Logistic regression

classification technique

① predict $P(y_i = 1 \mid \vec{x}_i) = p_i$

② apply threshold to p_i

e.g. $T = 0.5$

if $p_i \geq 0.5$
predict 1
else : predict 0

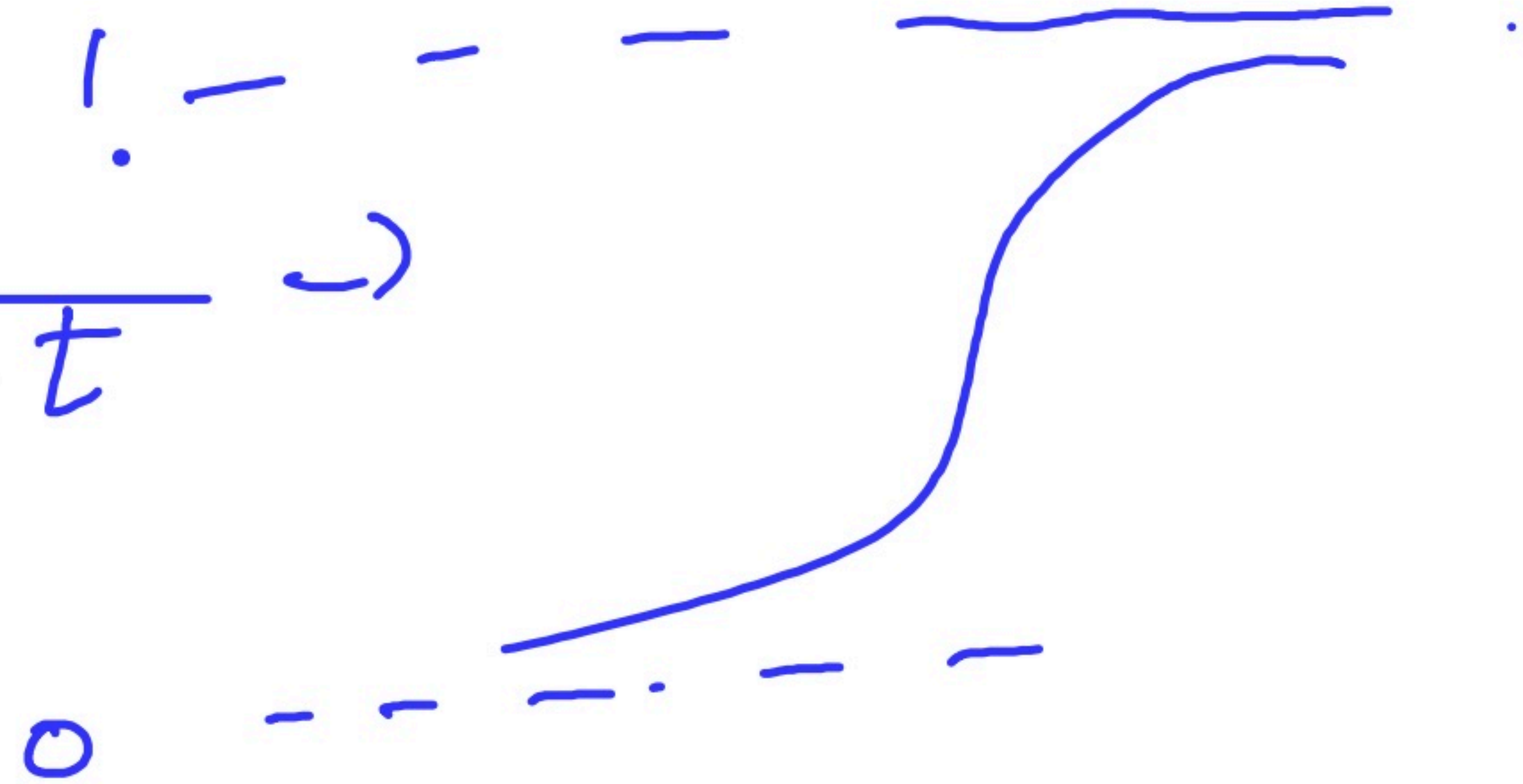
Logistic regression

- Logistic **regression** is a linear **classification** technique that builds upon linear regression.

- It models **the probability of belonging to class 1, given a feature vector**:

$$P(y_i = 1 | \vec{x}_i) = \sigma(w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_d x_i^{(d)}) = \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)$$

$$\sigma(t) = \frac{1}{1+e^{-t}}$$



Logistic regression

- Logistic **regression** is a linear **classification** technique that builds upon linear regression.

- It models **the probability of belonging to class 1, given a feature vector**:

$$P(y_i = 1|\vec{x}_i) = \sigma(w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_d x_i^{(d)}) = \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)$$

parameters, so this is a <u>parametric</u> model.

Logistic regression

- Logistic **regression** is a linear **classification** technique that builds upon linear regression.

- It models **the probability of belonging to class 1, given a feature vector**:

$$P(y_i = 1 | \vec{x}_i) = \sigma(w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_d x_i^{(d)}) = \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)$$

- Suppose we train a logistic regression model to predict the probability a patient has diabetes ($y = 1$) given their `'Glucose'` and `'BMI'`. If our optimal parameters end up being $\vec{w}^* = \begin{bmatrix} -7.85 & 0.04 & 0.08 \end{bmatrix}^T$, we then predict probabilities using:

$$P(y_i = 1 | \text{Glucose}_i, \text{BMI}_i) = \sigma(-7.85 + 0.04 \cdot \text{Glucose}_i + 0.08 \cdot \text{BMI}_i)$$

- To find the optimal parameters $\vec{w}^*$, we minimize mean **cross-entropy loss**:

There's no closed-form solution for $\vec{w}^*$, so we use some numerical method (or, rather, `sklearn` does).

$$\begin{aligned} R_{\text{ce}}(\vec{w}) &= -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log p_i + (1 - y_i) \log(1 - p_i) \right) \\ &= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log\left(\sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)\right) + (1 - y_i) \log\left(1 - \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)\right) \right] \end{aligned}$$

?

5.1

From binary to multiclass classification

- In binary classification, there are only two possible classes, typically either 0 or 1.

$$y_i \in \{0, 1\}$$

- In multiclass classification, there can be any finite number of classes, or **labels**. They need not be numbers, either.

$C = \{\text{Adelie, Chinstrap, Gentoo}\}$

$$y_i \in \{\text{Adelie, Chinstrap, Gentoo}\}$$

$|C| = 3$

- **Important**: Let C be the set of possible classes for our classification problem, and let $|C|$ be the number of classes total.

Multinomial logistic regression

- **Multinomial** logistic regression, also known as **softmax regression**, models the probability of belonging to **any class, given a feature vector** $\vec{x}_i$.

Think of it as a generalization of logistic regression.

$$p_{\text{Adelie}} = P(y_i = \text{Adelie} | \vec{x}_i) = \frac{e^{\vec{w}_{\text{Adelie}} \cdot \text{Aug}(\vec{x}_i)}}{e^{\vec{w}_{\text{Adelie}} \cdot \text{Aug}(\vec{x}_i)} + e^{\vec{w}_{\text{Chinstrap}} \cdot \text{Aug}(\vec{x}_i)} + e^{\vec{w}_{\text{Gentoo}} \cdot \text{Aug}(\vec{x}_i)}}$$

three parameter vectors:
one per species!

Multinomial logistic regression

- **Multinomial** logistic regression, also known as **softmax regression**, models the probability of belonging to **any class, given a feature vector** $\vec{x}_i$.

Think of it as a generalization of logistic regression.

$$p_{\text{Adelie}} = P(y_i = \text{Adelie} | \vec{x}_i) = \frac{e^{\vec{w}_{\text{Adelie}} \cdot \text{Aug}(\vec{x}_i)}}{e^{\vec{w}_{\text{Adelie}} \cdot \text{Aug}(\vec{x}_i)} + e^{\vec{w}_{\text{Chinstrap}} \cdot \text{Aug}(\vec{x}_i)} + e^{\vec{w}_{\text{Gentoo}} \cdot \text{Aug}(\vec{x}_i)}}$$

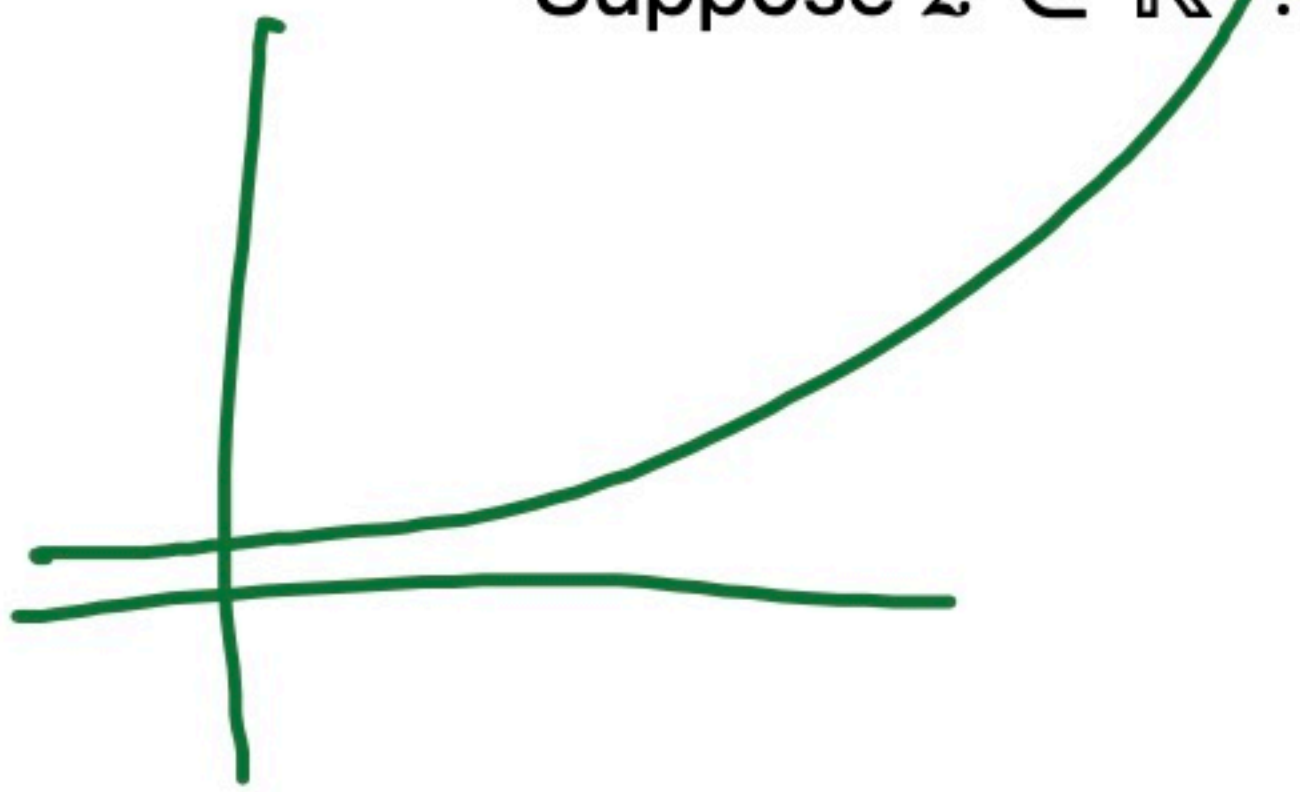
$$p_{\text{Chinstrap}} = P(y_i = \text{Chinstrap} | \vec{x}_i) = \frac{e^{\vec{w}_{\text{Chinstrap}} \cdot \text{Aug}(\vec{x}_i)}}{e^{\vec{w}_{\text{Adelie}} \cdot \text{Aug}(\vec{x}_i)} + e^{\vec{w}_{\text{Chinstrap}} \cdot \text{Aug}(\vec{x}_i)} + e^{\vec{w}_{\text{Gentoo}} \cdot \text{Aug}(\vec{x}_i)}}$$

denominator defined so that $p_{\text{Adelie}} + p_{\text{Chinstrap}} + p_{\text{Gentoo}} = 1.$

Aside: The softmax function

- The **softmax** function is a generalization of the logistic function to multiple dimensions.

  Suppose $\vec{z} \in \mathbb{R}^d$. Then, the softmax of $\vec{z}$ is defined element-wise as follows:



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}}$$

$$\sigma(\vec{z}) = \begin{bmatrix} \frac{e^{-5}}{e^{-5}+e^{2}+e^{4}} \\ \frac{e^{2}}{e^{-5}+e^{2}+e^{4}} \\ \frac{e^{4}}{e^{-5}+e^{2}+e^{4}} \end{bmatrix}$$

- For example, suppose $\vec{z} = \begin{bmatrix} -5 \\ 2 \\ 4 \end{bmatrix}$. Then:

biggest

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

"hard max"

$$\sigma(\vec{z}) = \begin{bmatrix} \sigma(\vec{z})_1 \\ \sigma(\vec{z})_2 \\ \sigma(\vec{z})_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{e^{-5}}{e^{-5}+e^{2}+e^{4}} \\ \frac{e^{2}}{e^{-5}+e^{2}+e^{4}} \\ \frac{e^{4}}{e^{-5}+e^{2}+e^{4}} \end{bmatrix}}_{\text{note the constant denominator!}} = \begin{bmatrix} 0.0001 \\ 0.1192 \\ 0.8807 \end{bmatrix}$$

output

Out[36]:

LogisticRegression

```
LogisticRegression(multi_class='multinomial')
```

- In total, the fit model has $3 \times 2 = 6$ coefficients and $3 \times 1 = 3$ intercepts.

In [37]:

```
1 model_log.coef_
```

Out[37]:

```
array([[-0.85,  0.  ],
       [ 0.84, -0.01],
       [ 0.02,  0.01]])
```

$$\vec{w}^*_{Adelie} = \begin{bmatrix} 36.4 \\ -0.85 \\ 0 \end{bmatrix}$$

In [39]:

```
1 model_log.intercept_
```

Out[39]:

```
array([ 36.4 , -10.96, -25.43])
```

$$\vec{w}^*_{Gentoo} = \begin{bmatrix} -25.43 \\ 0.02 \\ 0.01 \end{bmatrix}$$

In [40]:

```
1 model_log.classes_
```

Out[40]:

```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

Out[40]: array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)

- When calling `model_log.predict_proba`, we g

In [41]:
```
model_log.predict_proba(pd.DataFrame([{
    'bill_length_mm': 45,
    'body_mass_g': 4500
}]))
```

Out[41]: array([[0.14, 0.01, 0.85]])

P_{Adelie} $P_{chinstrap}$ P_{Gentoo}

Out[40]: array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)

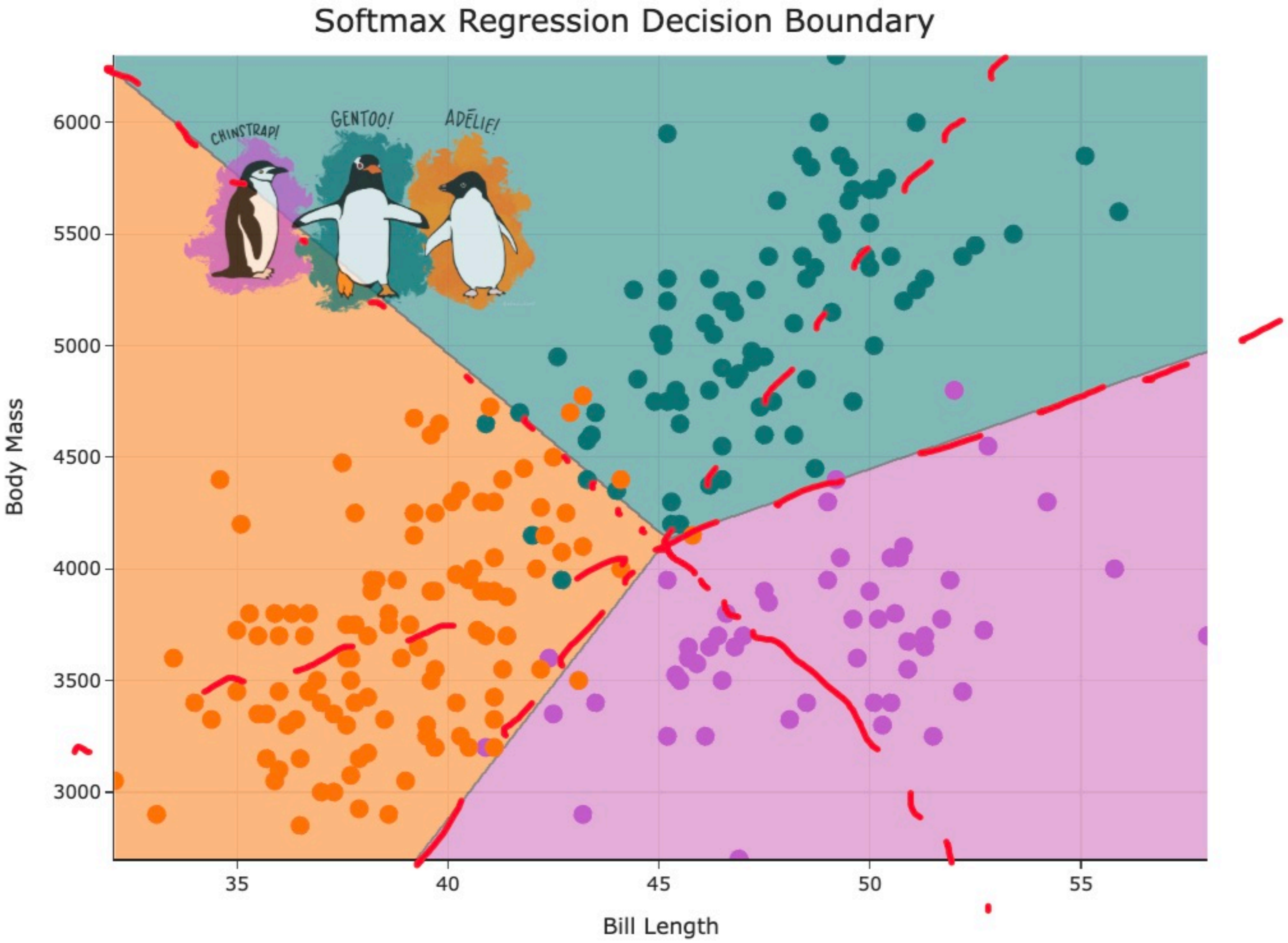
- When calling `model_log.predict_proba`, we g

In [44]:
```
model_log.predict(pd.DataFrame([{
    'bill_length_mm': 45,
    'body_mass_g': 4500
}]))
```

Out[44]: array(['Gentoo'], dtype=object)

What does this model *look* like?

In [47]:
```
util.penguin_decision_boundary(model_log, X_train, y_train, title="Softmax Regression Decision Boundary")
```

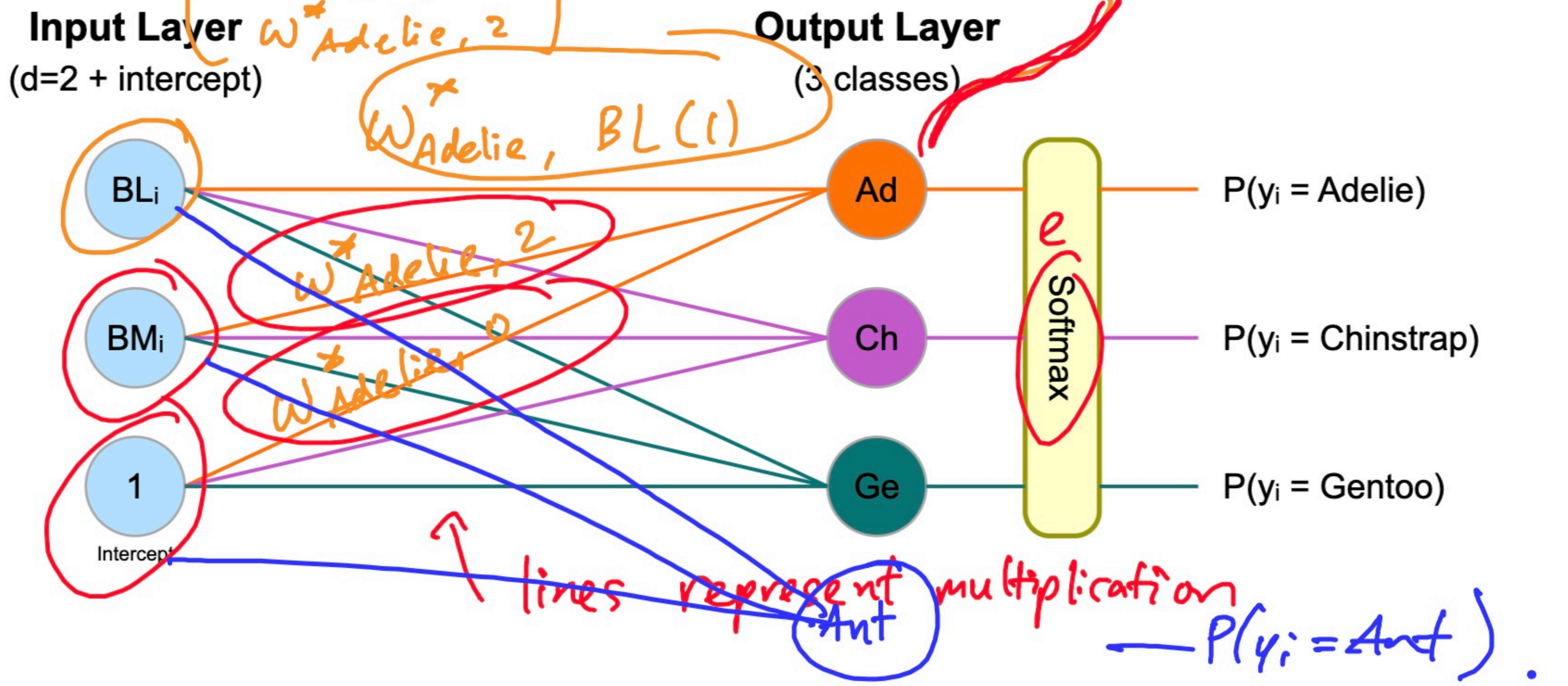


linear

decision boundary.

$$\vec{w}^{*}_{Adelie} = \begin{bmatrix} w^{*}_{Adelie,0} \\ w^{*}_{Adelie,1} \\ w^{*}_{Adelie,2} \end{bmatrix}$$

$$\vec{w}^{T}_{Adelie} \cdot Aug(\vec{x}_i)$$



Out[3]:

	pixel1	pixel2	pixel3	pixel4	...	pixel781	pixel782	pixel783	pixel784
0	0	0	0	0	...	0	0	0	0
1	0	0	0	0	...	0	0	0	0
2	0	0	0	0	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...
59997	0	0	0	0	...	0	0	0	0
59998	0	0	0	0	...	0	0	0	0
59999	0	0	0	0	...	0	0	0	0

$28^2.$

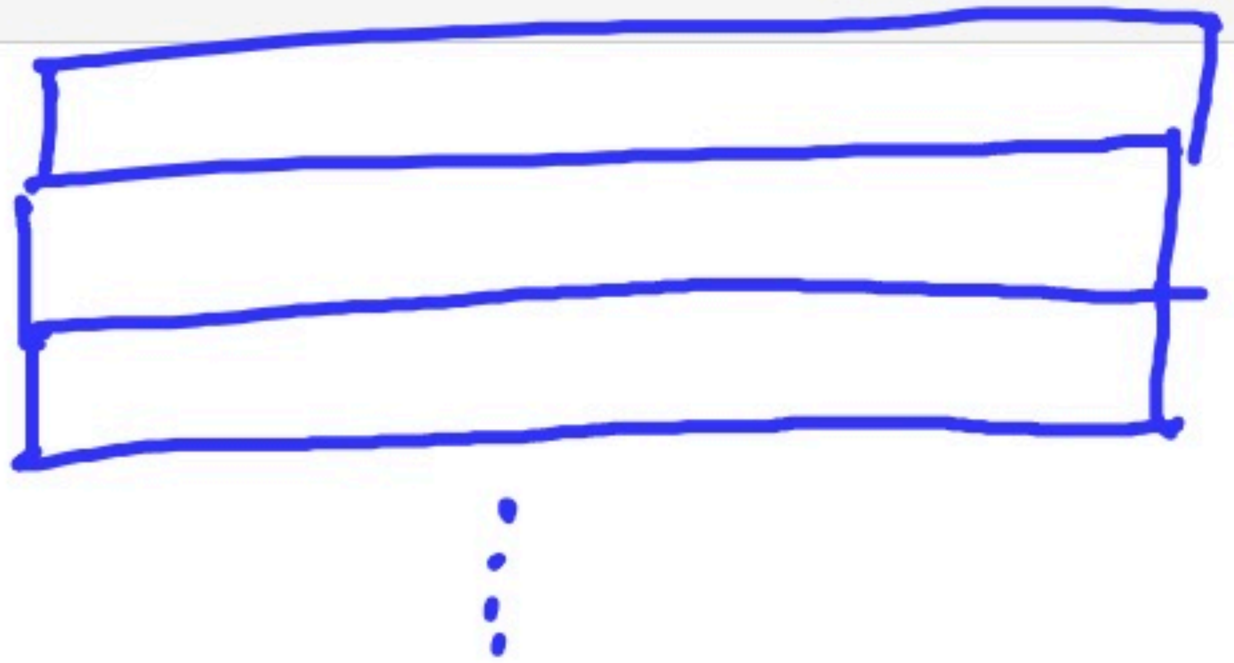
60000 rows × 784 columns

In [4]:

```
y_train
```

In [5]:
```
X_train.iloc[98]
```

Out[5]:
```
pixel1      0
pixel2      0
pixel3      0
           ..
pixel782    0
pixel783    0
pixel784    0
Name: 98, Length: 784, dtype: int64
```



- The first 28 pixels are the first **row** of the image, the second 28 pixels are the second row of the image, and so on. To view the image, we can **reshape** the vector into a 2D grid.

In [6]:
```
X_train.iloc[98].to_numpy().reshape((28, 28))
```

Out[6]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

localhost

Model #2: Multinomial logistic regression 📈

- Multinomial logistic regression, or softmax regression, predicts the probability that an image $\vec{x}_i \in \mathbb{R}^{784}$ belongs to each class.

$$P(\text{image } \vec{x}_i \text{ is of digit } j) = P(y_i = j | \vec{x}_i) = \frac{e^{\vec{w}_j \cdot \text{Aug}(\vec{x}_i)}}{\sum_{k=0}^{9} e^{\vec{w}_k \cdot \text{Aug}(\vec{x}_i)}}$$

Here, j could be 0, 1, 2, ..., 9.

- To train a multinomial logistic regression model for this task, we'll ultimately need to find 10 optimal parameter vectors, $\vec{w}_0^*, \vec{w}_1^*, ..., \vec{w}_9^*$.

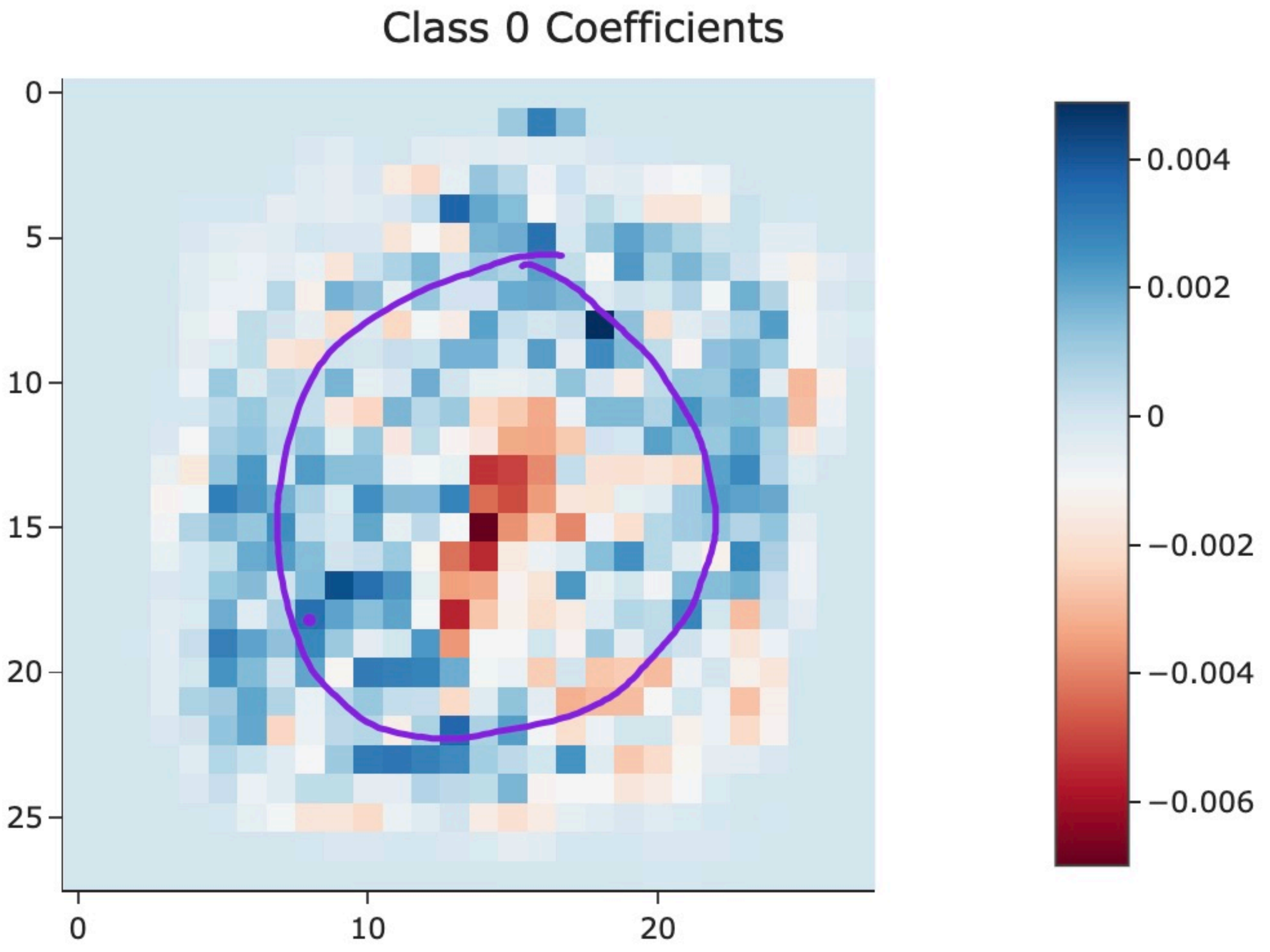
total parameters : 7850

$10 \times (784+1)$.

?

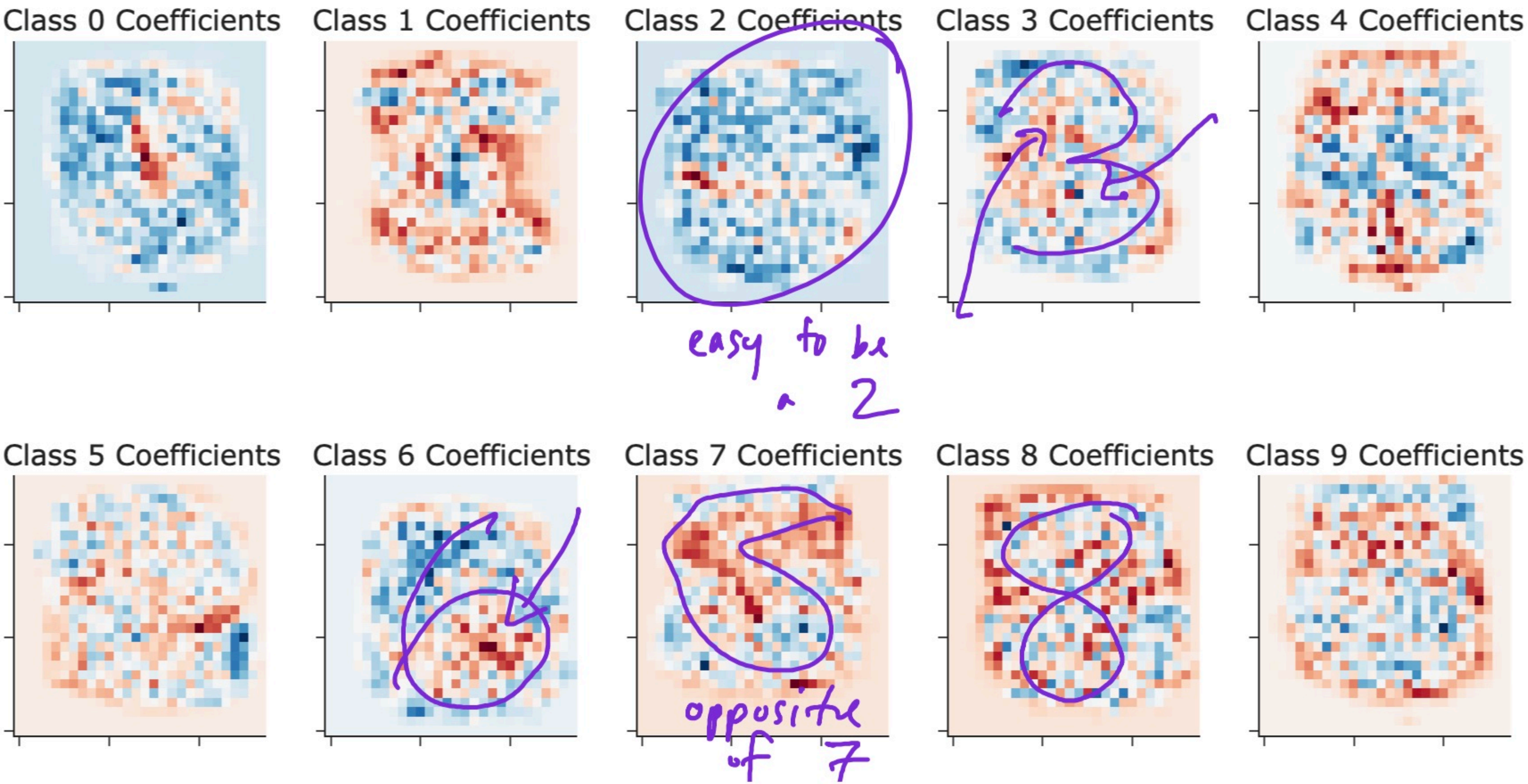
15 . 1

```
In [65]: 1 px.imshow(model_log.coef_[0].reshape((28, 28)), color_continuous_scale=
```



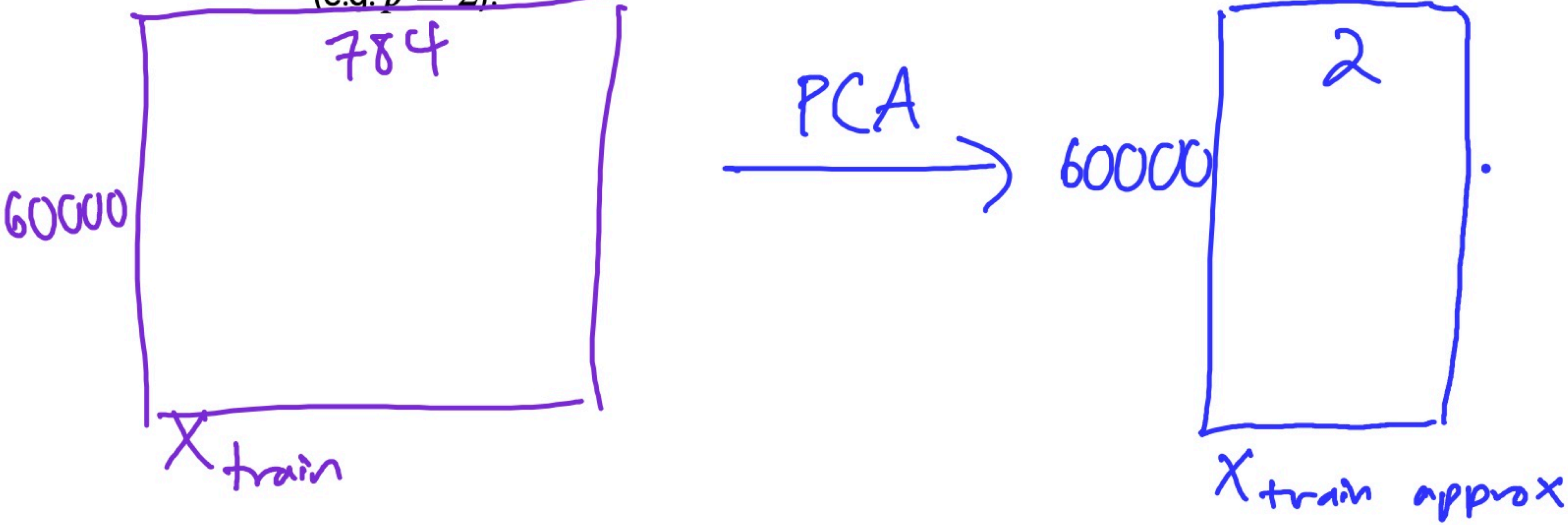
$$\vec{W}^{T}_{\text{class } 0} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

localhost


Class 0 Coefficients
Class 1 Coefficients
Class 2 Coefficients
Class 3 Coefficients
Class 4 Coefficients
easy to be a 2
Class 5 Coefficients
Class 6 Coefficients
Class 7 Coefficients
Class 8 Coefficients
Class 9 Coefficients
opposite of 7

- Principal component analysis (PCA) is an **unsupervised learning** technique used for **dimensionality reduction**.

- It'll allow us to take:
  - `X_train`, which has 60,000 rows and 784 columns, and transform it into
  - `X_train_approx`, which has 60,000 rows and p columns, where p is as small as we want (e.g. $p = 2$).



- It'll allow us to take:
  - `X_train`, which has 60,000 rows and 784 columns, and transform it into
  - `X_train_approx`, which has 60,000 rows and p columns, where p is as small as we want (e.g. $p = 2$).

- It creates p **new features**, each of which is a linear combination of all existing 784 features.

$$\text{new feature 1} = 0.05 \cdot \text{pixel 1} + 0.93 \cdot \text{pixel 2} + \ldots - 0.35 \cdot \text{pixel 784}$$

$$\text{new feature 2} = -0.06 \cdot \text{pixel 1} + 0.5 \cdot \text{pixel 2} + \ldots + 0.04 \cdot \text{pixel 784}$$

$$\ldots$$

These new features are chosen to capture as much variability (information) in the original data as possible.