CS 453X: Class 4

Jacob Whitehill

Linear regression in 2-d

- Let's examine again how linear regression works with just 2 dimensions and just a single training example:

$$f_{\mathrm{MSE}}(w_1, w_2) \quad = \quad \frac{1}{2}(x_1 w_1 + x_2 w_2 - y)^2$$

Linear regression in 2-d

- Let's examine again how linear regression works with just 2 dimensions and just a single training example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(w_1, w_2) &= \frac{1}{2}(x_1 w_1 + x_2 w_2 - y)^2 \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_1}(w_1, w_2) &= x_1(x_1 w_1 + x_2 w_2 - y)
\end{aligned}
$$

Linear regression in 2-d

- Let's examine again how linear regression works with just 2 dimensions and just a single training example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(w_1, w_2) &= \frac{1}{2}(x_1 w_1 + x_2 w_2 - y)^2 \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_1}(w_1, w_2) &= x_1(x_1 w_1 + x_2 w_2 - y) \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_2}(w_1, w_2) &= x_2(x_1 w_1 + x_2 w_2 - y)
\end{aligned}
$$

Linear regression in 2-d

- Let's examine again how linear regression works with just 2 dimensions and just a single training example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(w_1, w_2) &= \frac{1}{2}(x_1 w_1 + x_2 w_2 - y)^2 \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_1}(w_1, w_2) &= x_1(x_1 w_1 + x_2 w_2 - y) \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_2}(w_1, w_2) &= x_2(x_1 w_1 + x_2 w_2 - y) \\
\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) &= \begin{bmatrix} \frac{\partial f_{\mathrm{MSE}}}{\partial w_1} \\ \\ \frac{\partial f_{\mathrm{MSE}}}{\partial w_2} \end{bmatrix}
\end{aligned}
$$

Linear regression in 2-d

- Let's examine again how linear regression works with just 2 dimensions and just a single training example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(w_1, w_2) &= \frac{1}{2}(x_1 w_1 + x_2 w_2 - y)^2 \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_1}(w_1, w_2) &= x_1(x_1 w_1 + x_2 w_2 - y) \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_2}(w_1, w_2) &= x_2(x_1 w_1 + x_2 w_2 - y) \\
\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) &= \begin{bmatrix} \frac{\partial f_{\mathrm{MSE}}}{\partial w_1} \\ \\ \frac{\partial f_{\mathrm{MSE}}}{\partial w_2} \end{bmatrix} \\
&= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} (x_1 w_1 + x_2 w_2 - y)
\end{aligned}
$$

Linear regression in 2-d

- Let's examine again how linear regression works with just 2 dimensions and just a single training example:

$$
\begin{aligned}
f_{\mathrm{MSE}}(w_1, w_2) &= \frac{1}{2}(x_1 w_1 + x_2 w_2 - y)^2 \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_1}(w_1, w_2) &= x_1(x_1 w_1 + x_2 w_2 - y) \\
\frac{\partial f_{\mathrm{MSE}}}{\partial w_2}(w_1, w_2) &= x_2(x_1 w_1 + x_2 w_2 - y) \\
\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{w}) &= \begin{bmatrix} \frac{\partial f_{\mathrm{MSE}}}{\partial w_1} \\ \\ \frac{\partial f_{\mathrm{MSE}}}{\partial w_2} \end{bmatrix} \\
&= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} (x_1 w_1 + x_2 w_2 - y) \\
&= \mathbf{x}(\mathbf{x}^\top \mathbf{w} - y)
\end{aligned}
$$

Linear regression: matrix notation

- Suppose we have *n* images, each with just 2 pixels.

$$\hat{\mathbf{y}} = \mathbf{X}^\top \mathbf{w}$$

Linear regression: matrix notation

- Suppose we have *n* images, each with just 2 pixels.

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}^\top \mathbf{w} \\ &= \begin{bmatrix} \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_1^{(n)} \\ \mathbf{x}_2^{(1)} & \dots & \mathbf{x}_2^{(n)} \end{bmatrix}^\top \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}\end{aligned}$$

This is the index of the *image*.

Linear regression: matrix notation

- Suppose we have *n* images, each with just 2 pixels.

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}^\top \mathbf{w} \\ &= \begin{bmatrix} \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_1^{(n)} \\ \mathbf{x}_2^{(1)} & \dots & \mathbf{x}_2^{(n)} \end{bmatrix}^\top \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}\end{aligned}$$

This is the index of the *pixel*.

Linear regression: matrix notation

- Suppose we have *n* images, each with just 2 pixels.

$$
\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}^\top \mathbf{w} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_1^{(n)} \\ \mathbf{x}_2^{(1)} & \dots & \mathbf{x}_2^{(n)} \end{bmatrix}^\top \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(1)} \\ \vdots & \vdots \\ \mathbf{x}_1^{(n)} & \mathbf{x}_2^{(n)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}
\end{aligned}
$$

Linear regression: matrix notation

- Suppose we have *n* images, each with just 2 pixels.

$$
\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}^\top \mathbf{w} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_1^{(n)} \\ \mathbf{x}_2^{(1)} & \dots & \mathbf{x}_2^{(n)} \end{bmatrix}^\top \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(1)} \\ \vdots & \vdots \\ \mathbf{x}_1^{(n)} & \mathbf{x}_2^{(n)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} w_1 + \mathbf{x}_2^{(1)} w_2 \\ \vdots \\ \mathbf{x}_1^{(n)} w_1 + \mathbf{x}_2^{(n)} w_2 \end{bmatrix}
\end{aligned}
$$

Linear regression: matrix notation

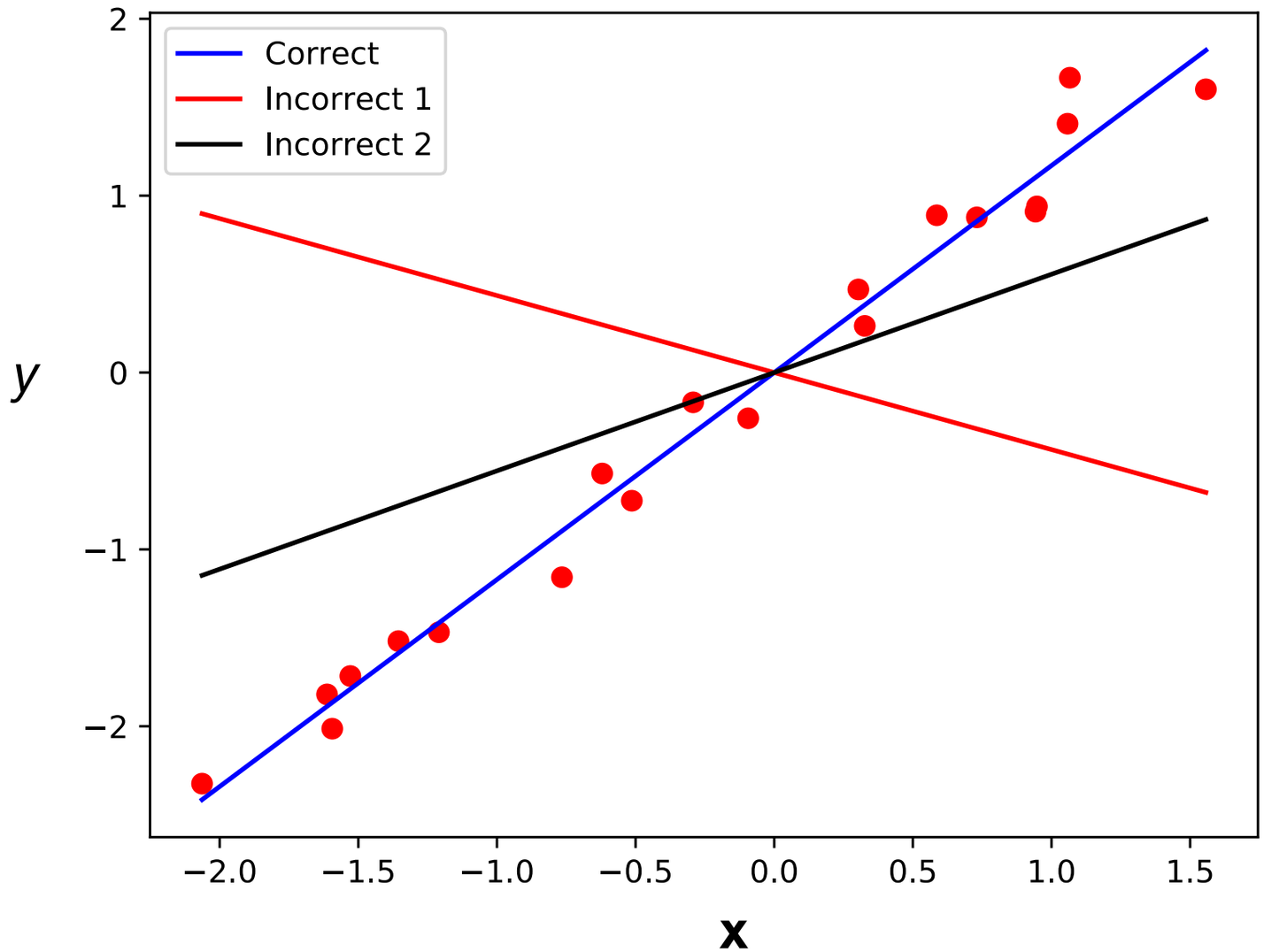
- Suppose we have *n* images, each with just 2 pixels.

$$
\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}^\top \mathbf{w} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_1^{(n)} \\ \mathbf{x}_2^{(1)} & \dots & \mathbf{x}_2^{(n)} \end{bmatrix}^\top \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(1)} \\ \vdots & \vdots \\ \mathbf{x}_1^{(n)} & \mathbf{x}_2^{(n)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}_1^{(1)} w_1 + \mathbf{x}_2^{(1)} w_2 \\ \vdots \\ \mathbf{x}_1^{(n)} w_1 + \mathbf{x}_2^{(n)} w_2 \end{bmatrix} \\
&= \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}
\end{aligned}
$$

With just a single matrix-vector multiplication, we have computed our predictions for *all* of the *n* images simultaneously.

1-d example

- Linear regression finds the weight vector **w** that minimizes the f_{MSE}. Here's an example where each **x** is just 1-d…

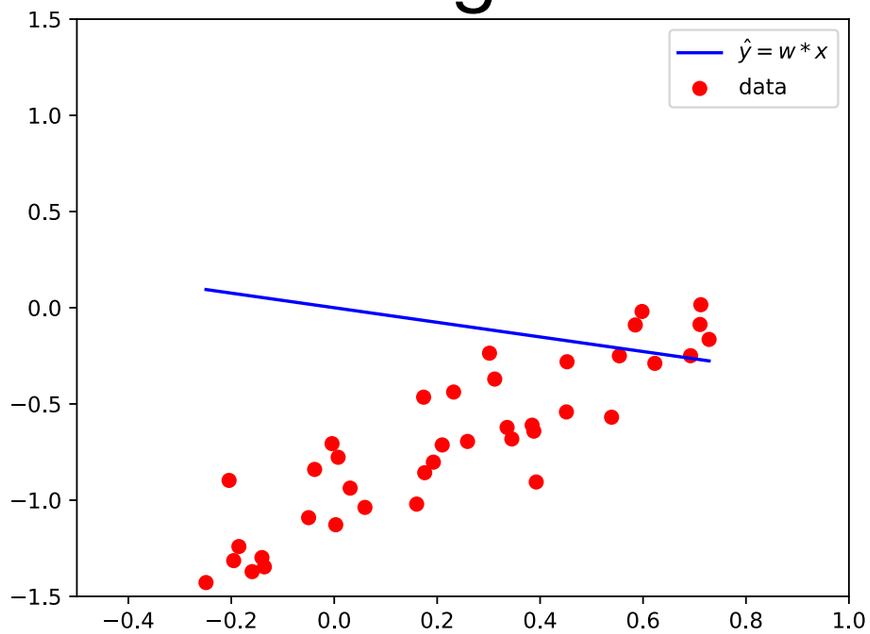


The best **w** is the one such that $f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}})$ is as small as possible, where each $\hat{y} = \mathbf{x}^{\mathsf{T}}\mathbf{w}$.

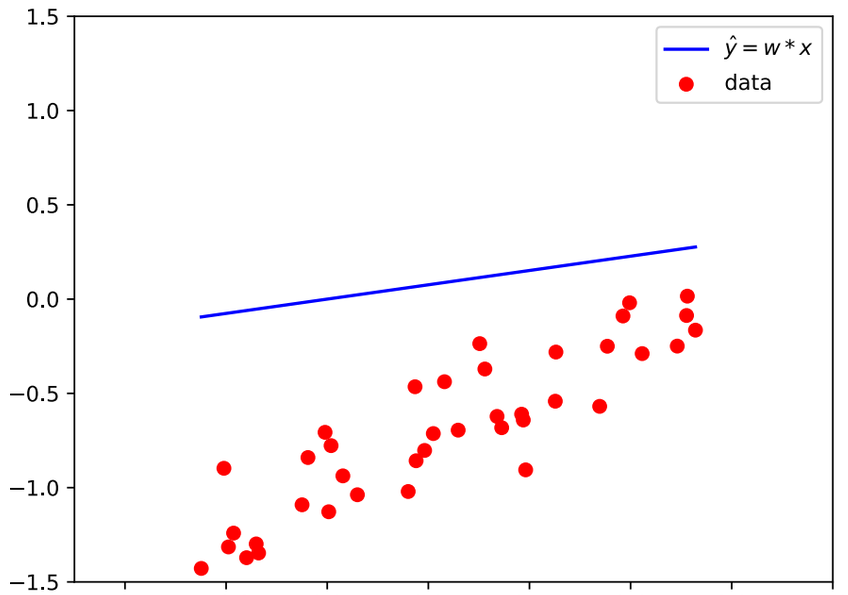
Exercise

- Which of the following regression lines would be predicted using the model described above?

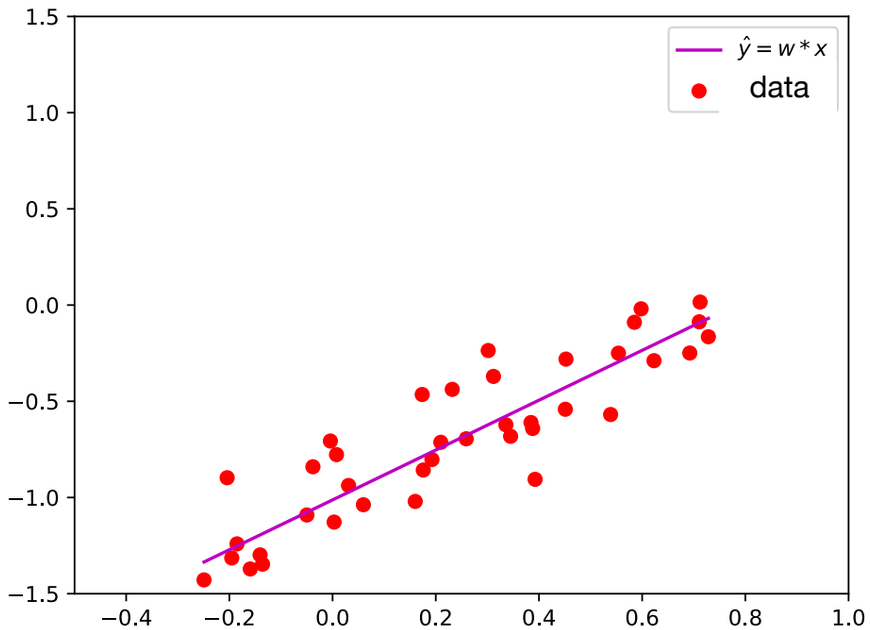
1.



2.



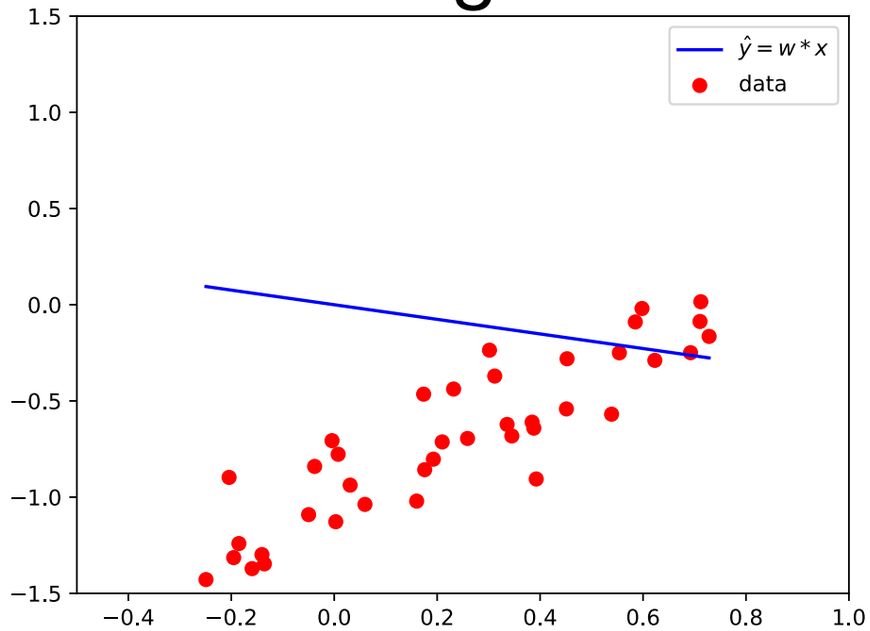
3.



Exercise

- Which of the following regression lines would be predicted using the model described above?

1.



Notice that the model enforces that (*x*,*y*)=(0,0) lie in the graph. Because of this constraint, the model learns the wrong slope to minimize the MSE.

Bias term

- In order to account for target values with non-zero mean, we could add a bias term to our model:

$$\hat{y} = \mathbf{x}^\top \mathbf{w} + b$$

- We could then compute the gradient w.r.t. both **w** and *b* and solve.

$$\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}, b) = \nabla_{\mathbf{w}} \left[\frac{1}{2n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - y^{(i)} \right)^2 \right]$$

$$\nabla_{b} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}, b) = \nabla_{b} \left[\frac{1}{2n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)\top} \mathbf{w} + b - y^{(i)} \right)^2 \right]$$

Bias term

- Alternatively, we can implicitly include a bias term by augmenting each input vector **x** with a 1 at the end:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

- Correspondingly, our weight vector **w** will have an extra component (bias term) at the end.

$$\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

Bias term

- To see why, notice that:

$$
\begin{aligned}
\hat{y} &= \tilde{\mathbf{x}}^\top \tilde{\mathbf{w}} \\
&= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\
&= \mathbf{x}^\top \mathbf{w} + b
\end{aligned}
$$

Bias term

- We can find the optimal **w** and *b* based on all the training data using matrix notation.
- First define an augmented design matrix:

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}^{(1)} & \dots & \mathbf{x}^{(n)} \\ 1 & \dots & 1 \end{bmatrix}$$

- Then compute:

$$\tilde{\mathbf{w}} = \left(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top\right)^{-1}\tilde{\mathbf{X}}\mathbf{y}$$

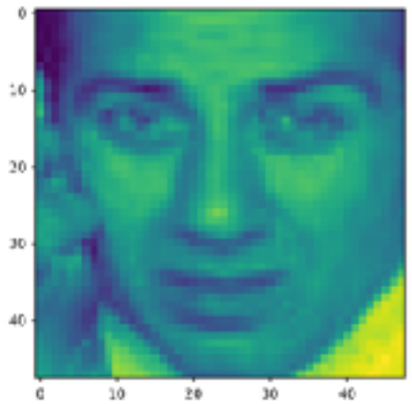
Example 1: age estimation

- Regress the age from 48x48 face images.

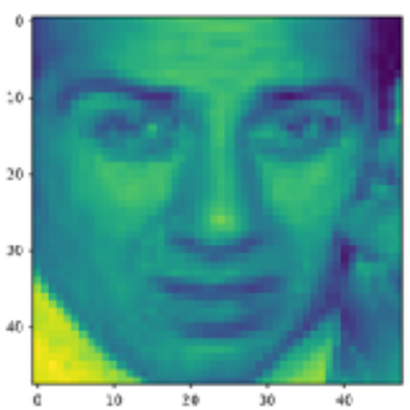
- Show demo…

Data Augmentation

- How can we easily increase the number of training images?

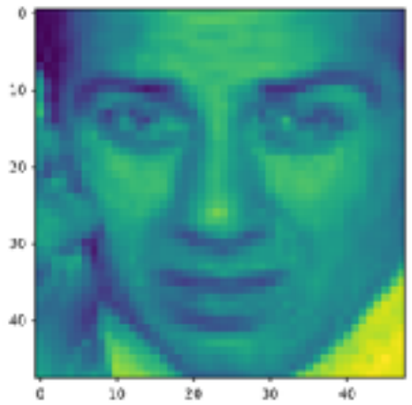


- Flip them left-right:



Data Augmentation

- How can we easily increase the number of training images?



Data Augmentation

- In **`numpy`**:

  - Let **`faces`** be a (*n* x 48 x 48) matrix containing *n* images.

  - Then **`facesFlipped = faces[:, :, ::-1]`** contains the left-right flipped images.

All *n* images.

Data Augmentation

- In **`numpy`**:

  - Let **`faces`** be a (*n* x 48 x 48) matrix containing *n* images.

  - Then **`facesFlipped = faces[:, :, ::-1]`** contains the left-right flipped images.

All 48 rows.

Data Augmentation

- In **`numpy`**:

  - Let **`faces`** be a (*n* x 48 x 48) matrix containing *n* images.

  - Then **`facesFlipped = faces[:, :, ::-1]`** contains the left-right flipped images.

All 48 columns but in reverse order.

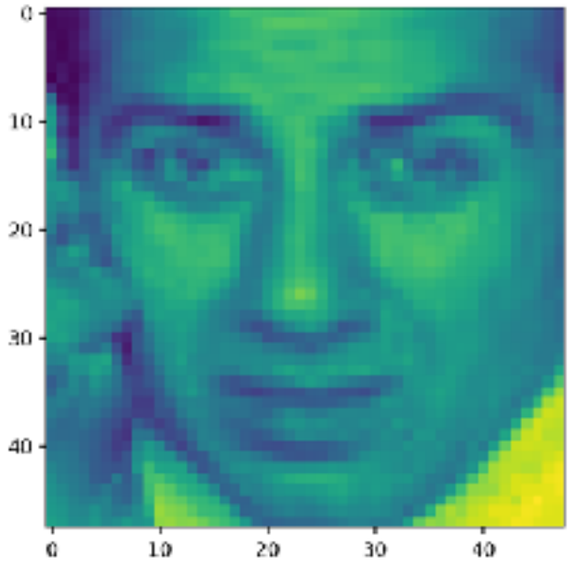
Data Augmentation

- Avoid leakage of facial identity information:
    - Make sure that no “flipped” image in the testing set has an “unflipped” image in the training set (and vice-versa).
- **Data leakage**: information in the training set which divulges information about the test set and which can bias the accuracy estimates.

Learned weights

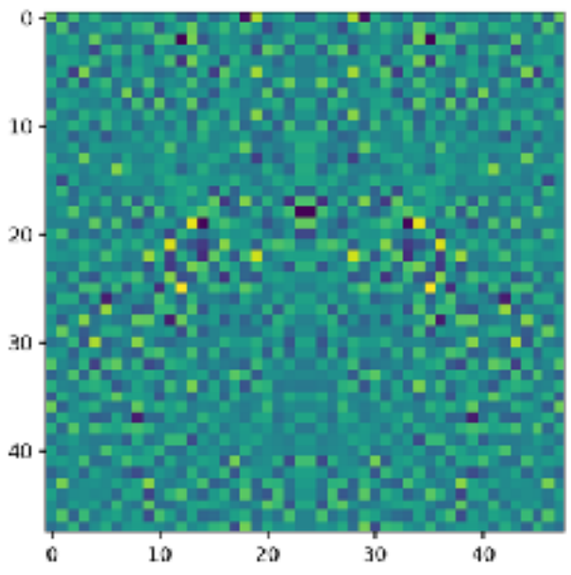
- Inspecting what the machine learned can be useful for debugging (and kinda fun to look at).

- For age estimation:



x

Furrows around nose?
Wrinkles in forehead?



w

Higher temperatures associated with larger age values.