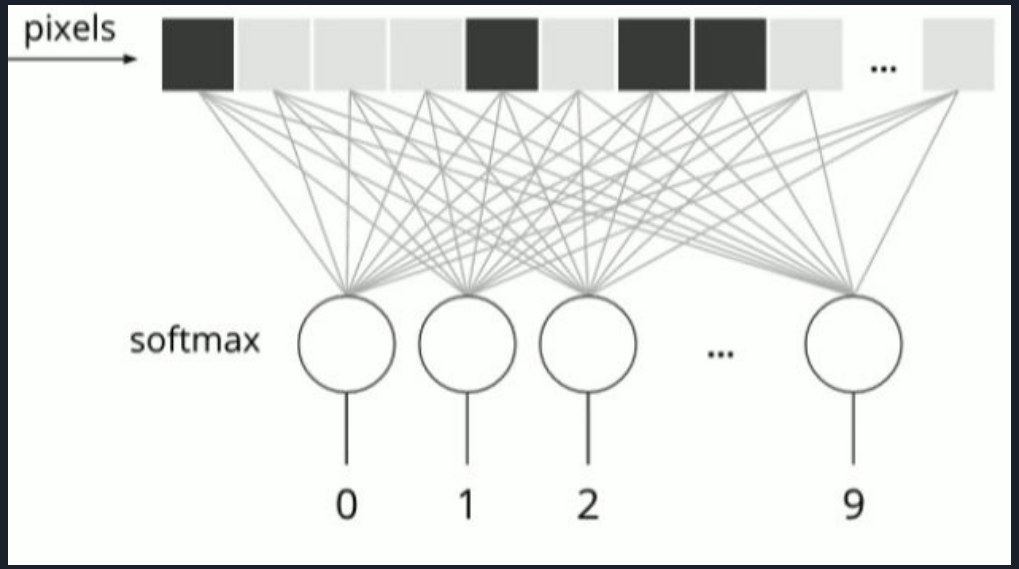
Introducere in Retele Neurale Convolutionale

Studiu de caz pe dataset MNIST

- MNIST : dataset clasic ; recunoasterea cifrelor scrise de mana [0 , 1 , 2 ... 9]
- Format imagine : 28 x 28 pixeli (784 flattened pixels)
- Cea mai simpla forma a unei retele neurale :
  - Un singur layer Fully Connected (FC)
  - 10 neuroni



Functii de activare (1)

- Introduc o non-liniaritate, cu scopul de a putea modela complex features (caracteristici care nu se pot modela cu functii liniare, sau combinatii de functii liniare)
- Pentru clasificare, functia de activare cea mai des intalnita este Softmax :
  - “Softmax function, or normalized exponential function, is a generalization of the logistic function that "squashes" a K-dimensional vector z of arbitrary real values to a K-dimensional vector σ (z) of real values in the range [0, 1] that add up to 1”

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Exemplu Python :

```
>>> z = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
>>> z_exp = [math.exp(i) for i in z]
>>> sum_z_exp = sum(z_exp)
softmax = [round(i / sum_z_exp, 3) for i in z_exp]
>>> print(softmax)
[0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]
```

Functii de activare (2)

- Procesul de normalizare se poate face cu diferite “norme” :
  - L1 (folosit de obicei cu softmax : “Least Absolute Deviations”)
  - L2
  - Norma Euclidiana

- Diferentele dintre L1 si L2 :
  - http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/

- Alte functii de activare :
  - Sigmoid
  - Relu
  - Elu
  - https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Layer de baza pentru o Retea Neurala

$$Y = softmax(X.W + b)$$

MNIST :

- X = Input in retea (batch de 100 de imagini de cate 784 pixeli) => Structura [100,784]
- W = “Weights” / ponderi (variabilele care trebuiesc “invatate”) [784,10]
- B = Biasuri (variabila care shifteaza datele in domeniul potrivit. Este aceeasi pentru toate ponderile) [10]
- Y = Predictii [100,10] (batch de 100 de predictii, 10 valori per predictie)

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

Distanta. Loss. Masura de Eroare (1)

Calitatea predictiilor (Y) este data de valorile din W si b. Pentru a obtine predictii bune, variabilele W si b trebuie antrenate cu pereci [imagine , ground-truth]. Aceasta antrenare corespunde masurarii unei “distante” intre predictii si rezultatul dorit.

- Initial, Y, W si b sunt random. (Tip : look up Xavier initialisation)
- Loss / Distanta / Masura de Eroare :
  - diferenta intre ceea ce masoara sistemul la momentul “t” si ceea ce stim ca este adevarat (ground-truth)

- MNIST :
  - Y va avea 10 valori softmaxed (suma lor = 1)
  - Pentru fiecare imagine din X, construim ground truth tot ca 10 valori (toate elementele 0, mai putin cel pe care imaginea X il reprezinta. Acesta este 1)
  - In acest fel, putem sa calculam in mod direct o distanta dintre elementele din Y si elementele din GT

Distanta. Loss. Masura de Eroare (2)

Exemple des intalnire de functie de calcul pentru distanta :

1. Cross-Entropy $\sum Y_i' . log(Y_i)$
2. L1 $\sum_{i=1}^{n} |y_i - f(x_i)|.$
3. L2 $\sum_{i=1}^{n} (y_i - f(x_i))^2$

Scopul in timpul antrenarii (invatarii) : **Minimizarea distantei**

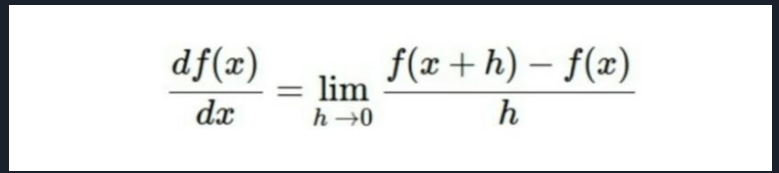
“ Ce valori pentru W si b corespund distantei minime? “

Tips : http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Optimizatori (1)

- Algoritmi existenti care ne ajuta sa minimizam (sau maximizam) o functie E(x)
- E(x) este dependenta de parametrii care se pot antrena / “learnable” folositi in calculul valorilor target (Y) dintr-un set de predictori (X)

- First Order Optimization Algorithms : minimizeaza sau maximizeaza o functie de loss E(x) folosing valorile gradientilor sai, raportat la parametrii. (derivata de ordinul 1)
  - Cel mai popular : Gradient Descent
  - Derivata de ordinul 1 ne ofera directia functiei (o linie tangentiala cu punctul in care o calculam)
  - Un gradient este un vector (generalizare peste mai multe variabile a dY/dX), sau “rata instantanee de schimbare a lui Y, raportat la X”



- Tip : https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f

Optimizatori (2) Stocastic Gradient Descent

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum expensive when N is large!

Approximate sum using a **minibatch** of examples

- R = termen de regularizare (constrange reteaua sa aleaga cea mai simpla solutie)
- Lambda = hyperparametru folosit pentru ponderea lui R

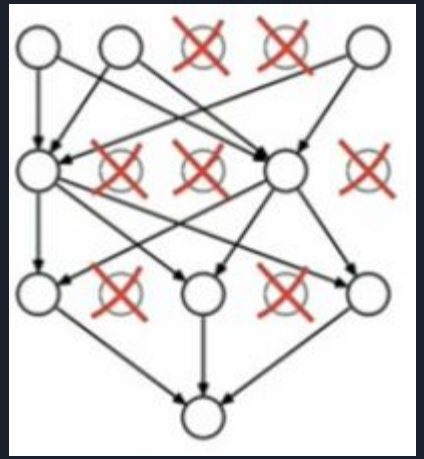
- Minibatching :
  - Optimizare pentru intreaga capacitate a GPU'ului.
  - Asigura ca gradientii sunt calculati peste mai multe exemple (generalizeaza peste dataset mai bine)

Regularizare (1)

```
Yf = tf.nn.relu(tf.matmul(X, W) + B)
Y  = tf.nn.dropout(Yf, pkeep)
```

Dropout :

- La fiecare iteratie, selecteaza random, cu probabilitatea “pkeep”, neuronii care vor ramane in retea
- Pentru ceilalti neuroni ignora outputul (W si b) atat in forward cat si in backward propagation
- Gradientii nu vor avea termeni pentru acesti neuroni ignorati



- Forteaza reteaua sa construiasca “redundant pathways” pentru a reprezenta acelasi “feature-space”. Aceste cai redundante vor face reteaua mai robusta, permitand o generalizare mai buna pe test (mai putin overfitting)
- Se aplica doar pe training! Pe test, toti neuronii sunt activi (pkeep=1.0)

Regularizare (2)

Regularizare L2 (“weight decay”) :

- Cea mai comuna forma de regularizare.
- Penalizeaza vectorii W care sunt foarte puternic reprezentativi (peaks)
- Prefera vectori W care sunt difuzi.

$$L_\lambda(\mathbf{w}) = L(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2.$$

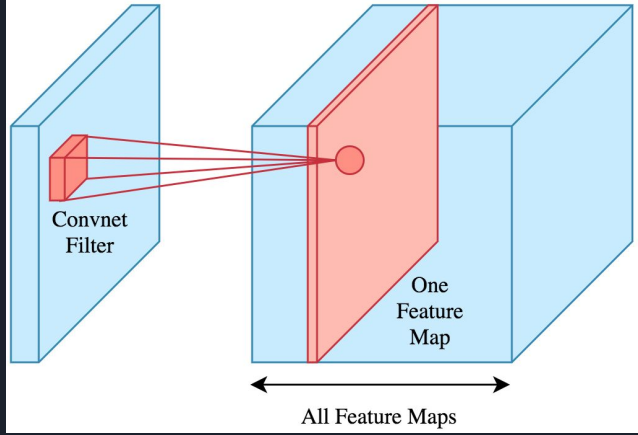
Regularizare L1 :

- Pentru fiecare W adaugam doar termenul $\lambda|w|$
- Efectul este folosirea unui subset din cele mai importante inputuri => invarianta la noise

Tip : https://arxiv.org/pdf/1706.05350.pdf

Retele Convolutionale (1)

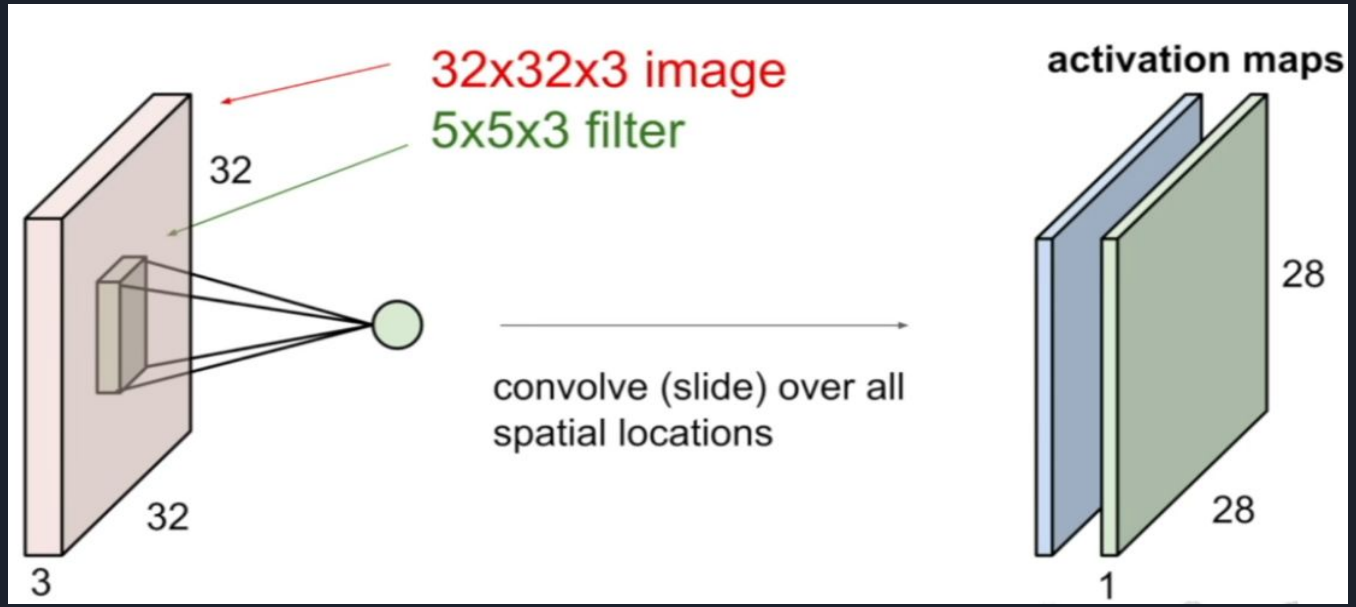
Functioneaza pe acelasi principiu de multiplicare a W cu valori din input si adaugare de bias.



- Un filtru este conectat la un “patch” din imagine, nu la fiecare pixel (diferit de FC)
- Fiecare filtru este mutat (cu padding corespunzator) peste toata imaginea, pentru a produce un “feature map”.
- Un feature map corespunde aceluiasi set de W folositi de un filtru.
- Filtrele au mereu acelasi depth cu volumul de input (3 in cazul primului layer Conv, daca imaginea este RGB)
- “Convolve” : (terminologie provenita din procesarea de semnal)
  - “Slide over the image, spatially, computing dot products”

- Pentru grade de libertate aditionale, putem sa construim un layer convolutional cu multiple “feature maps”, care corespund filtrelor asociate,si ale caror parametrii pot fi antrenati.

Retele Convolutionale (2)

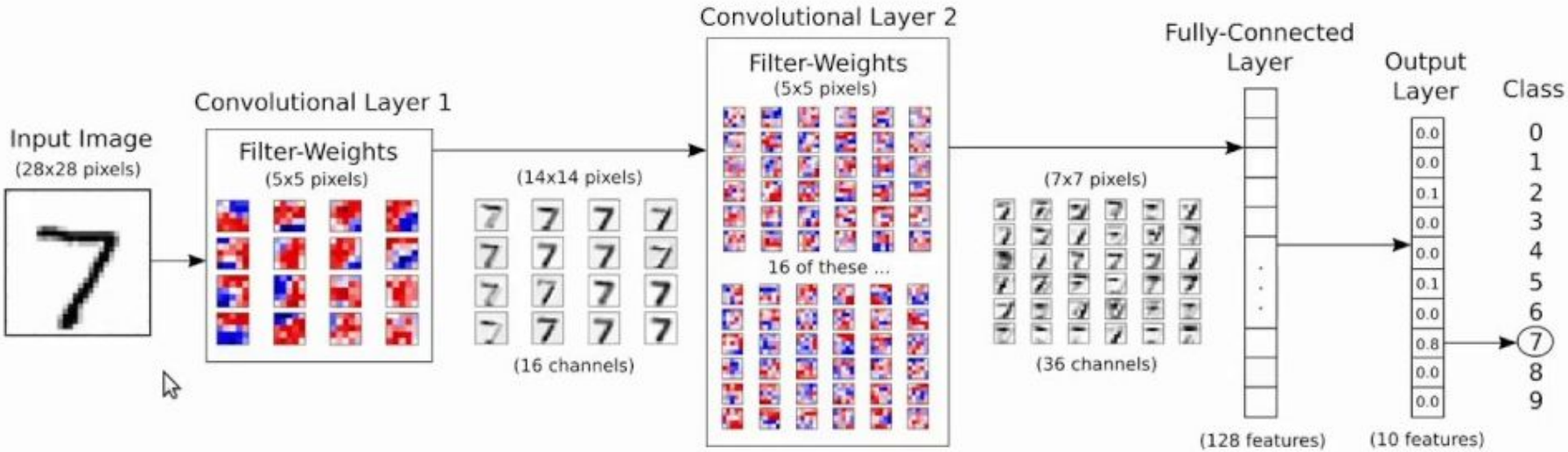


Distilarea informatiei :

- Stride
- Max pool

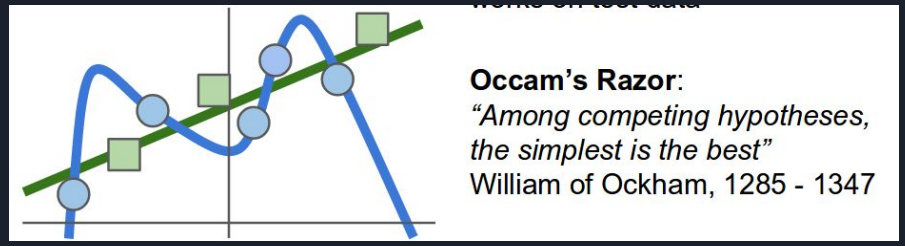
Tips :
https://youtu.be/bNb2fEVKeEo

Retele Convolutionale (3)



- Fiecare filtru “evolueaza” intr-un “shape recognizer” / “feature recognizer”
- Pentru a distila informatia prin retea, pana la valorile de Y de la sfarsit, jonglam cu numarul de filtre, stride si padding, sau “downsampling” prin maxpooling.

Tricks. Terminologie.



Epoca :

- Numarul de iteratii necesare pentru ca reteaua sa vada intregul set de training o data.
- DatasetSize / (batchSize * NumIterations) :
  - Trebuie sa fie suficient de mare incat Loss'ul sa se stabilizeze
  - Trebuie sa fie suficient de mic incat sa nu apara efectele "overfittingului"

Overfitting :

- Fenomen care apare in momentul in care reteaua incepe sa invete mot-a-mot anumite caracteristici ale datasetului (pierde capacitatea de generalizare pe seturi noi)
- Se intampla de obicei cand sunt prea multe grade de libertate (spatiul parametric al W si b) pentru un set suficient de mic / non-divers de training.
- Se poate observa prin plotarea curbelor de invatare pe seturi de "validare" si "test"
  - High accuracy on validation/train + Lower accuracy on test ⇒ Overfitting

Q & A
:)