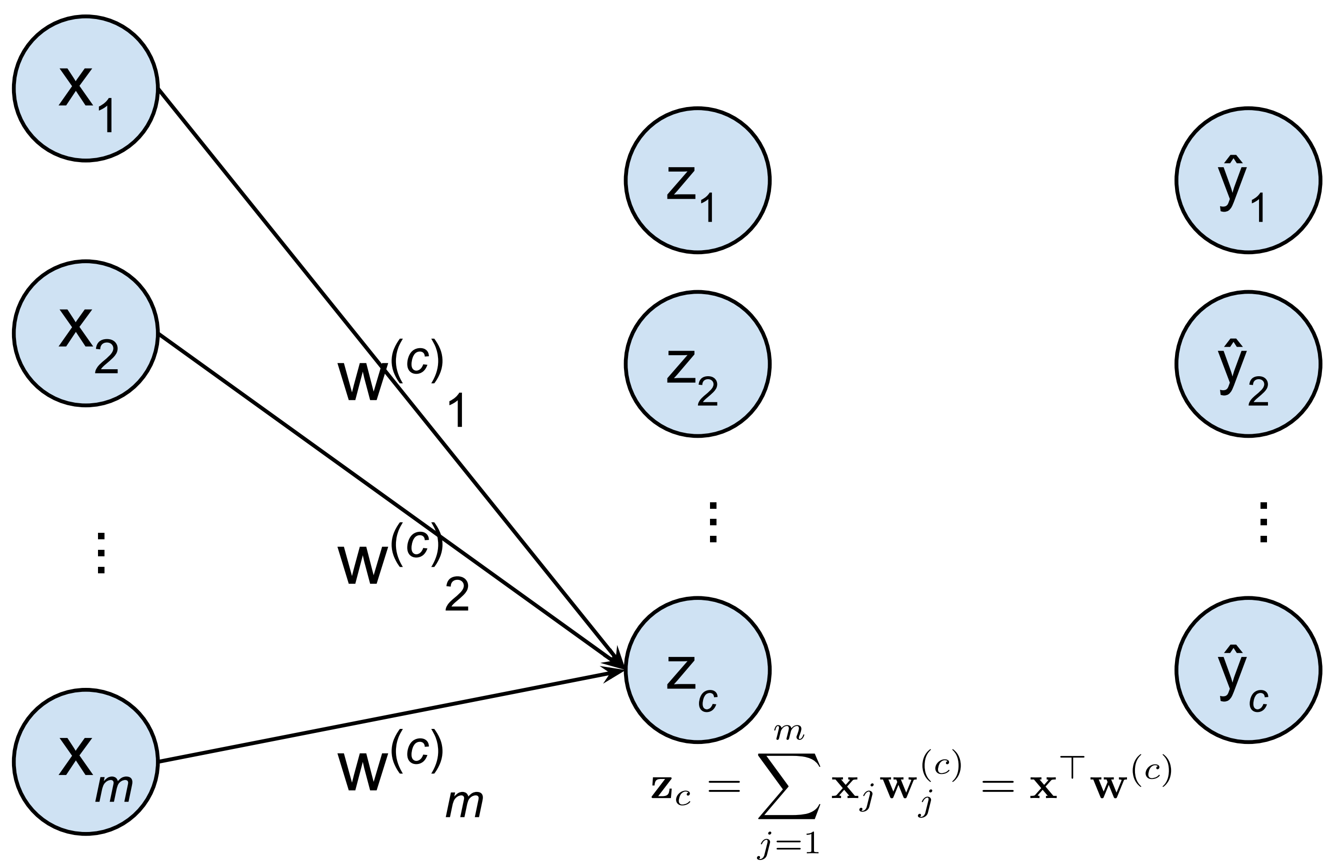
Softmax regression diagram



- With softmax regression, we first compute:

$$\mathbf{z}_1 = \mathbf{x}^\top \mathbf{w}^{(1)}$$

Softmax regression diagram



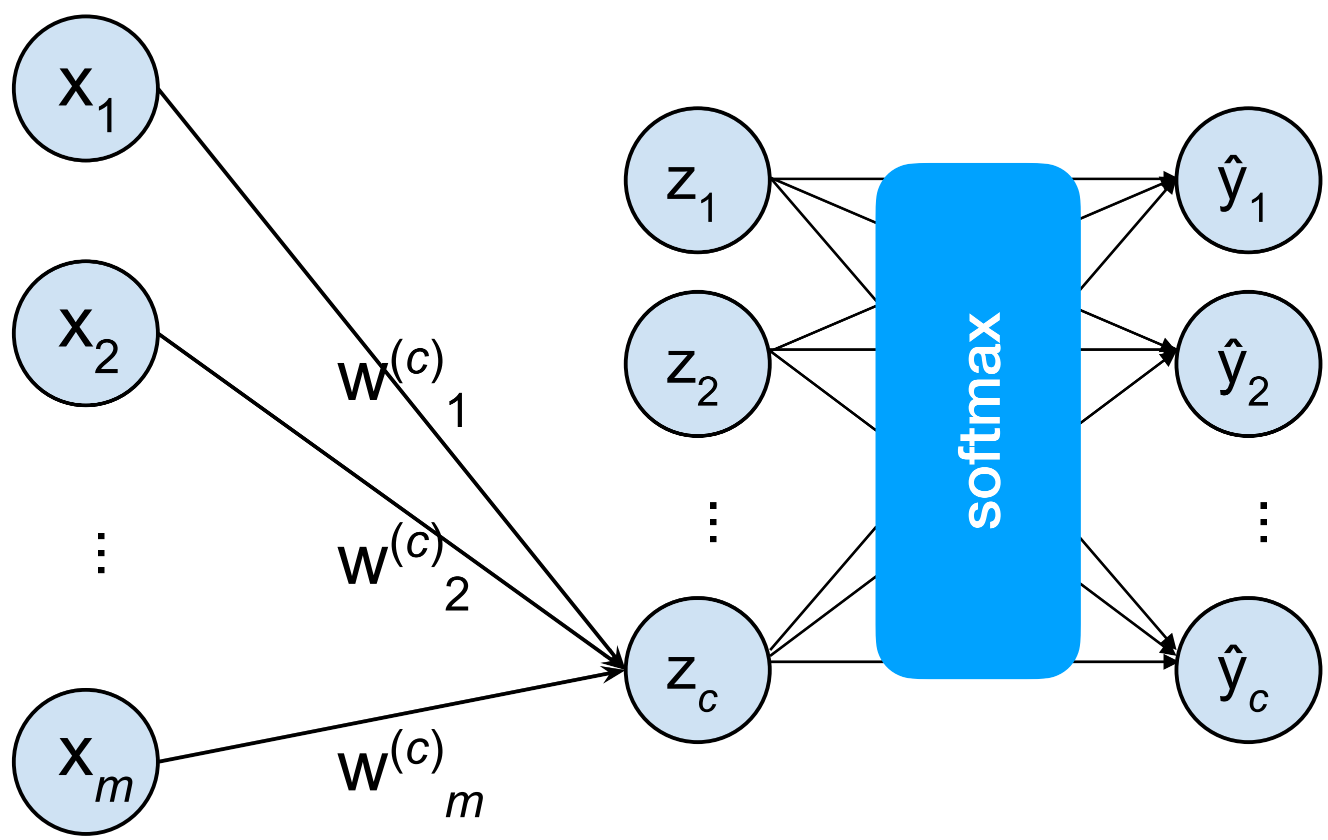
- With softmax regression, we first compute:

$$\mathbf{z}_1 = \mathbf{x}^\top \mathbf{w}^{(1)}$$

...

$$\mathbf{z}_c = \mathbf{x}^\top \mathbf{w}^{(c)}$$

Softmax regression diagram



$$\hat{\mathbf{y}}_k = \frac{\exp(\mathbf{z}_k)}{\sum_{k'=1}^{c} \exp(\mathbf{z}_{k'})}$$

- We then **normalize** across all *c* classes.

Illustration

- Let m=2, c=3.
- Let: $$\mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\mathbf{w}^{(1)} = \begin{bmatrix} -2.5 \\ -1 \end{bmatrix} \qquad \mathbf{w}^{(2)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \mathbf{w}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Which class will have highest estimated probability?

$$\mathbf{z} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

Illustration

- Let *m*=2, *c*=3.
- Let: $\mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

$$\mathbf{w}^{(1)} = \begin{bmatrix} -2.5 \\ -1 \end{bmatrix} \qquad \mathbf{w}^{(2)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \mathbf{w}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Which class will have highest estimated probability?

$$\mathbf{z} = \begin{bmatrix} 1.5 \\ 1 \\ -1 \end{bmatrix}$$

Illustration

- Let m=2, c=3.
- Let: $\mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

$$\mathbf{w}^{(1)} = \begin{bmatrix} -2.5 \\ -1 \end{bmatrix} \quad \mathbf{w}^{(2)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \mathbf{w}^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Which class will have highest estimated probability?

$$\mathbf{z} = \begin{bmatrix} 1.5 \\ 1 \\ -1 \end{bmatrix} \quad \hat{\mathbf{y}} = \begin{bmatrix} .592 \\ .359 \\ .049 \end{bmatrix}$$