Introduction to Machine Learning and Artificial Intelligence

Presented by: Moran Davoodi

What is learning?

Learning is a process that leads to *change*, which occurs as a result of running through *data* and increases the potential of improved *performance*

How do we learn?

At the end of the training period, we will apply what we have learned and hope for the best ☺

Get a task (input-X) → Try to solve (apply $\hat{f}$) → Make a decision (output- $\hat{y}$) → Get the correct answer (y) → Evaluate the error ($L(\hat{y}, y)$)

Fix $\hat{f}$ accordingly

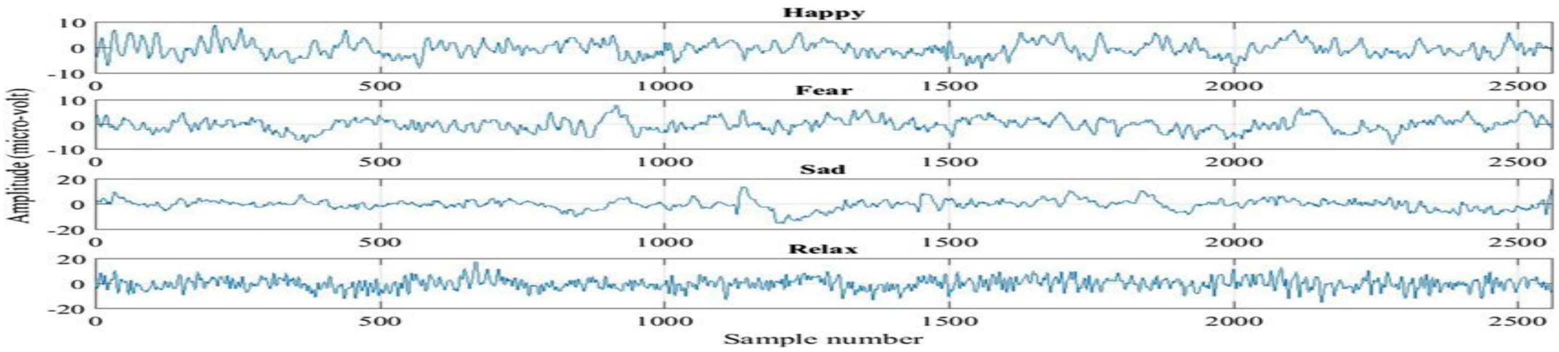
$$f(X) = y$$
$$f = ?$$
$$\hat{y} = \hat{f}(X)$$
$$\hat{y} \approx y \rightarrow L(\hat{y}, y) \approx \min(L) \rightarrow \boxed{\hat{f} \approx f}$$

Actually we are looking for $P(y|X)$

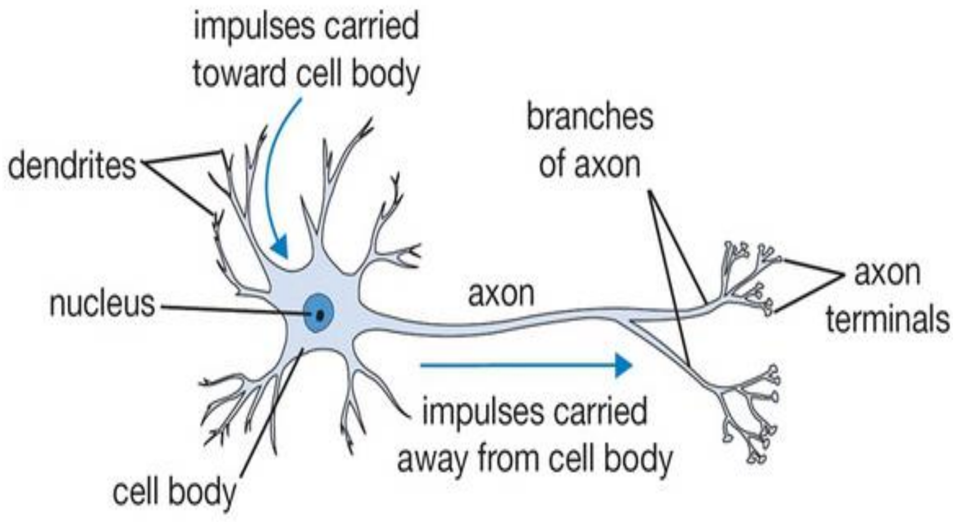
$$P\left(y = 1[shekel] | X = \begin{pmatrix} 4.5[g] \\ 19[mm] \end{pmatrix}\right) = 0.9$$
$$P\left(y = 5[shekel] | X = \begin{pmatrix} 4.5[g] \\ 19[mm] \end{pmatrix}\right) = 0.1$$

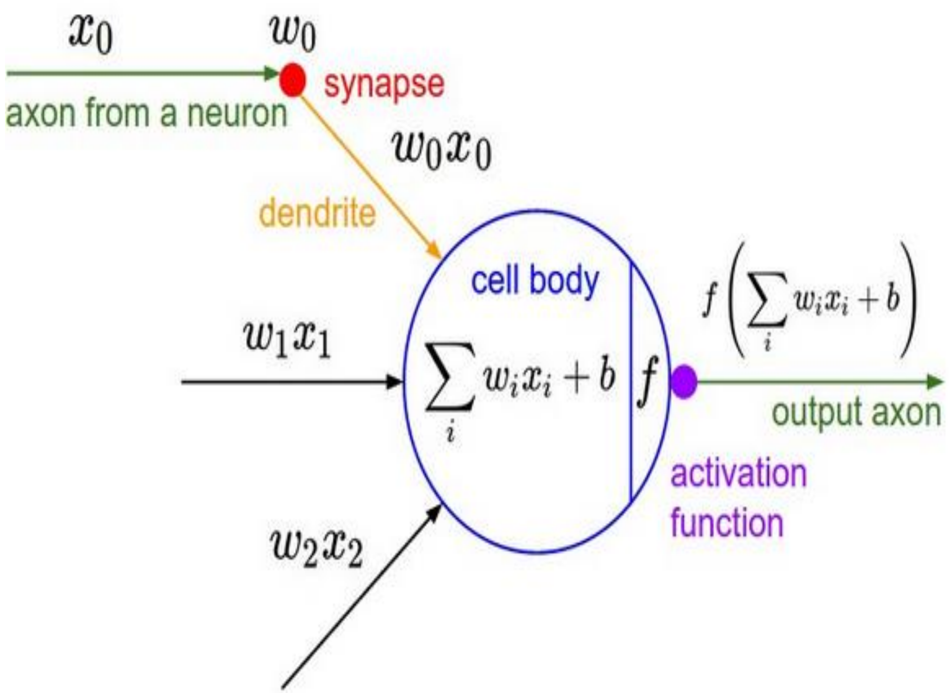
Estimating current feeling with EEG

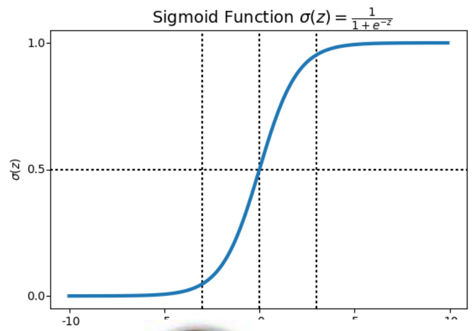
- Get an EEG signal as an input.
- Estimate the probability of every feeling (apply $\hat{f}$).
- Show the computer the correct answer (feeling).
- Fix the probability function through the weights using our loss function.
- Get a new signal with its' label (adequate feeling) and repeat.



How do we decide?

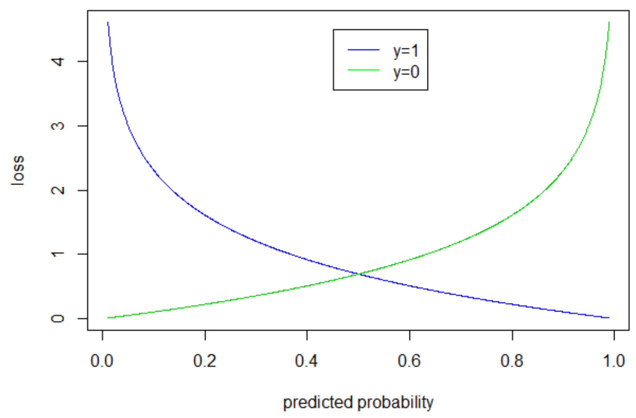




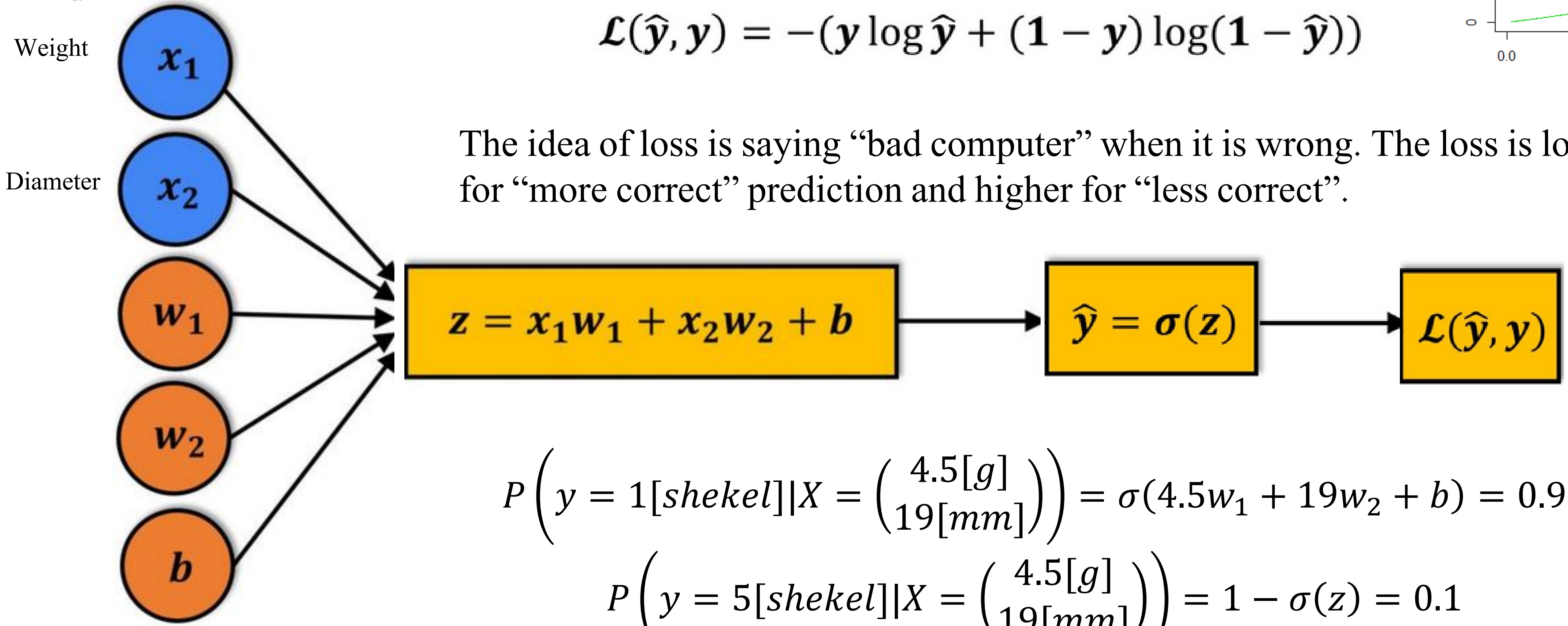


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$



The idea of loss is saying “bad computer” when it is wrong. The loss is lower for “more correct” prediction and higher for “less correct”.



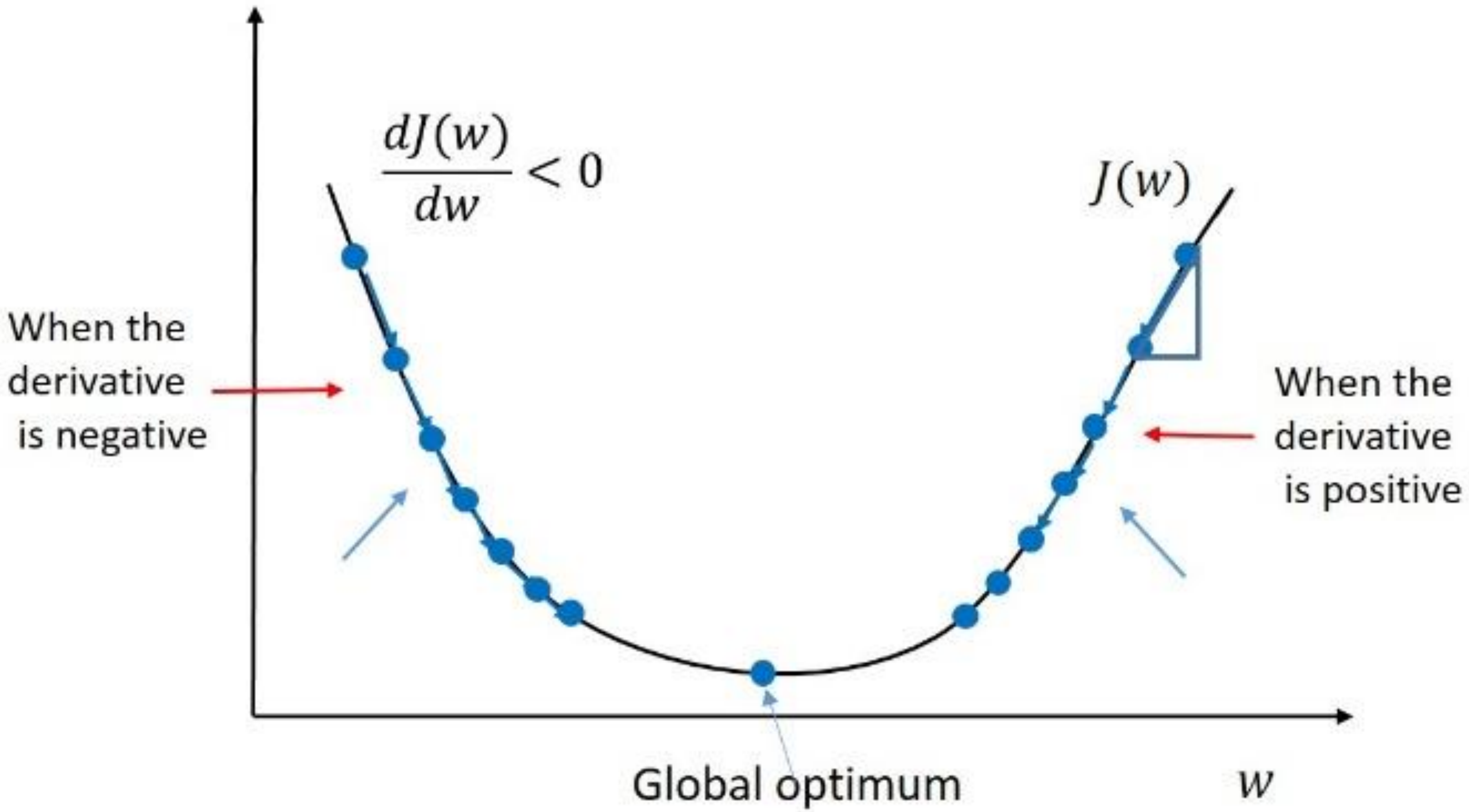
$$P\left(y = 1[shekel] | X = \begin{pmatrix} 4.5[g] \\ 19[mm] \end{pmatrix}\right) = \sigma(4.5w_1 + 19w_2 + b) = 0.9$$

$$P\left(y = 5[shekel] | X = \begin{pmatrix} 4.5[g] \\ 19[mm] \end{pmatrix}\right) = 1 - \sigma(z) = 0.1$$

$$P(y = 1|X) > P(y = 5|X) \rightarrow \hat{y} = 1$$

Our aim is to LEARN correctly (w_1, w_2, b) i.e. the set of parameters that minimizes the loss

Calculus as learning factor



Gradient descent

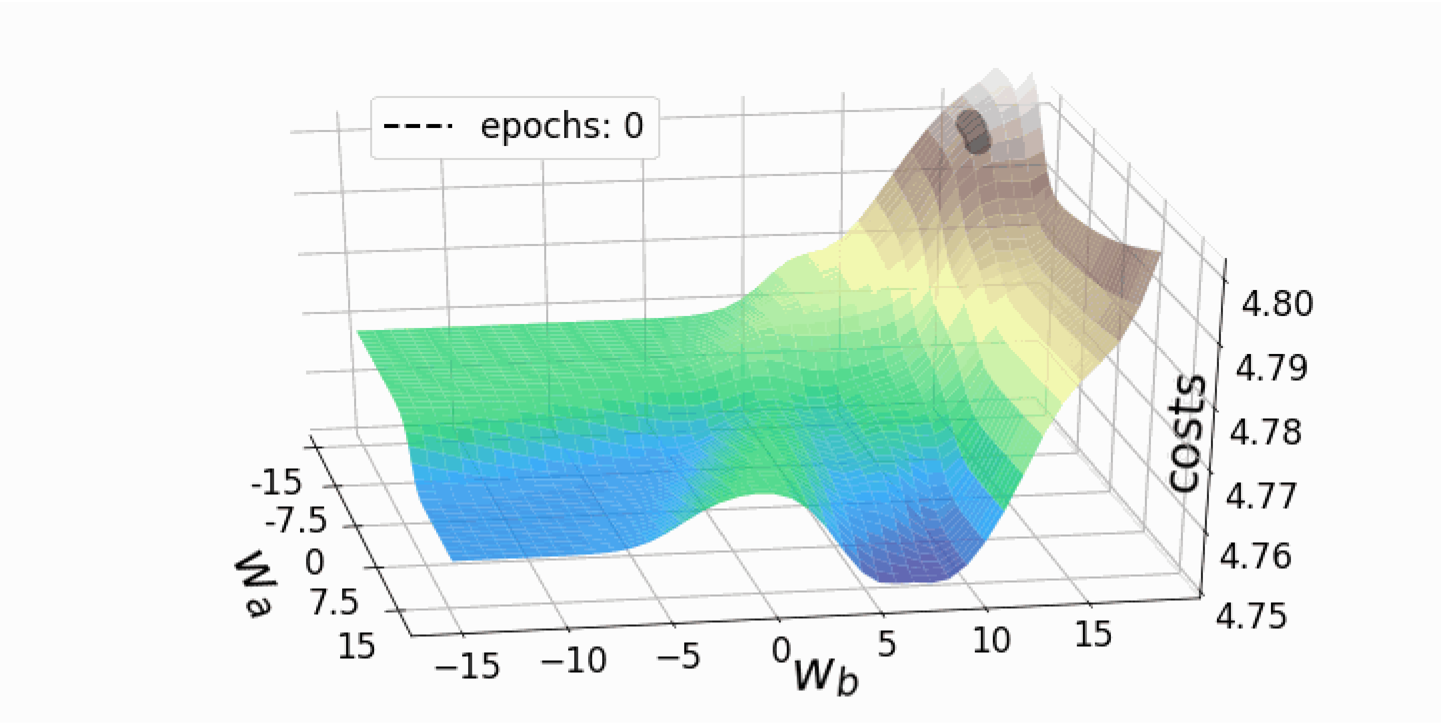
Derivative of Error with respect to weight

Old weight

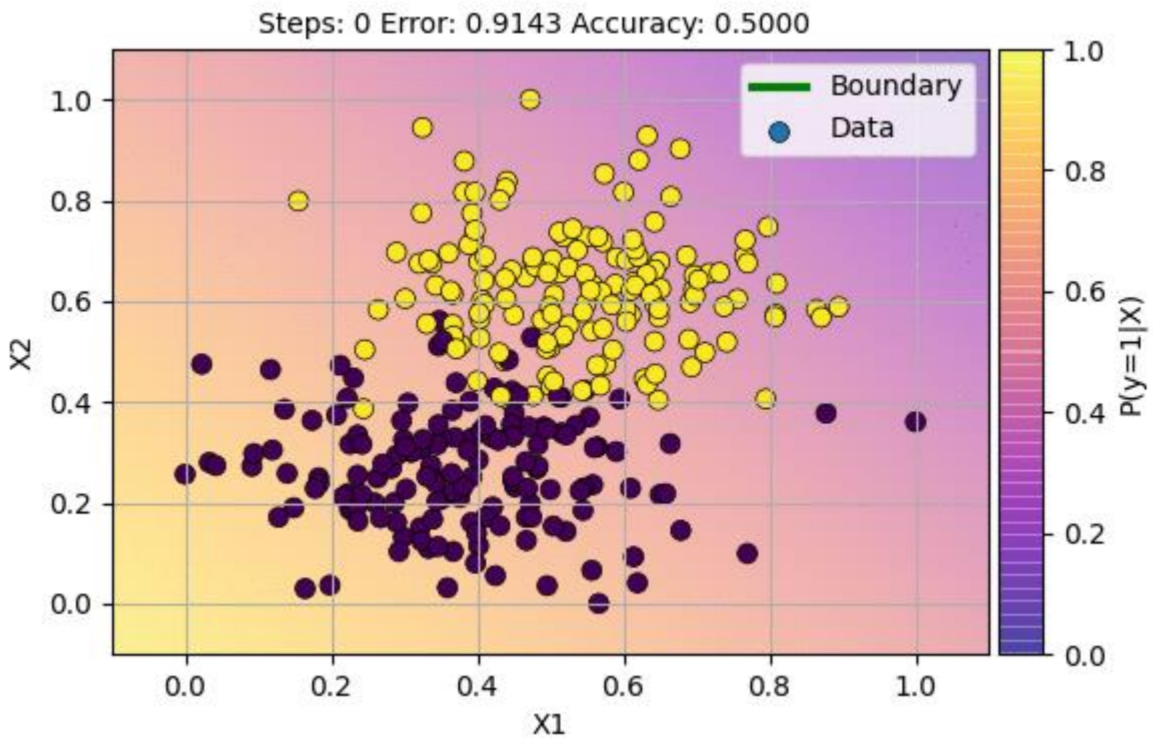
$$ {}^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right) $$

New weight

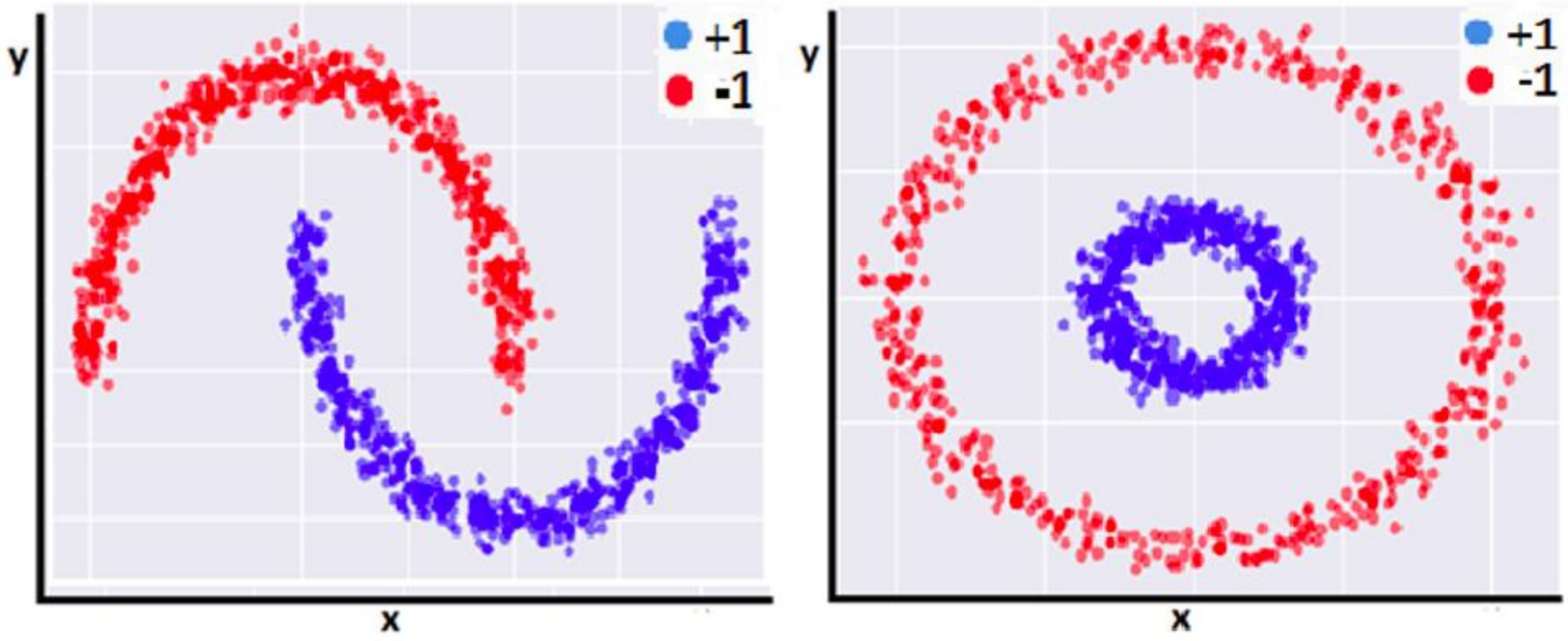
Learning rate



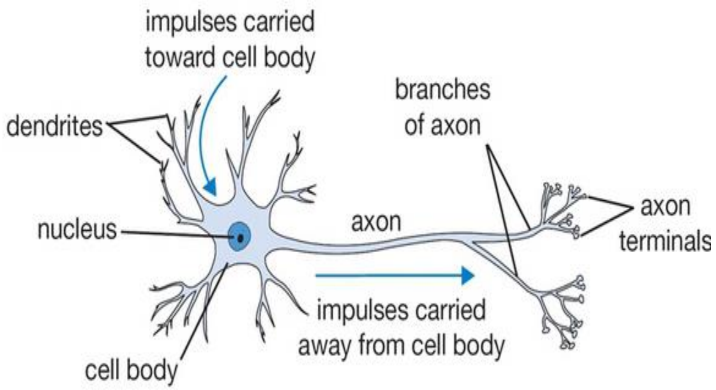
Visualize learning

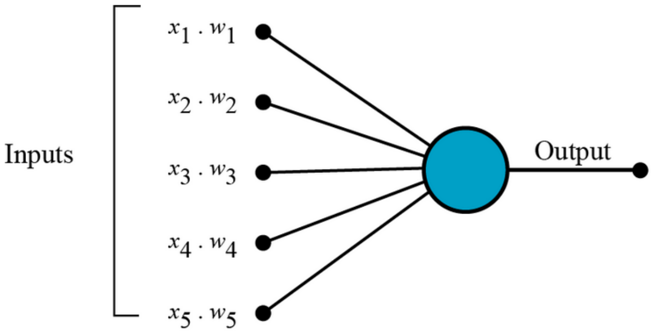


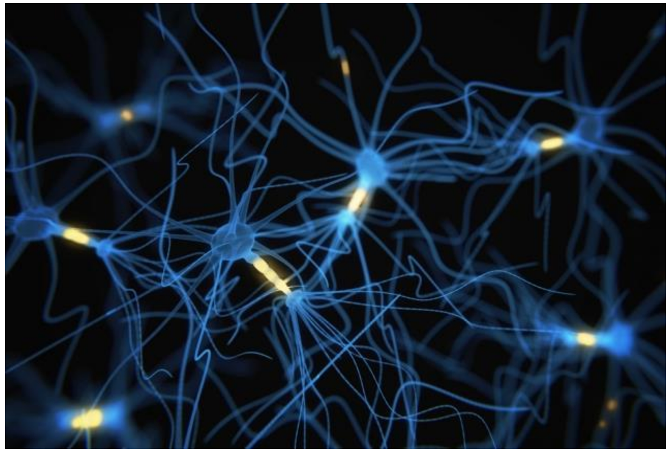
Why not stop here?

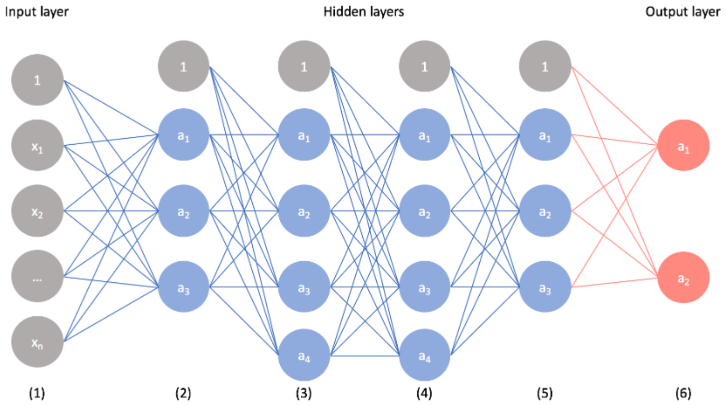


Mimicking our brain

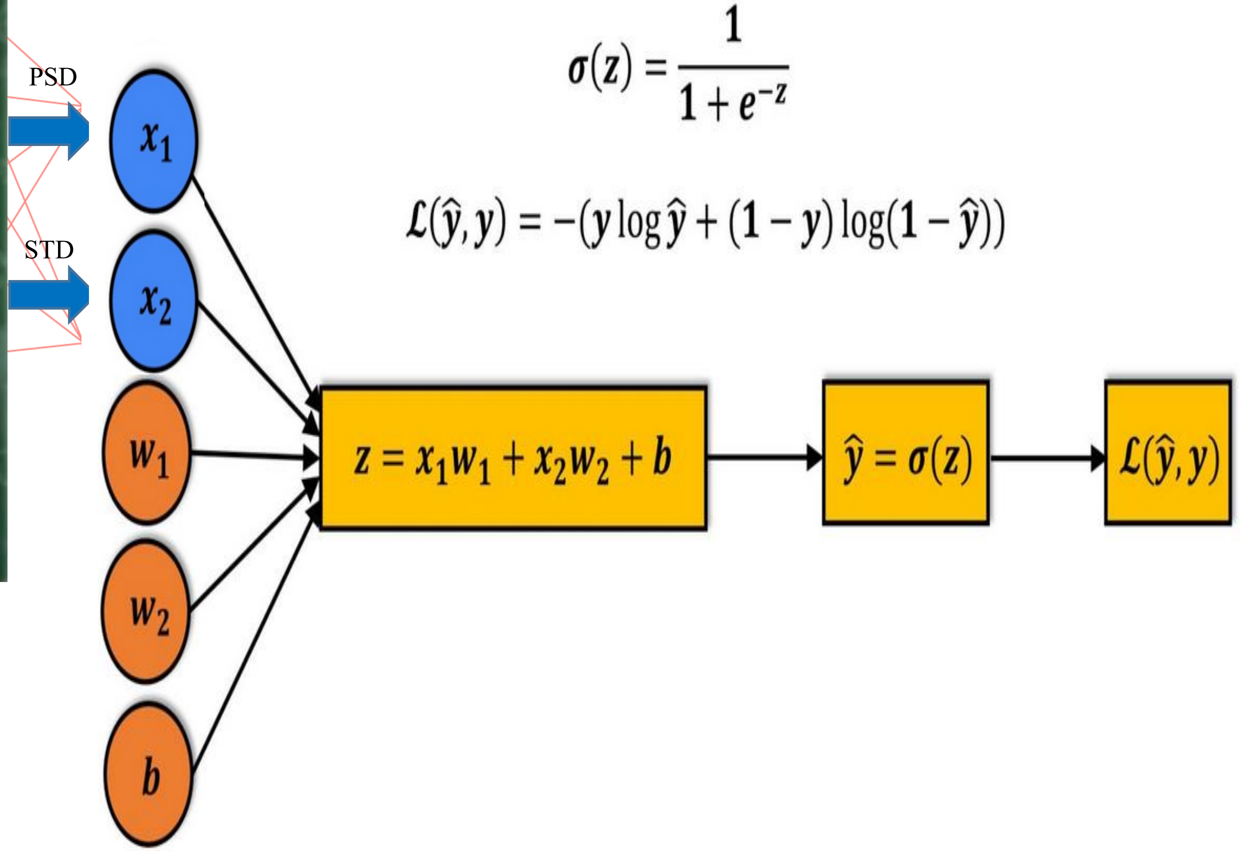




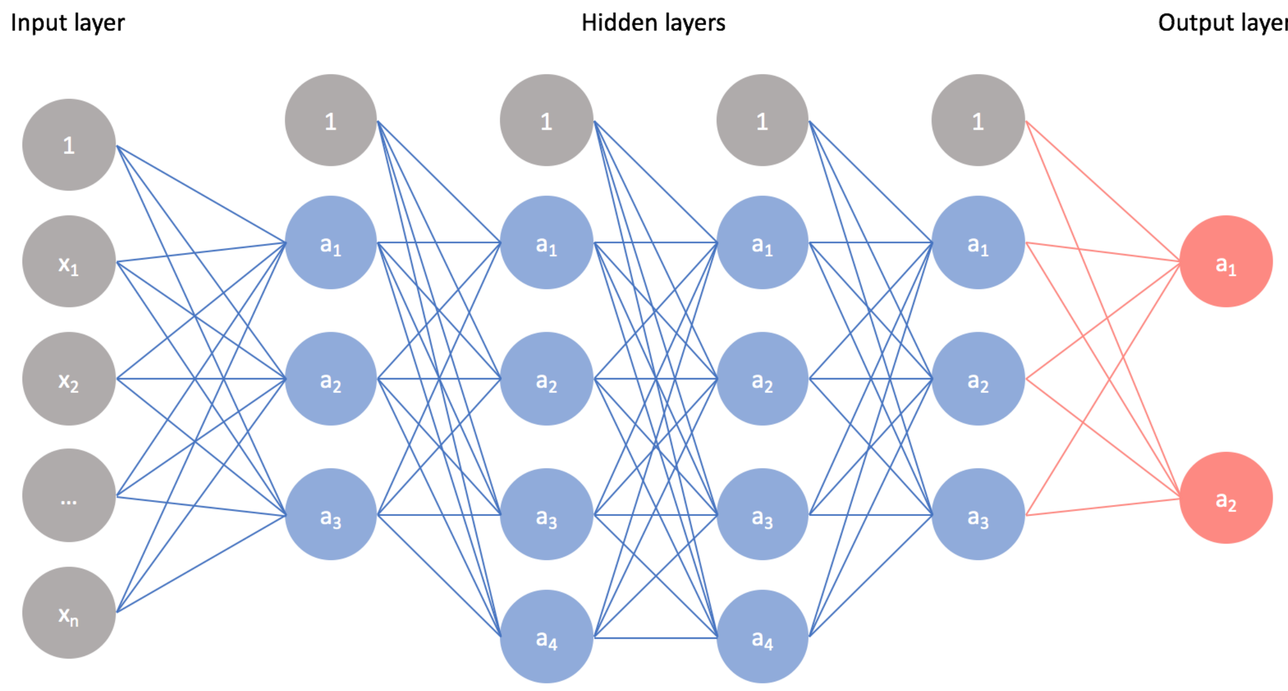


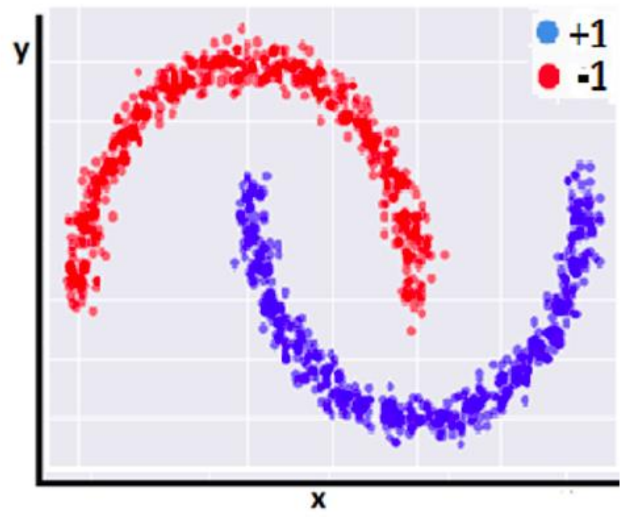


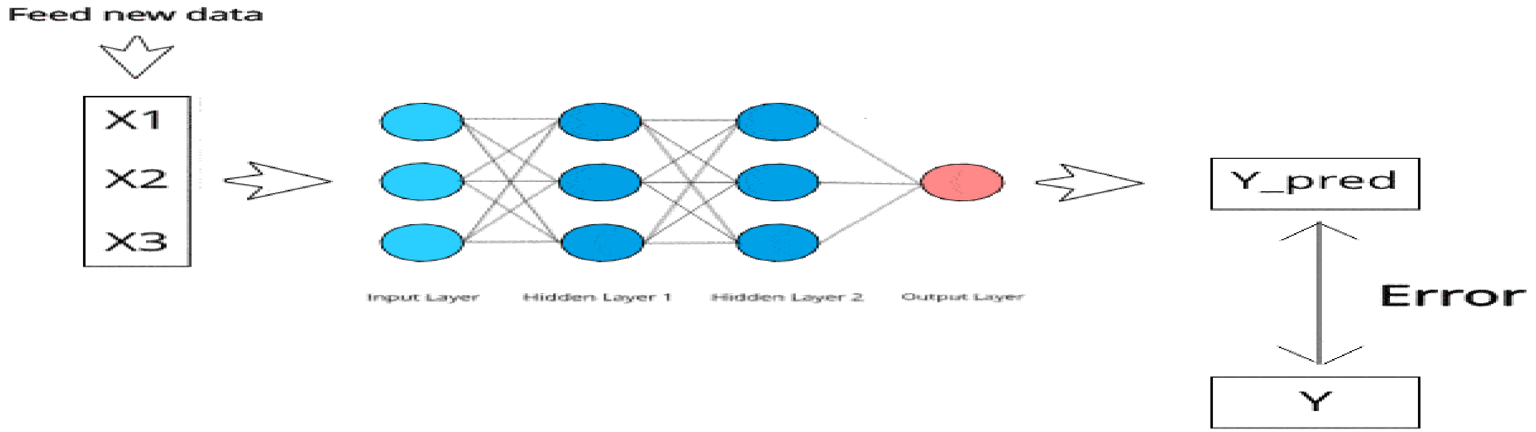
From feature engineering to representation learning



Investing ≠ Wasting


Input layer
Hidden layers
Output layer
1
x_1
x_2
...
x_n
a_1
a_2
a_3
a_4


y
x
+1
-1


Feed new data
X1
X2
X3
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer
Y_pred
Error
Y

Hands on with Python

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```python
X # nX2
y # 1Xn
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```python
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

```python
y_pred_test = log_reg.predict(X_test)
y_pred_proba_test = log_reg.predict_proba(X_test)
```

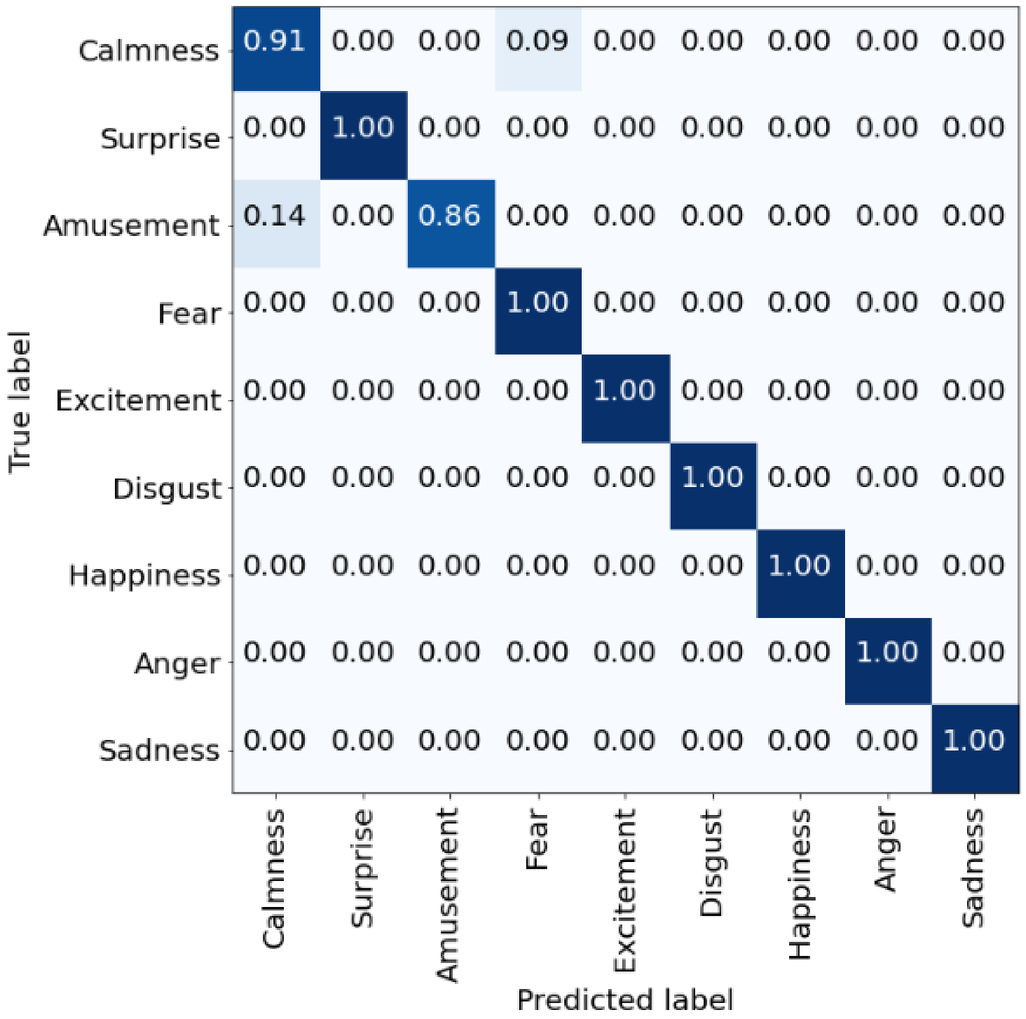
```python
EEG # nXt
```

```python
X_train, X_test, y_train, y_test = train_test_split(EEG, y, test_size=0.2)
```

```python
window_size=60
n_filters_start=64
n_hidden_start=512
len_sub_window=10
dropout=0.5
model = Sequential()
model.add(Conv1D(n_filters_start, len_sub_window, activation='relu', input_shape=(60, 1)))
#----------------------Implement your code here:------------------------------
model.add(Conv1D(2 * n_filters_start, len_sub_window, activation='relu'))
model.add(MaxPool1D())
model.add(Conv1D(4 * n_filters_start, len_sub_window, activation='relu'))
model.add(Dropout(dropout))
model.add(Flatten())
model.add(Dense(n_hidden_start, activation='relu'))
model.add(Dense(int(n_hidden_start / 2), activation='relu'))
model.add(Dense(int(n_hidden_start / 4), activation='relu'))
model.add(Dropout(dropout))
#------------------------------------------------------------------------------
model.add(Dense(1, activation='sigmoid'))
```

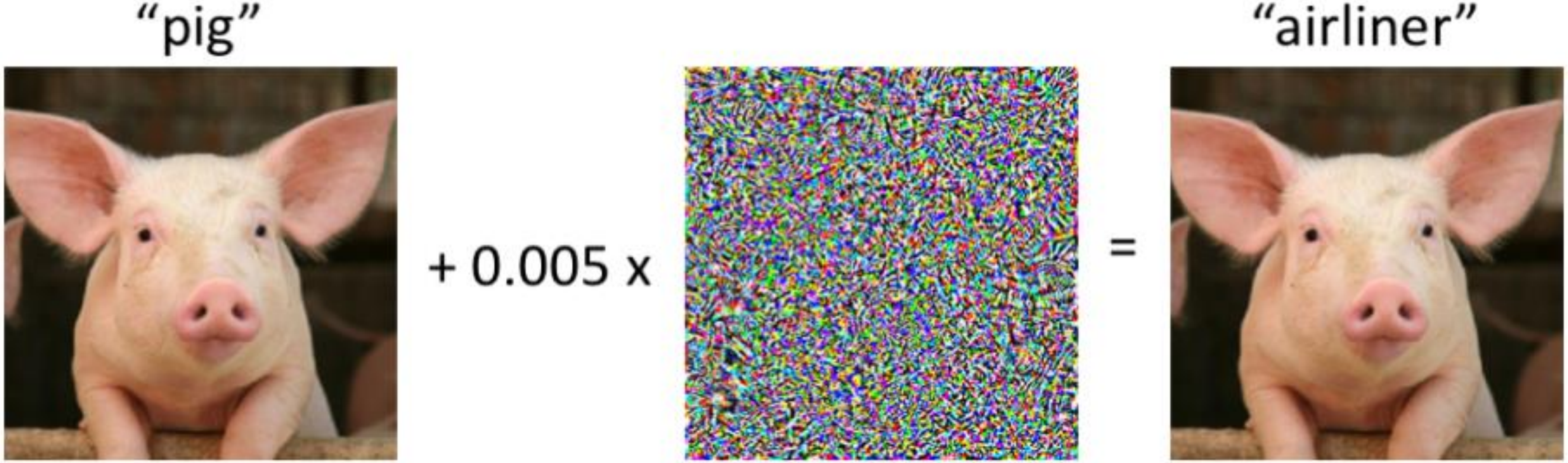
```python
model.compile(optimizer='adam', metrics=['accuracy'], loss='binary_crossentropy')
model.fit(rr_train, y_train, batch_size=1024, epochs=20)
model.predict(X_test)
```

Results for NN




Marketing
is
Everything

Why we should **not** throw away engineering



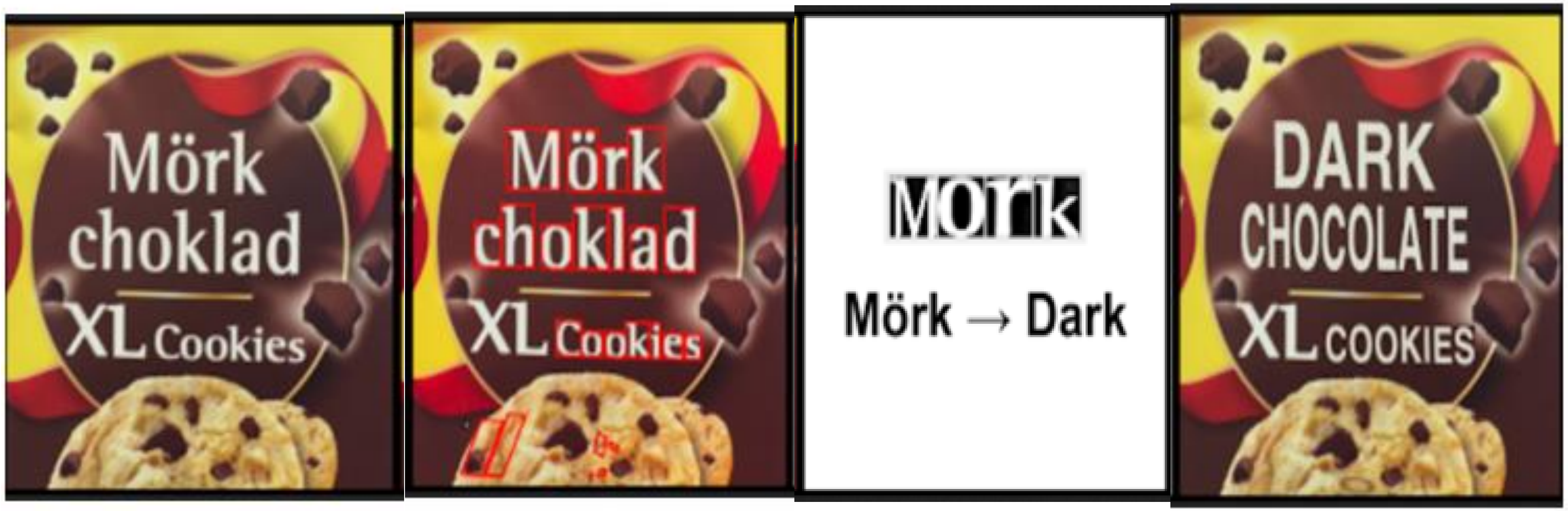
(A) **Cow: 0.99**, Pasture:

(B) No Person: 0.99, Water:

(C) No Person: 0.97,

Beyond Classification

Beyond Classification




A
B
C
D

"That's all Folks!"