

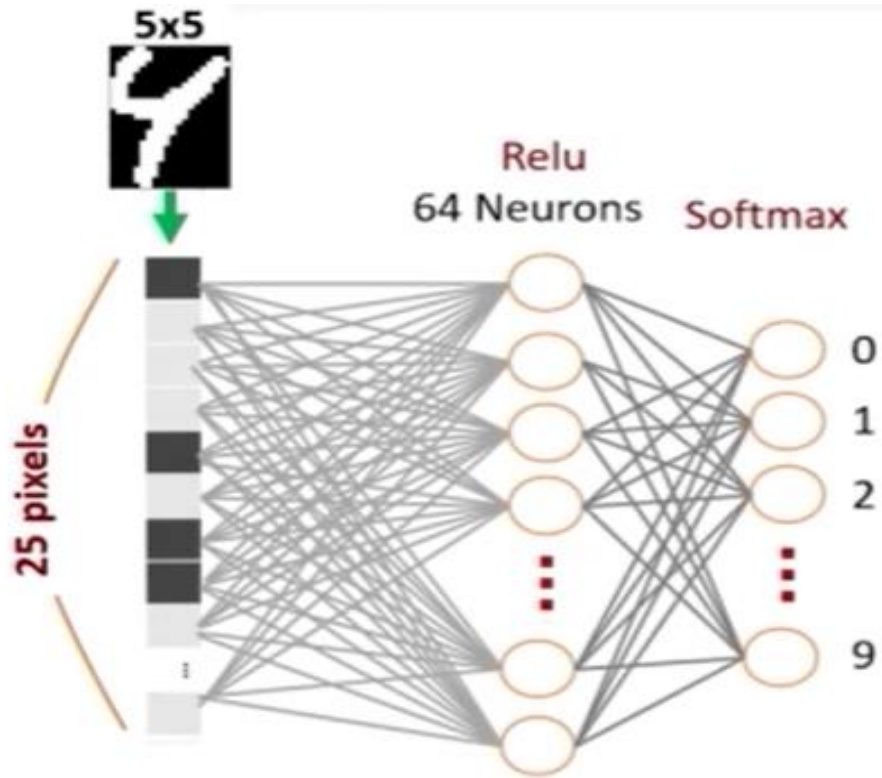
پیاده سازی در کراس

Implementation with Keras!



پیاده سازی در کراس - معماری مدل

```
model = Sequential()  
model.add(Dense(64, activation='relu', input_dim=25))  
model.add(Dense(10, activation='softmax'))
```



تابع Softmax

			One-hot label
● 12	→	$\frac{12}{12 + 20 + 25} \approx 0.21$	0
● 20	→	$\frac{20}{12 + 20 + 25} \approx 0.35$	0
● 25	→	$\frac{25}{12 + 20 + 25} \approx 0.43$	1

$-\log(0.43)$

$$\text{softmax}(L_n) = \frac{e^{L_n}}{\|e^L\|}$$

پیاده سازی در کراس - تابع هزینه و بهینه سازی

```
model.compile(loss='categorical_crossentropy',  
              optimizer='rmsprop',  
              metrics=['accuracy'])
```

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

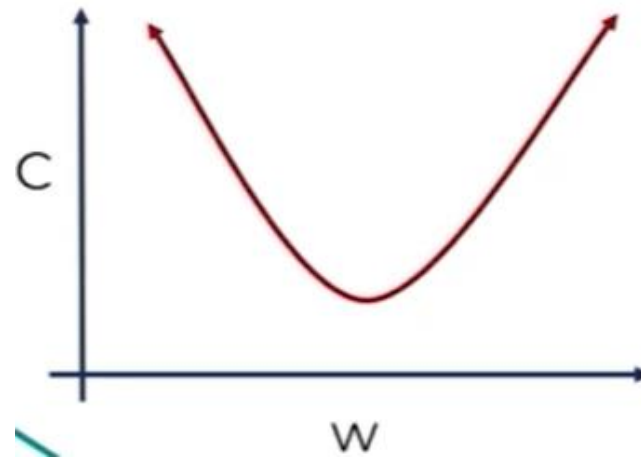
actual probabilities, "one-hot" encoded

$$-\sum Y_i' \cdot \log(Y_i)$$

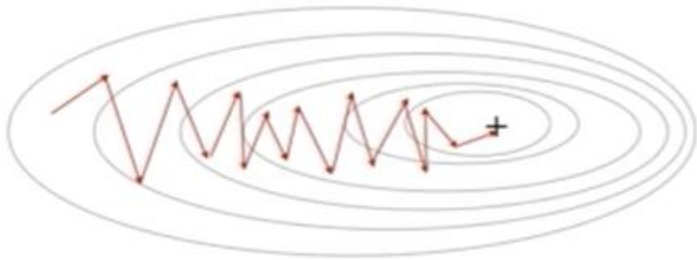
0.1	0.2	0.1	0.3	0.2	0.1	0.9	0.2	0.1	0.1
0	1	2	3	4	5	6	7	8	9

پیاده سازی در کراس:
تعیین داده آموزشی، اندازه mini-batch و تعداد epoch

```
model.fit(x_train, y_train,  
          epochs=30,  
          batch_size=64)
```



Stochastic Gradient Descent



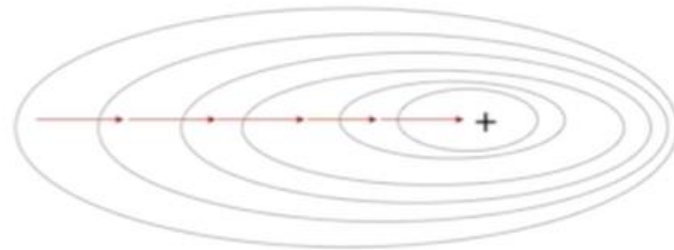
Repeat Until Convergence {

for $i = 1 \dots m$ {

$$\omega \leftarrow \omega - \alpha * \nabla_w L_m(w)$$

}

(Batch) Gradient Descent



Repeat Until Convergence {

$$\omega \leftarrow \omega - \alpha * \nabla_w \sum_{i=1}^m L_m(w) / m$$

}