Stochastic Gradient Descent

Foundations of Data Analysis

04/17/2023

Stochastic Gradient Descent

- Goal of machine learning :
  - Minimize expected loss (error)

$$\min_h L(h) = \mathbf{E}\left[\text{loss}(h(x), y)\right]$$

given samples

$$(x_i, y_i) \;\; i = 1, 2 \ldots m$$

Gradient Descent

- Process all examples together in each step

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial L(w, x_i, y_i)}{\partial w} \right)$$

where L is the regularized loss function

- Entire training set examined at each step
- Very slow when *n* is very large

Stochastic Gradient Descent

- “Optimize” one example at a time
- Choose examples randomly (or reorder and choose in order)
  - Learning representative of example distribution

for $i = 1$ to n:

$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \frac{\partial L(w, x_i, y_i)}{\partial w}$$

where L is the regularized loss function

Stochastic Gradient Descent

for $i = 1$ to n:

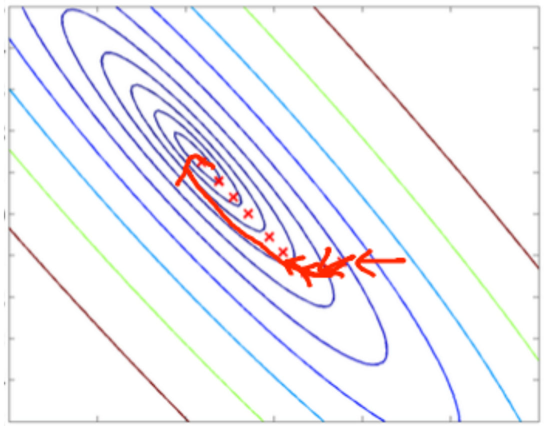
$$w^{(k+1)} \leftarrow w^{(k)} - \eta_t \frac{\partial L(w, x_i, y_i)}{\partial w}$$

where L is the regularized loss function

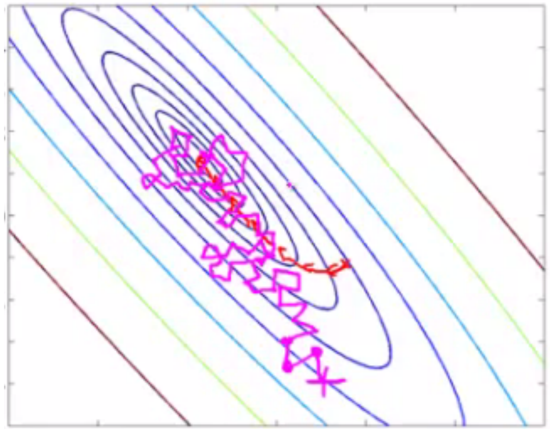
- Equivalent to online learning (the weight vector *w* changes with every example)
- Convergence guaranteed for convex functions (to local minimum)

Issues of Stochastic Gradient Descent

- Convergence very sensitive to the learning rate (oscillations near solution due to probabilistic nature of sampling)
  - Might need to decrease with time to ensure the algorithm converges eventually
- Basically – SGD is good for machine learning with large data sets!



GD



SGD

Network Parameters

- How are the weights initialized?
- How many hidden layers and how many neurons?
- How many examples in the training set?

Weights

- In general, initial weights are randomly chosen, with typical values between -1.0 and 1.0 or -0.5 and 0.5.

- There are two types of NNs. The first type is known as
  - Fixed Networks – where the weights are fixed
  - Adaptive Networks – where the weights are changed to reduce prediction error.

Size of Training Data

- Rule of thumb:
  - the number of training examples should be at least five to ten times the number of weights of the network.

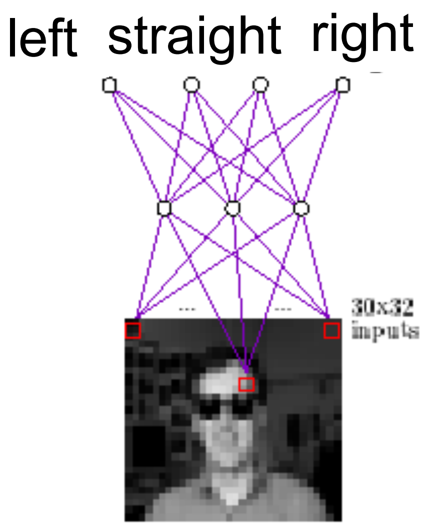
- Other rule:

$$N > \frac{|W|}{(1-a)}$$

|W|= number of weights

a = expected accuracy on test set

Face Recognition



Input images

90% accurate learning head pose, and recognizing 1-of-20 faces

Disadvantages of Network

- The individual relations between the input variables and the output variables are not developed by engineering judgment so that the model tends to be a black box or input/output table without analytical basis.

- The sample size has to be large.

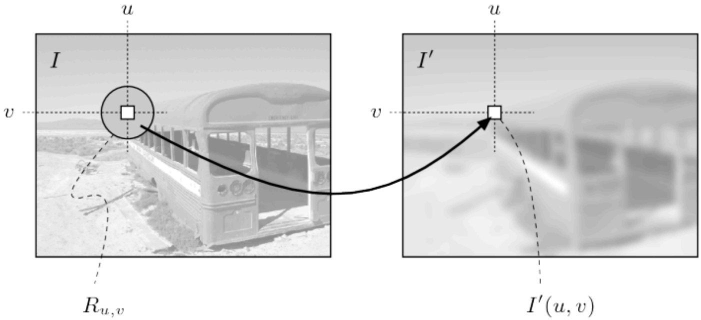
- Requires lot of computation so training can be time consuming.

Convolution

Spatial Filters

Definition

A **spatial filter** is an image operation where each pixel value $I(u, v)$ is changed by a function of the intensities of pixels in a neighborhood of (u, v).



What Spatial Filters Can Do

Blurring/Smoothing

→

What Spatial Filters Can Do

Sharpening

$\rightarrow$

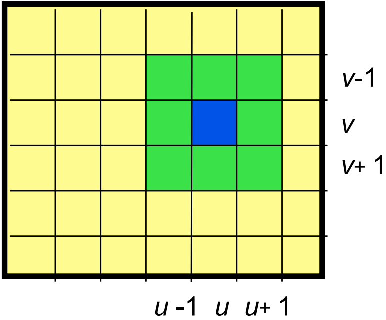
What Spatial Filters Can Do

Weird Stuff

→

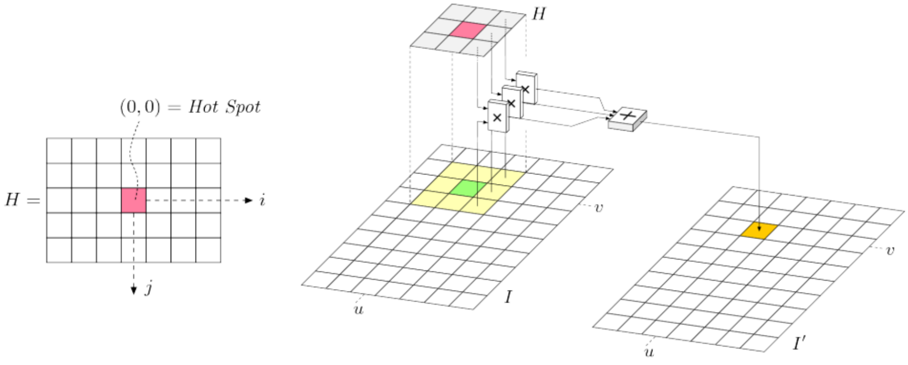
Example: The Mean of a Neighborhood

Consider taking the mean in a 3×3 neighborhood:



$$I'(u,v) = \frac{1}{9} \sum_{i=-1}^{1} \sum_{j=-1}^{1} I(u+i, v+j)$$

How a Linear Spatial Filter Works



H is the filter "kernel" or "matrix"

For the neighborhood mean: $H(i, j) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

General Filter Equation

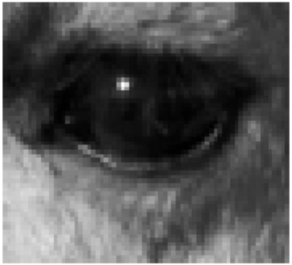
Notice that the kernel *H* is just a small image!

Let $H : R_H \rightarrow [0, K-1]$

$$I'(u,v) = \sum_{(i,j) \in R_H} I(u+i, v+j) \cdot H(i,j)$$

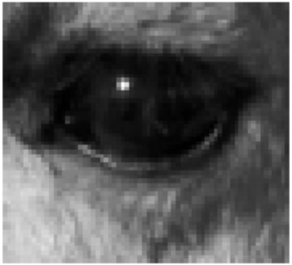
This is known as a **correlation** of *I* and *H*

What Does This Filter Do?



0	0	0
0	1	0
0	0	0

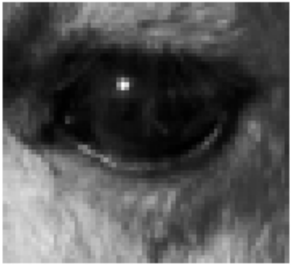
What Does This Filter Do?



0	0	0
0	1	0
0	0	0

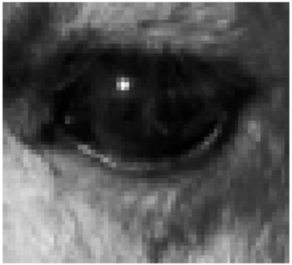
Identity function (leaves image alone)

What Does This Filter Do?



0	0	0
0	0	1
0	0	0

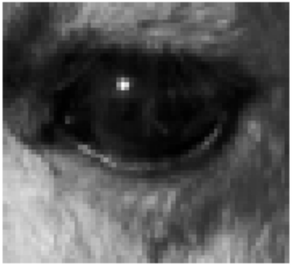
What Does This Filter Do?



0	0	0
0	0	1
0	0	0

Shift left by one pixel

What Does This Filter Do?



$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen (identity minus mean filter)

Filter Normalization

1. Notice that all of our filter examples sum up to one
2. Multiplying all entries in H by a constant will cause the image to be multiplied by that constant
3. To keep the overall brightness constant, we need H to sum to one

$$
\begin{aligned}
I'(u,v) &= \sum_{i,j} I(u+i, v+j) \cdot (cH(i,j)) \\
&= c \sum_{i,j} I(u+i, v+j) \cdot H(i,j)
\end{aligned}
$$

Effect of Filter Size

Mean Filters:

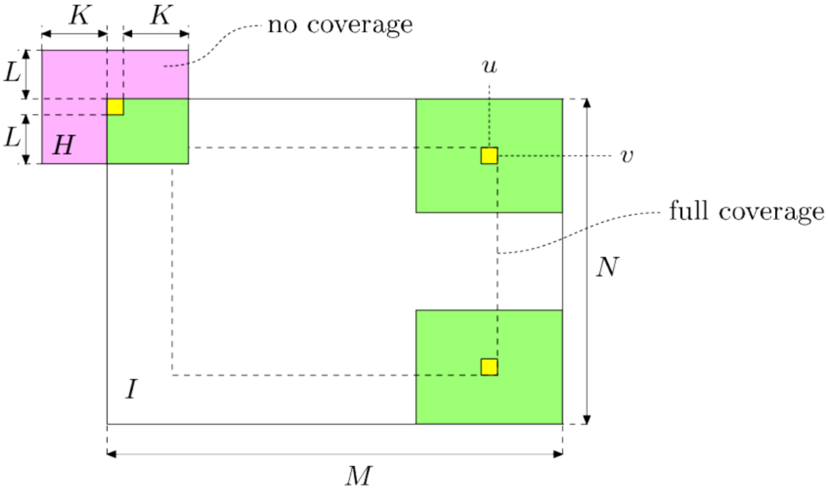
Original

7×7

15×15

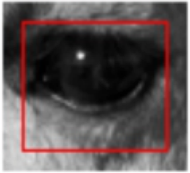
41×41

What To Do At The Boundary?



What To Do At The Boundary?

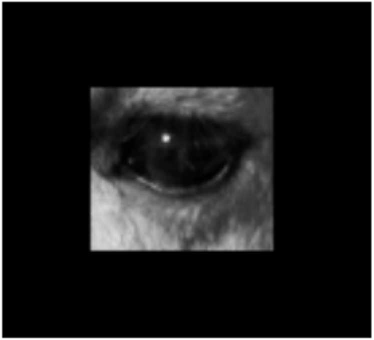
Crop



What To Do At The Boundary?

Crop

Pad

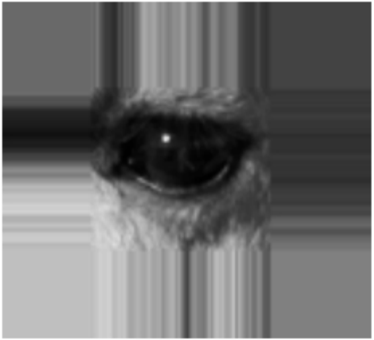


What To Do At The Boundary?

Crop

Pad

Extend



What To Do At The Boundary?

Crop

Pad

Extend

Wrap

Convolution

Definition

Convolution of an image I by a kernel H is given by

$$I'(u,v) = \sum_{(i,j)\in R_H} I(u-i, v-j) \cdot H(i,j)$$

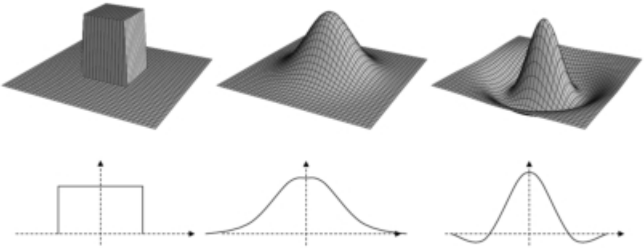
This is denoted: $I' = I * H$

1. Notice this is the same as correlation with H , but with negative signs on the I indices
2. Equivalent to vertical and horizontal flipping of H:

$$I'(u,v) = \sum_{(-i,-j)\in R_H} I(u+i, v+j) \cdot H(-i,-j)$$

Some More Filters

Box | Gaussian | Laplace



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

(a)

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

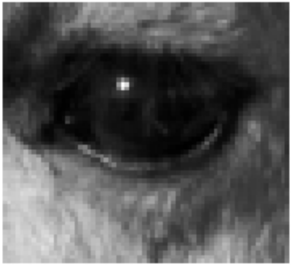
(b)

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

(c)

Convolutional Neural Networks (CNNs)

Learning a Filter

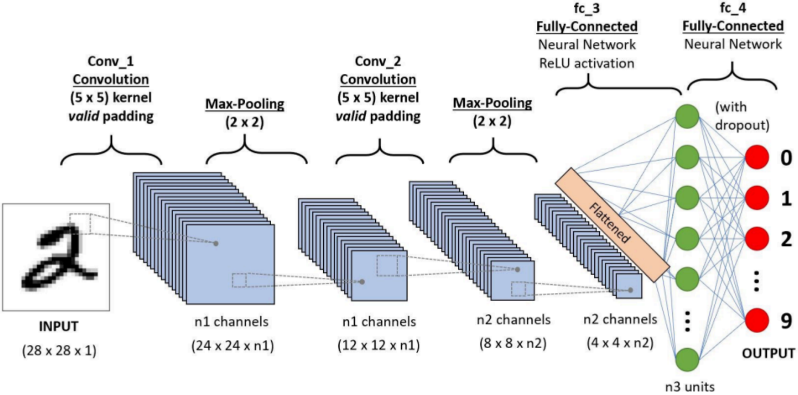


w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

?

Filter consists of weights that need to be learned.

Convolutional Neural Networks



http://towardsdatascience.com/