中国传媒大学

深度学习与类脑计算

（六）

曹立宏

脑科学与智能媒体研究院

The Last Homework

- Theory
  - 请叙述并推导Logistic Regression中参数的MAP估计表达。
  - 请叙述并推导SVM在不可分数据情况下的最优解表达。
  - 请叙述并推导线性PCA中的主成分表达式。
- Practice
  - （本题可以组队完成）收集中传男女学生身高体重信息，
    - 分别采用1维和2维Logistic Regression的方法实现ML和MAP估计。
    - 用SMV的方法，依据中传学生身高体重信息对性别分类。
    - 比较以上两种方法的结果。
  - 数据Cat4D3Groups是4维观察数据，
    - 请先采用MDS方法降维到3D，形成Cat3D3Groups数据，显示并观察。
    - 对Cat3D3Groups数据采用线性PCA方法降维到2D，形成Cat2D3Groups数据，显示并观察。
    - 对Cat2D3Groups数据采用K-Mean方法对数据进行分类并最终确定K，显示分类结果。
    - 对Cat2D3Groups数据采用Hierarchical分类法对数据进行分类，并显示分类结果。

在Python中使用最优化算法可以参考scipy.optimize
http://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

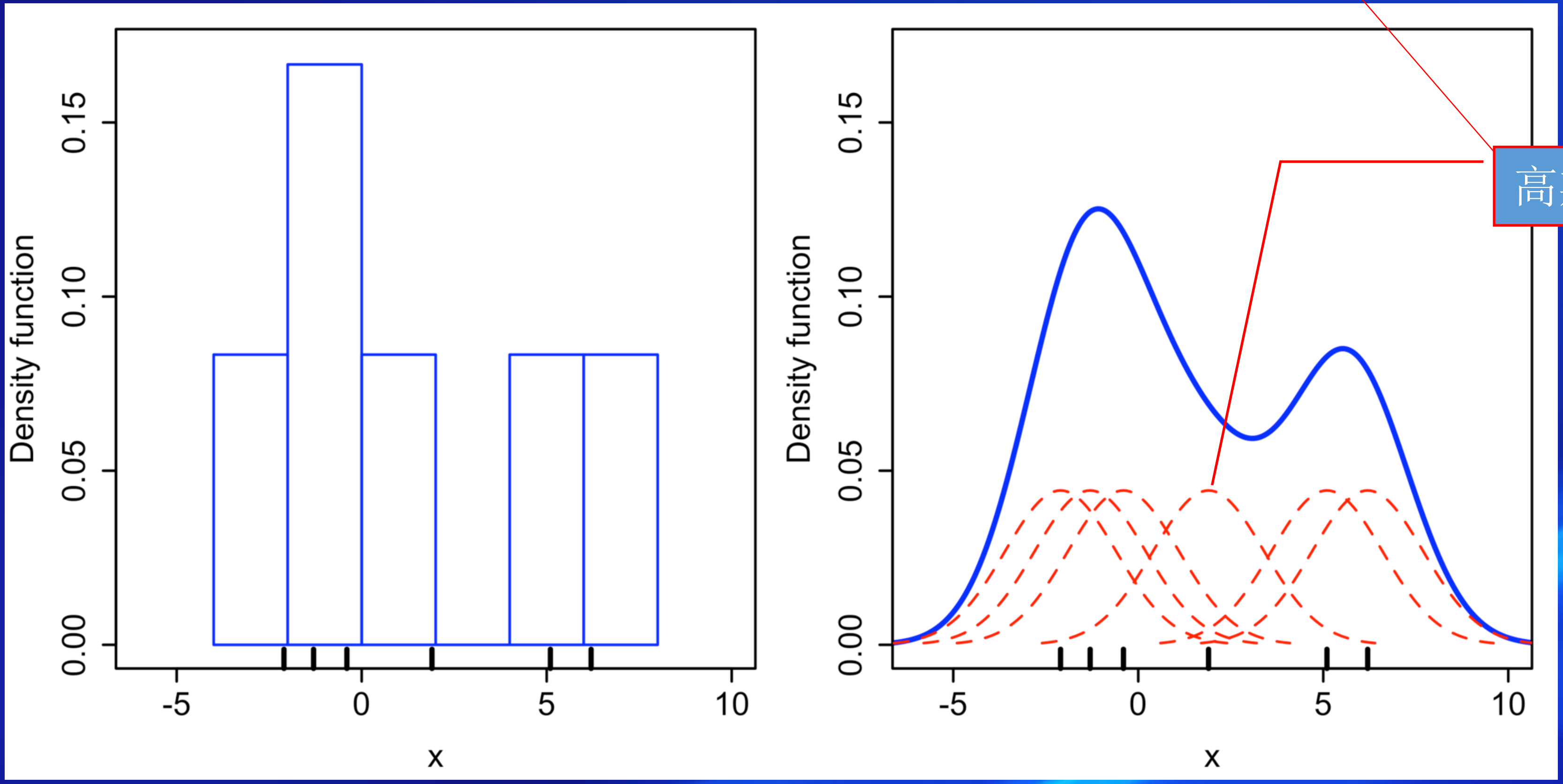
复习

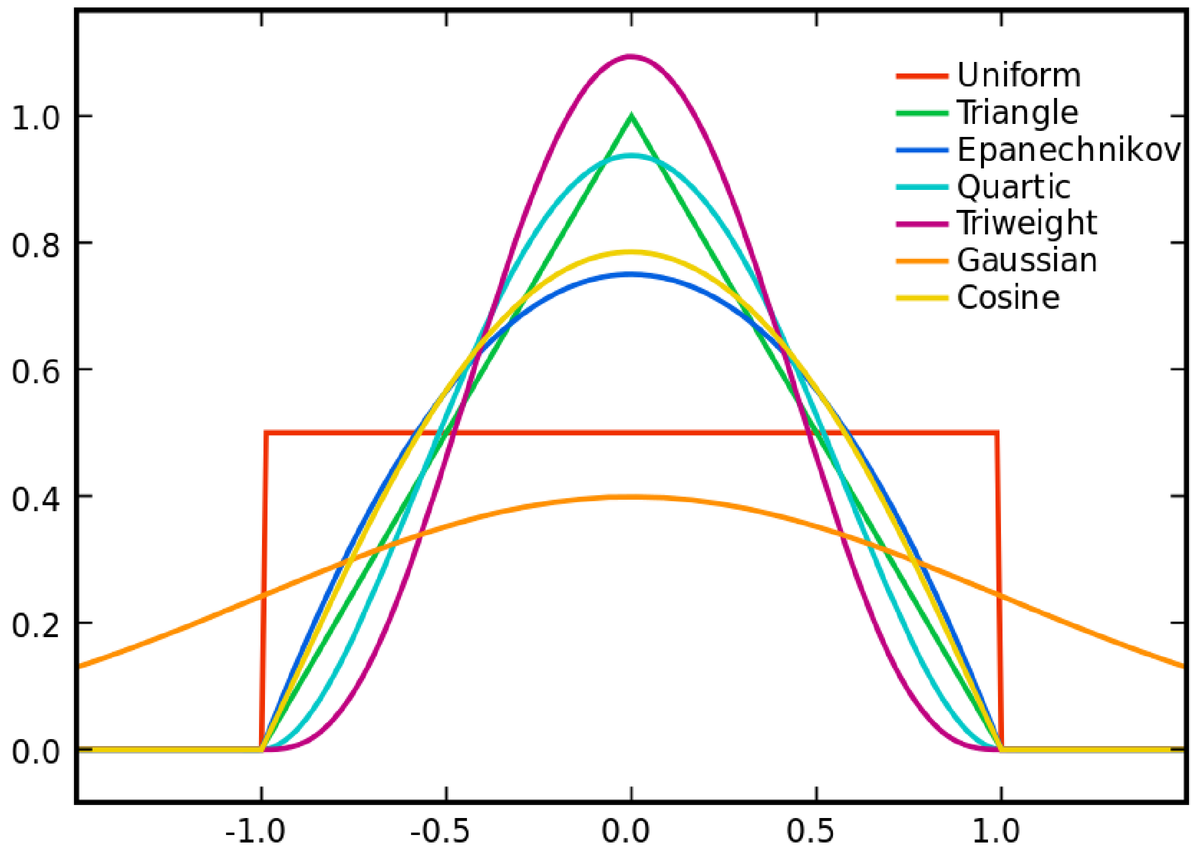
- 分类问题
  - Supervised
    - Logistic Regression
    - SVM
  - Unsupervised
    - Dimension Reduction
      - CPA
      - MDS
      - IsoMap
    - Clustering
      - K-Mean
      - Hierarchical

Kernel Method – Density Estimation

Histograms

$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} k(\frac{x - x_i}{h})$$





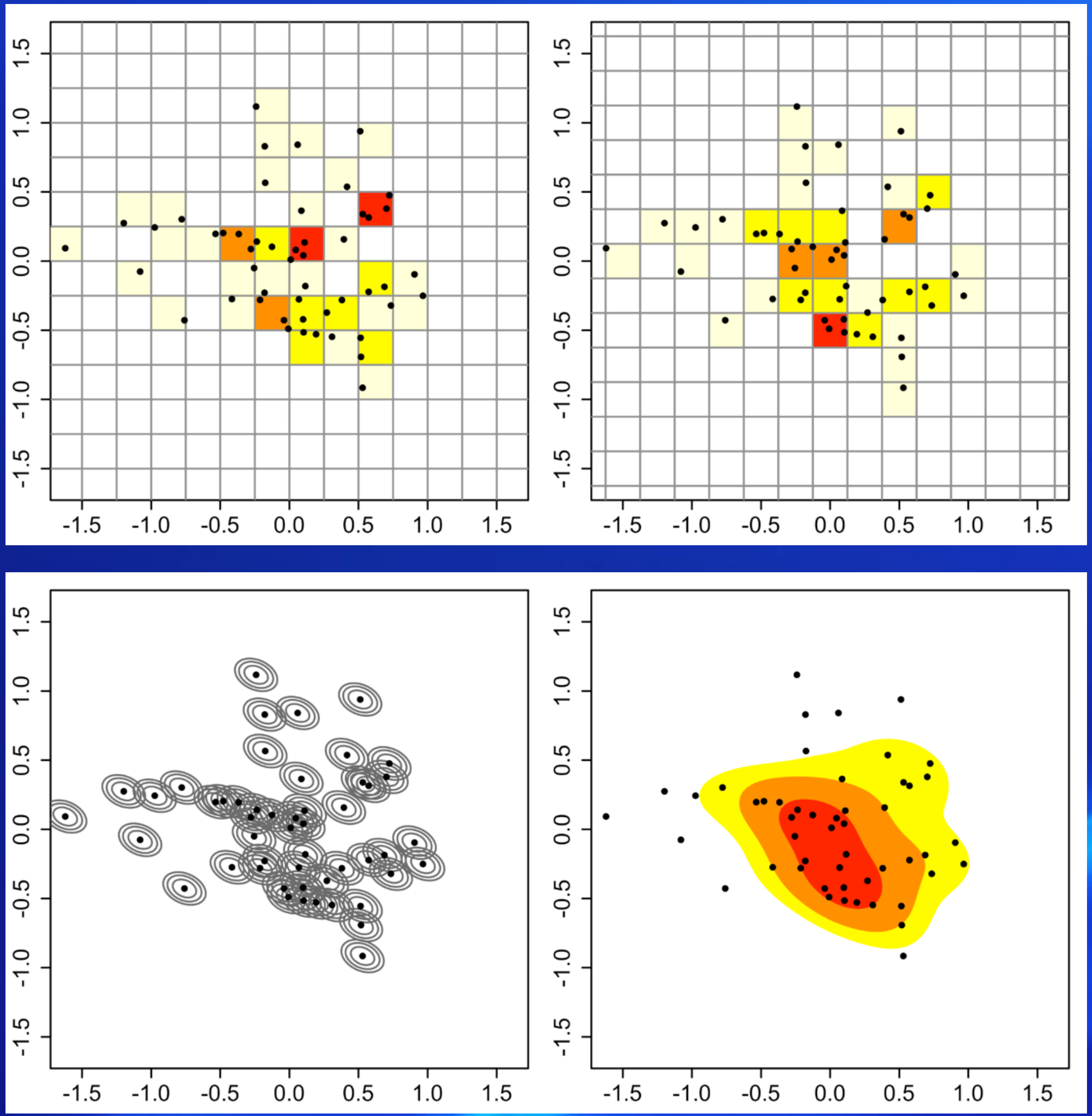
Kernel是距离、相似度的一种度量

2D

Histograms

$$\hat{f}(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} k_{\Sigma}(\vec{x} - \vec{x}_i)$$

$$k_{\Sigma}(\vec{x}) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}}\exp(-\frac{1}{2}\vec{x}^T\Sigma^{-1}\vec{x})$$



From Euclidean Space to Hilbert Space

- Extension 1
  - Inner Product Space
    - Symmetric
    - Linearity
    - **Positive Definite**
- Extension 2
  - Vector to Function

$$\langle , \rangle : \quad R^m \times R^m \to R$$

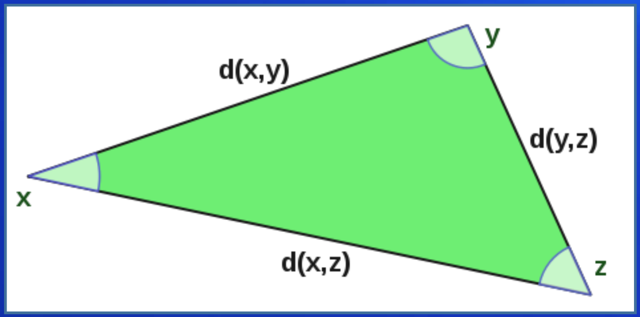
$$\langle \vec{x}, \vec{y} \, \rangle = \sum_{i=1}^{m} x_i y_i$$

$$\langle f, g \rangle = \int f(x) g(x) dx$$

Distance in Hilbert Space

- Vector to vector

$$d(\vec{x}, \vec{y}) = \sqrt{\langle \vec{x}, \vec{y} \rangle}$$



- Function to function

$$d(f, g) = \sqrt{\langle f, g \rangle} = \left(\int f(x) g(x) dx \right)^{1/2}$$

基向量-基函数

- 基向量

$$\langle \vec{e}_i, \vec{e}_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

$$\vec{x} = \sum_{i=1}^{m} \lambda_i \vec{e}_i$$

$$\lambda_i = \langle \vec{x}, \vec{e}_i \rangle$$

- 基函数

$$\langle e_i, e_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

$$\varphi(x) = \sum_{i=1}^{+\infty} \lambda_i e_i(x)$$

$$\lambda_i = \langle \varphi, e_i \rangle$$

特征向量-特征函数

- 特征向量 of A:

$$A\vec{x} = \lambda\vec{x}$$

- 特征函数 of 线性变换T:

$$T(f) = \lambda f$$

- 核函数变换是线性变换：

$$K(f(x)) \triangleq [Kf](x) = \int K(x,s)f(s)ds$$

如果A是半正定矩阵 存在正交矩阵P，使得

$$P^{-1}AP = \begin{pmatrix} \lambda_1 & \cdots & \\ \vdots & \lambda_i & \vdots \\ & \cdots & \lambda_m \end{pmatrix}, \quad \lambda_i \geq 0$$

正定核函数

- 对于对称核函数$K(x,s)$，任选格点(u_i,v_j)，以下矩阵为半正定矩阵

$$\begin{pmatrix} k(u_1,v_1) & \cdots & k(u_1,v_N) \\ \vdots & k(u_i,v_j) & \vdots \\ k(u_N,v_1) & \cdots & k(u_N,v_N) \end{pmatrix}$$

$$K(x,y) = x^T y, x, y \in \mathbb{R}^d.$$

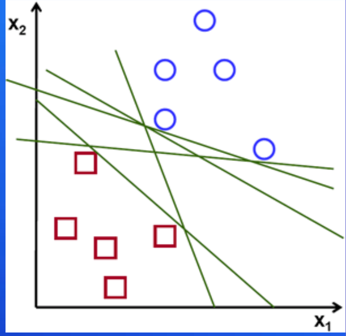
$$K(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}, x, y \in \mathbb{R}^d, \sigma > 0.$$

Mercer's theorem

- 对于正定核函数$K(s,t)$，存在正交特征函数集$\varphi_j(x)$和特征值集λ_i，使得

$$k(s,t) = \sum_{j}^{\infty} \lambda_i \varphi_j(s)\,\varphi_j(t)$$

SVM-Kernel Trick



- Dual Problem

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \vec{x}_i^{\,T} \vec{x}_j \qquad \alpha_i \geq 0$$

- Generalization

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K\left(\vec{x}_i, \vec{x}_j\right) \qquad \alpha_i \geq 0$$

The Kernel Trick

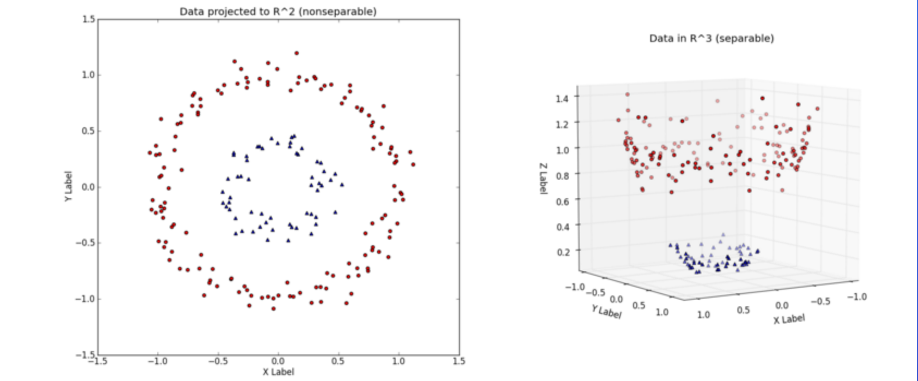


Figure 5: (Left) A dataset in $\mathbb{R}^2$, not linearly separable. (Right) The same dataset transformed by the transformation: $[x_1, x_2] = [x_1, x_2, {x_1}^2 + {x_2}^2]$.

- 给定数据，可以得到距离

$$d_{ij} = \langle \vec{x}_i, \vec{x}_j \rangle$$

- 对于复杂（如线性不可分）问题，通过变换(如升维）也许可以转化为简单（如线性可分）问题。
- Idea:
  - Take advantage of “Distance Dependency”
  - Transform data to “feature space”
  - Take distance measure in “feature space”

$$\varphi(\cdot): \qquad R^m \to R^k$$

$$k(\vec{x}, \vec{y}) = \langle \varphi(\vec{x}), \varphi(\vec{y}) \rangle_k$$

SVM-Kernel Trick

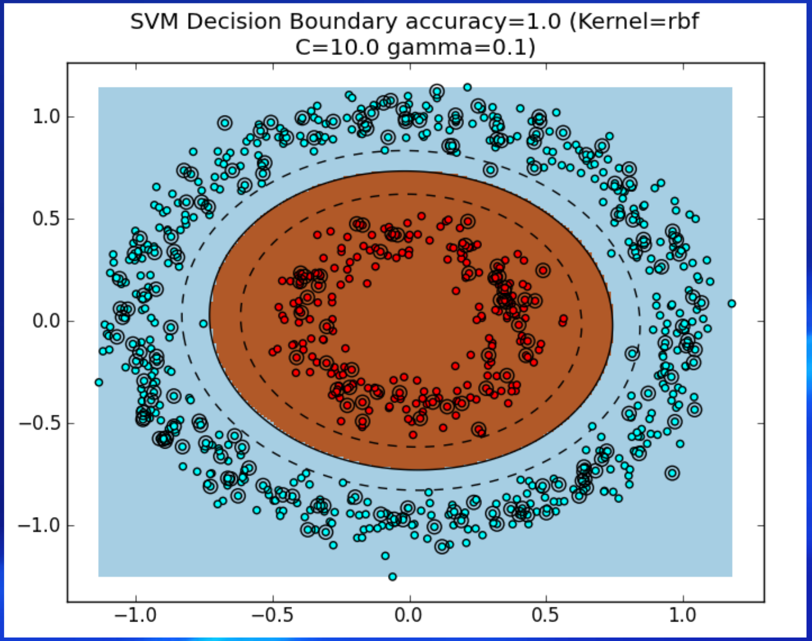
$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(\vec{x}_i, \vec{x}_j) \qquad \alpha_i \geq 0$$

Polynomial Kernel

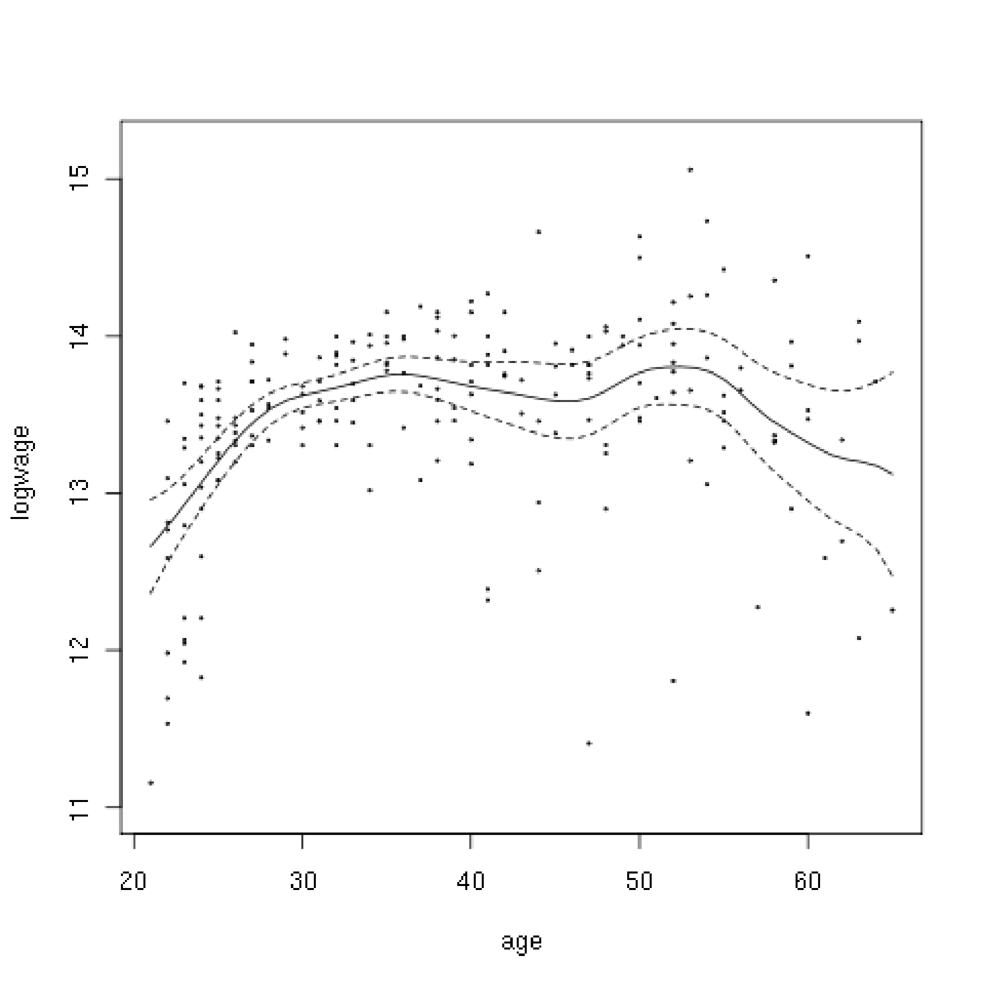
$$\mathrm{K}(\vec{x}_i, \vec{x}_j) = (-\gamma \langle \vec{x}_i, \vec{x}_j \rangle + \beta)^{d}$$

Radial Basis Function (RBF) Kernel

$$\mathrm{K}(\vec{x}_i, \vec{x}_j) = \exp(-\gamma |\vec{x}_i - \vec{x}_j|^2)$$



Kernel Regression -Nadaraya-Watson



$$f(s) = E(y|x = s)$$

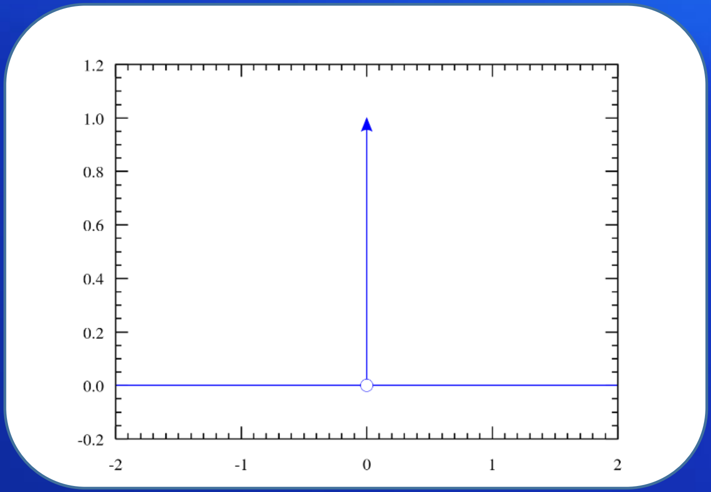
$$f(s) = \int y\, p(y|s) dy$$

$$f(s) = \int y \frac{p(y,s)}{p(s)} dy$$

$$\hat{f}(s) = \frac{\sum_{i=1}^{N} y_i K_h(s - x_i)}{\sum_{i=1}^{N} K_h(s - x_i)}$$

Dirac delta function, or δ function

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$



$$\int_{-\infty}^{+\infty} \delta(x)dx = 1$$

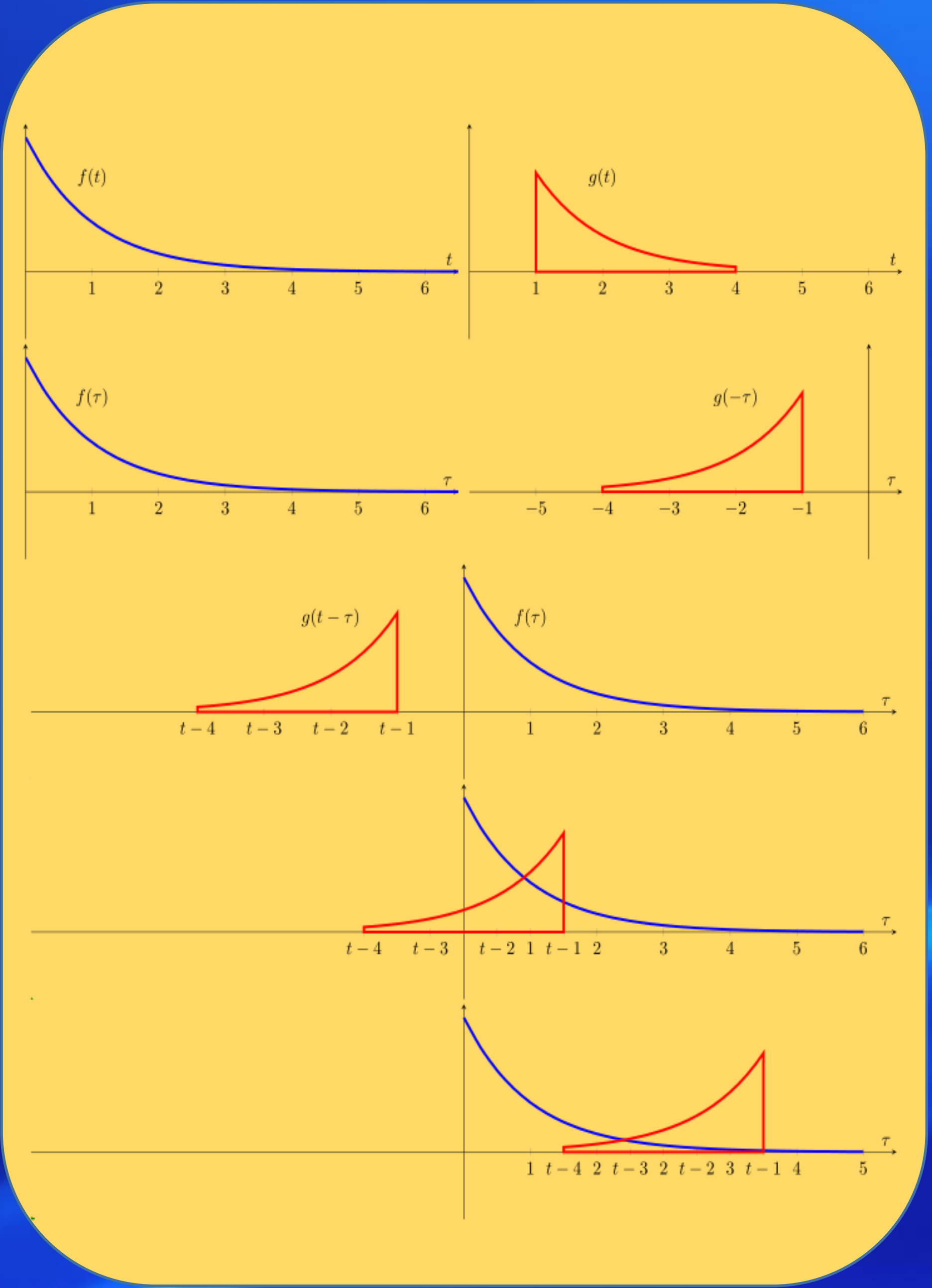
$$\int_{-\infty}^{+\infty} k(s)\delta(s-t)dx = k(t)$$

Convolution

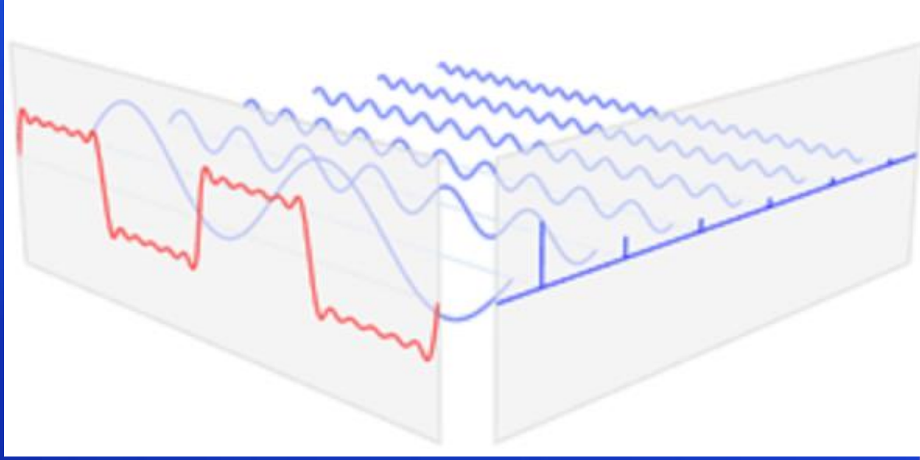
$$(f * g)(x) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} f(s)g(x-s)ds$$

$$= \int_{-\infty}^{+\infty} f(x-t)g(t)dt$$

$$\stackrel{\text{def}}{=} (g * f)(x)$$



Fourier Transform

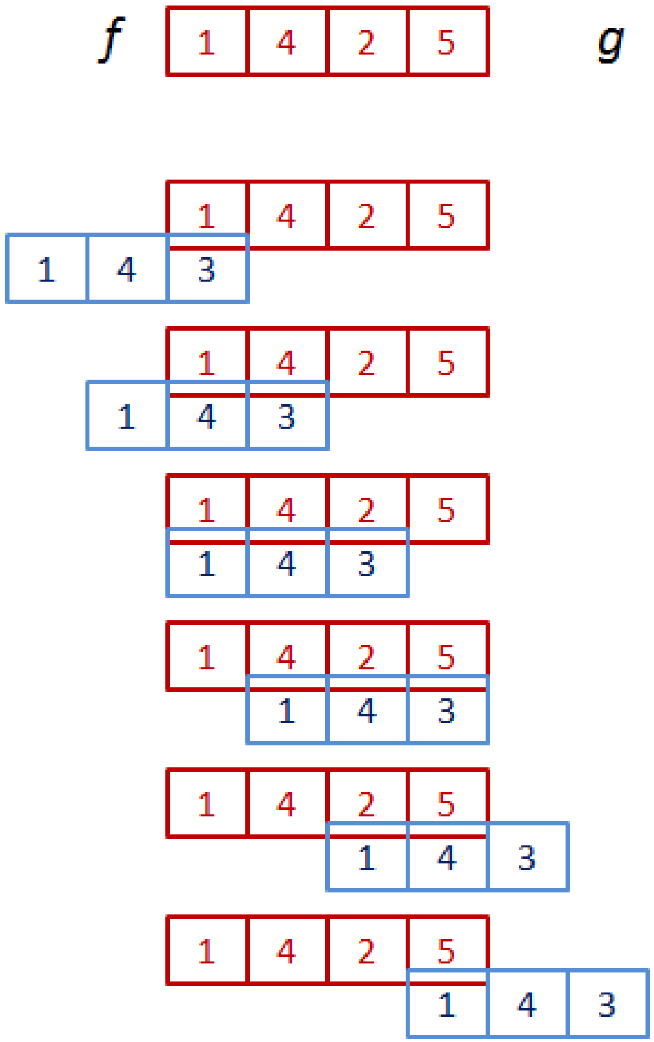


$$\mathcal{F}: f(t) \rightarrow \hat{f}(\xi) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} f(t) e^{-2\pi i t \xi} dt$$

$$\mathcal{F}\{f * g\}(t) = \mathcal{F}\{f\}(t) \cdot \mathcal{F}\{g\}(t)$$

离散卷积

$$(f * g)(n) \stackrel{\text{def}}{=} \sum_{i=-\infty}^{+\infty} f(i)g(n-i)$$



$$c = f * g$$

C[0] = 1*3 = **3**

C[1] = 1*4 + 4*3 = **16**

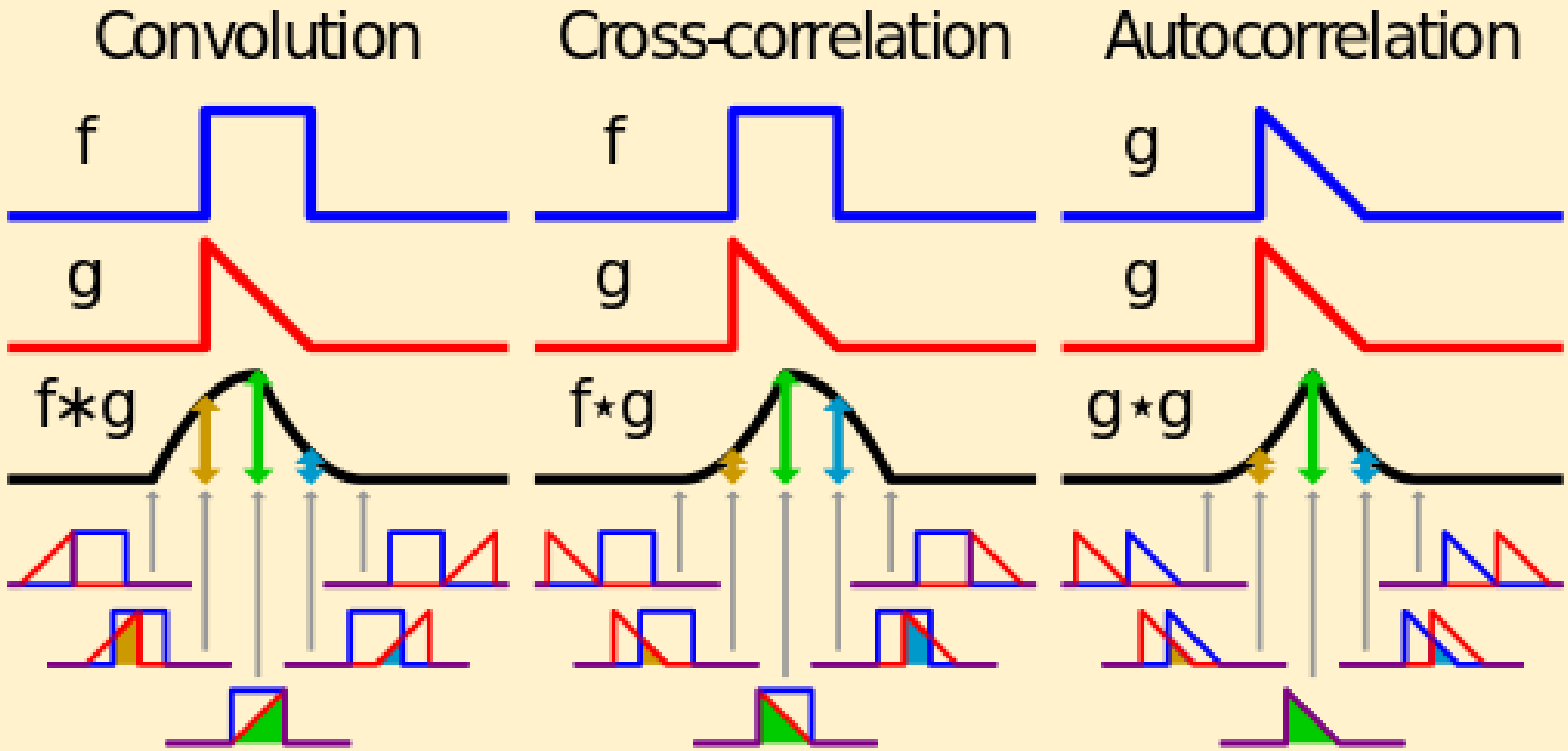
C[2] = 1*1 + 4*4 + 2*3 = **23**

C[3] = 4*1 + 2*4 + 5*3 = **27**

C[4] = 2*1 + 5*4 = **22**

C[5] = 5*1 = **5**

http://toto-share.com


Convolution
Cross-correlation
Autocorrelation
f
g
f∗g
f
g
f⋆g
g
g
g⋆g

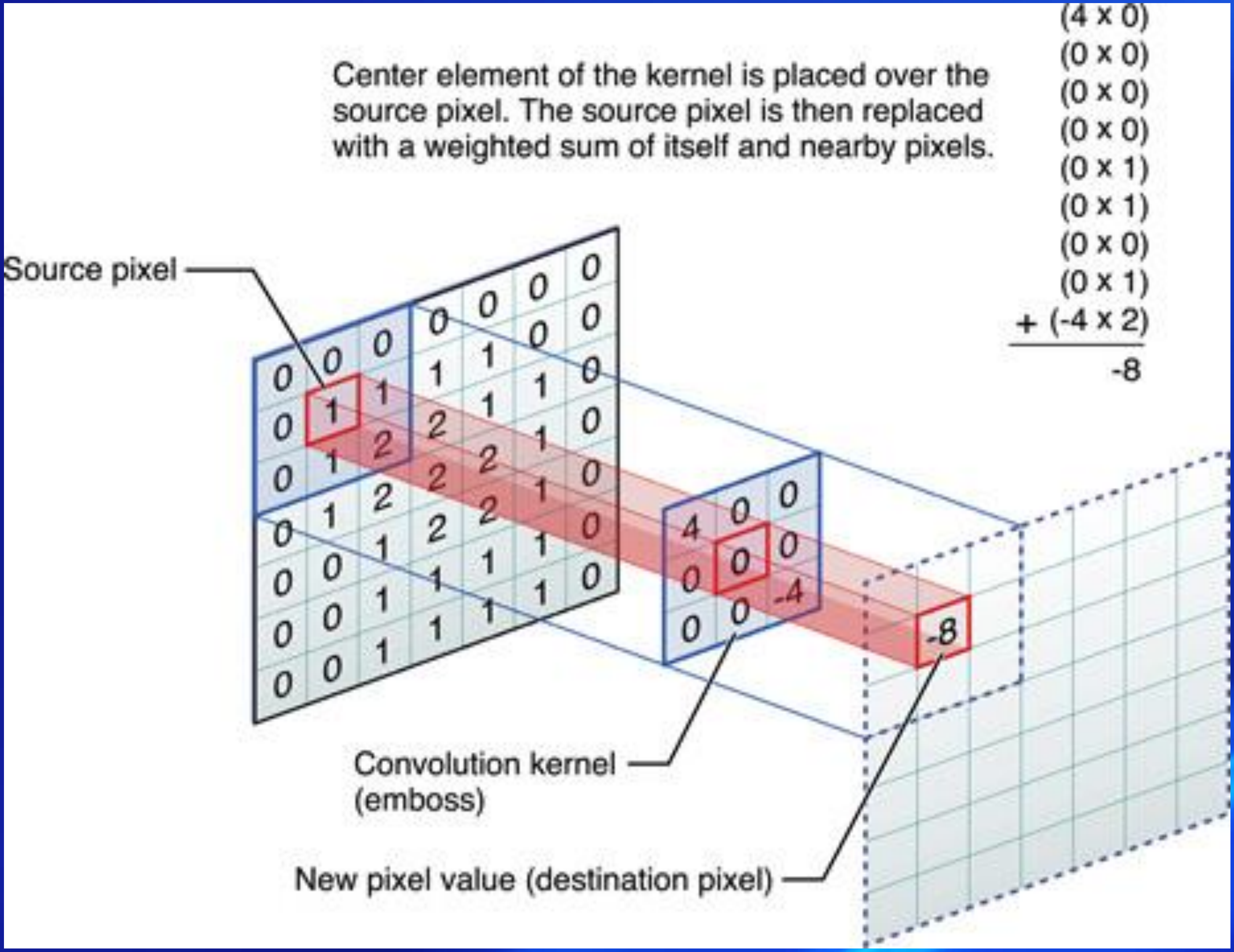
Convolution in 2D

- Continuous

$$(f * g)(x, y) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(s, t) g(x - s, y - t) ds dt$$

- Discrete

$$(f * g)(n, m) \stackrel{\text{def}}{=} \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f(i, j) g(n - i, m - j)$$


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Why Convolution?

- 1-D
  - Memory
  - Average
  - Differentiation
  - …
- 2D
  - Average
  - Differentiation
  - Direction
  - …

1	1	1

-1	0	1

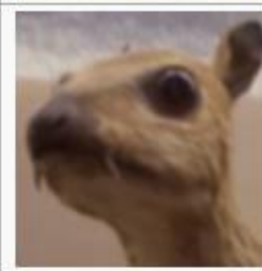
1	1	1
1	1	1
1	1	1

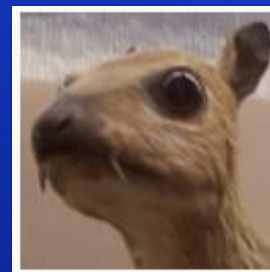
-1	0	1
-1	0	1
-1	0	1

0	0	1
0	1	0
1	0	0

1	0	0
0	1	0
0	0	1

Convolution for Image Processing

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	



Filter: Emboss

$$\ast \begin{array}{|c|c|c|}\hline -2 & -1 & 0 \\ \hline -1 & 1 & 1 \\ \hline 0 & 1 & 2 \\ \hline \end{array} =$$

(GIMP documentation)

Filter: Edge-Detect

$*$

0	-1	0
-1	4	-1
0	-1	0

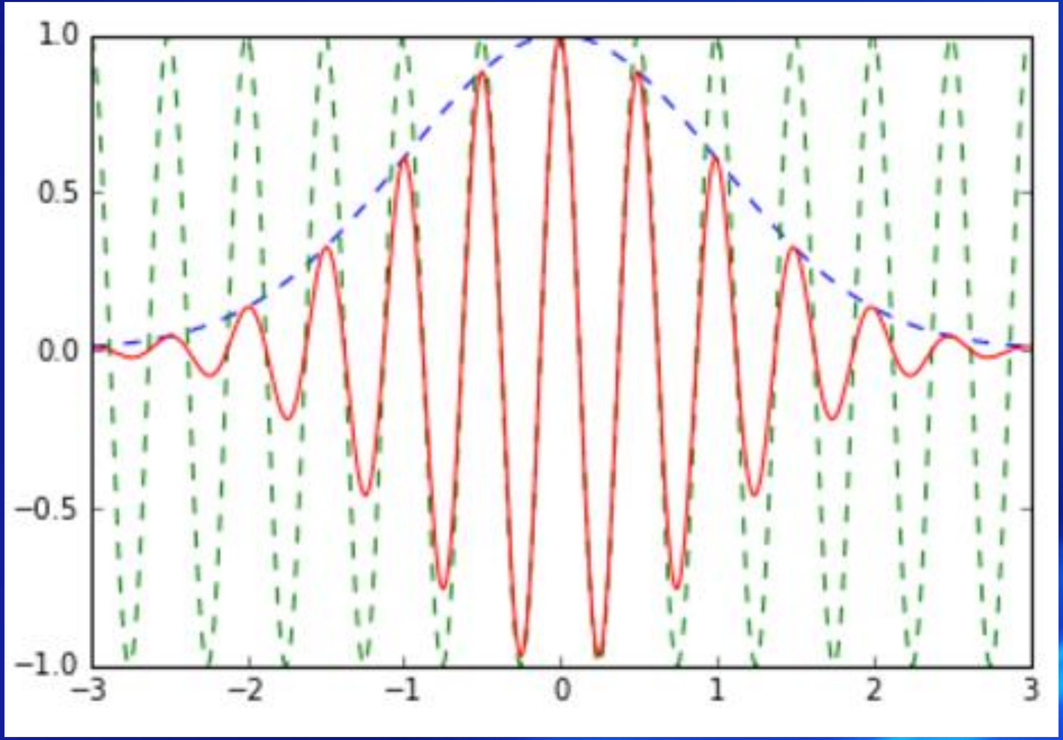
$=$

(GIMP documentation)

Canny edge detection

Gabor Filters – 1D

$$g(x;\sigma,f,\phi) = \underbrace{\exp\left(-\frac{x^2}{2\sigma^2}\right)}_{\text{Gaussian}}\underbrace{\exp\left(i\left(\frac{2\pi x}{f}+\phi\right)\right)}_{\text{Sinusoid}}$$



Gabor Filters – 2D

Complex

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right)$$

Real

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right)$$

Imaginary

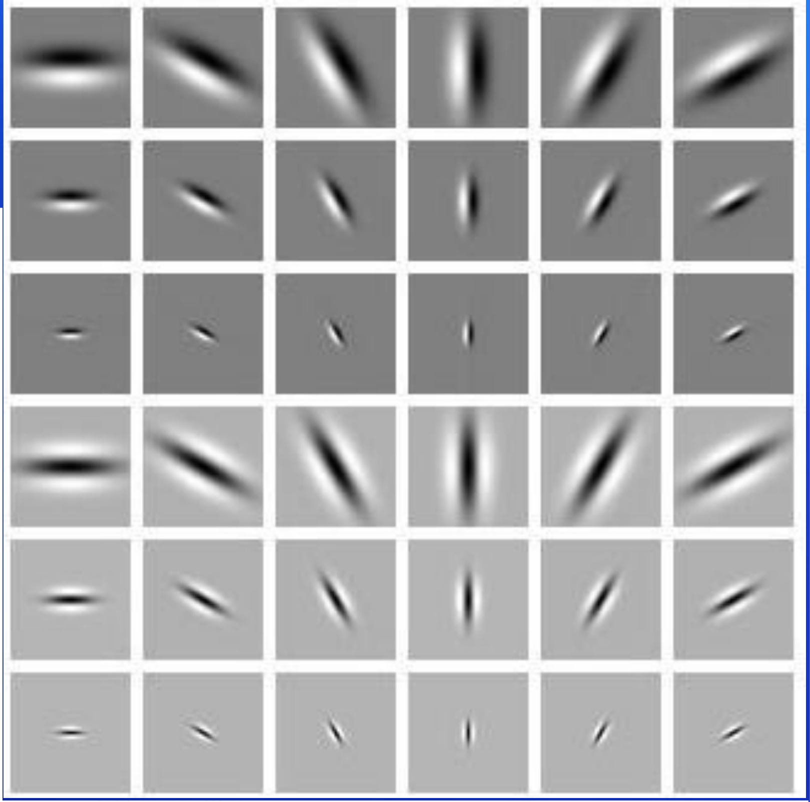
$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}\right)\sin\left(2\pi\frac{x'}{\lambda}+\psi\right)$$

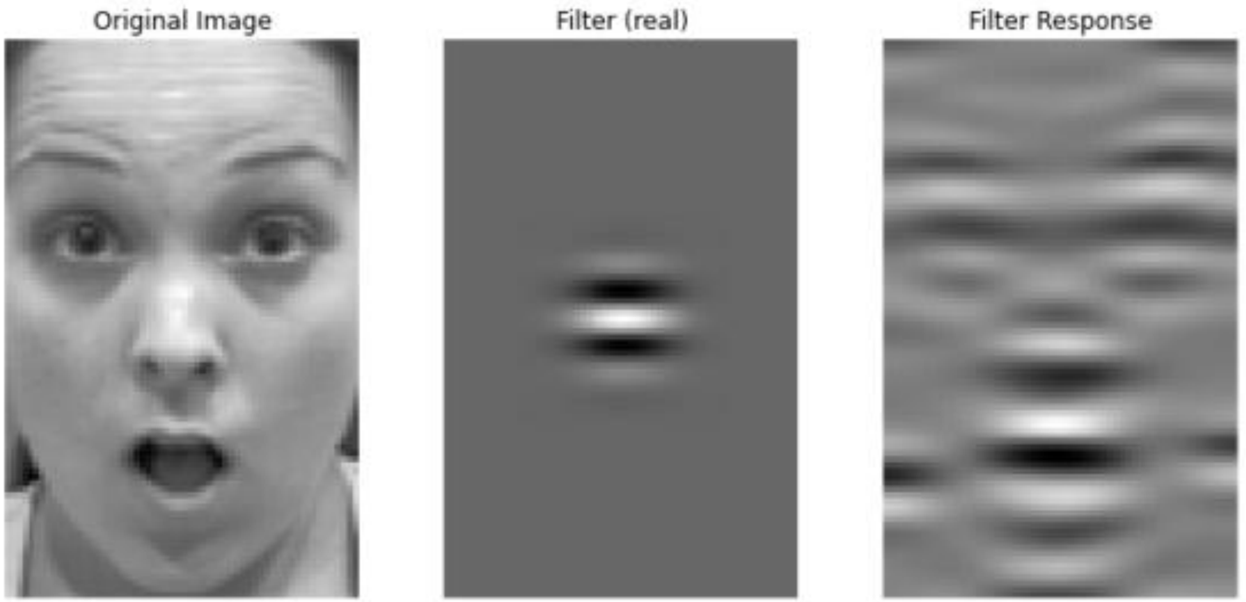
where

$$x' = x\cos\theta + y\sin\theta$$

and

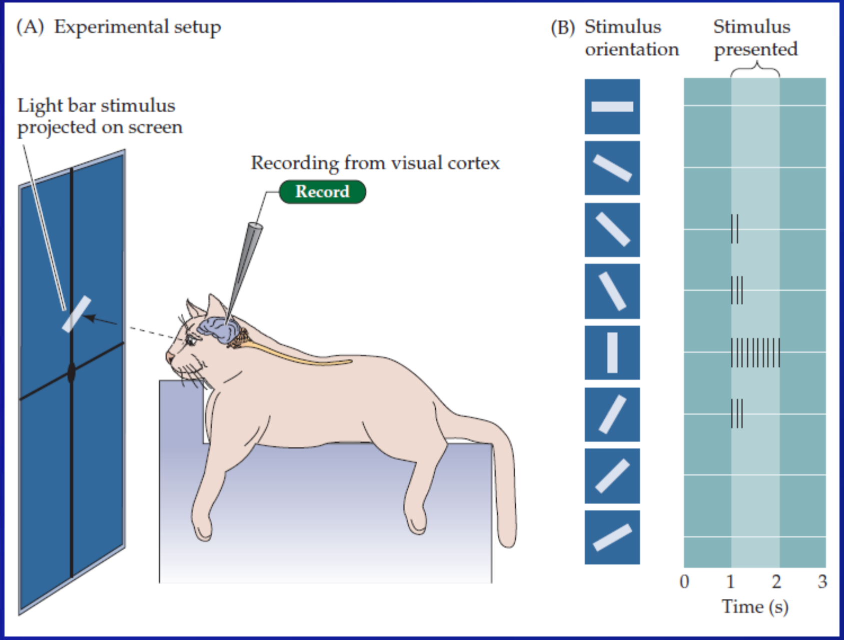
$$y' = -x\sin\theta + y\cos\theta$$

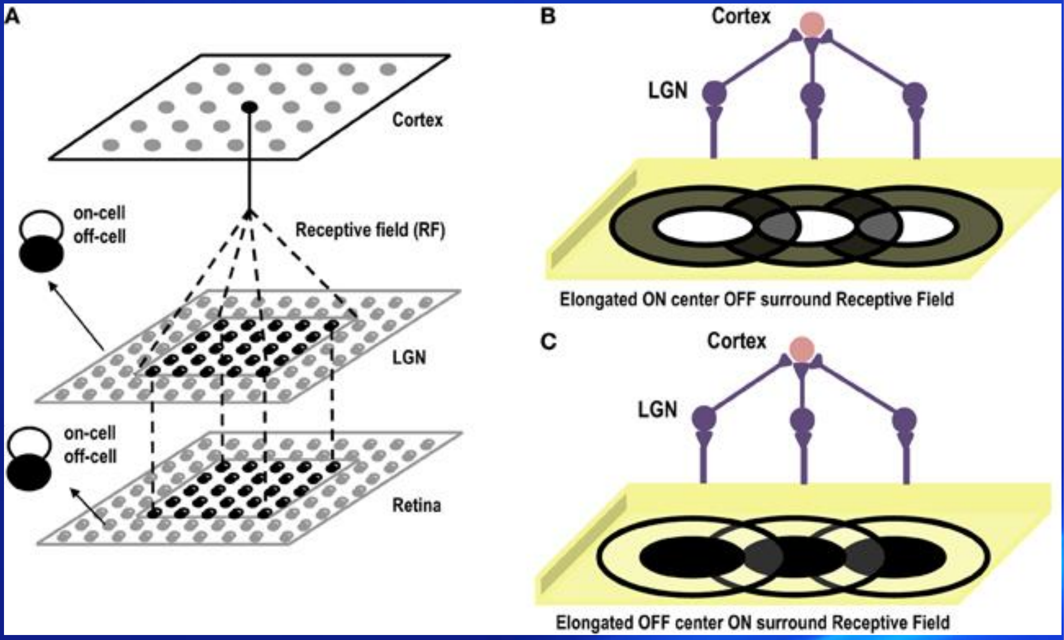




Convolution happens in brain

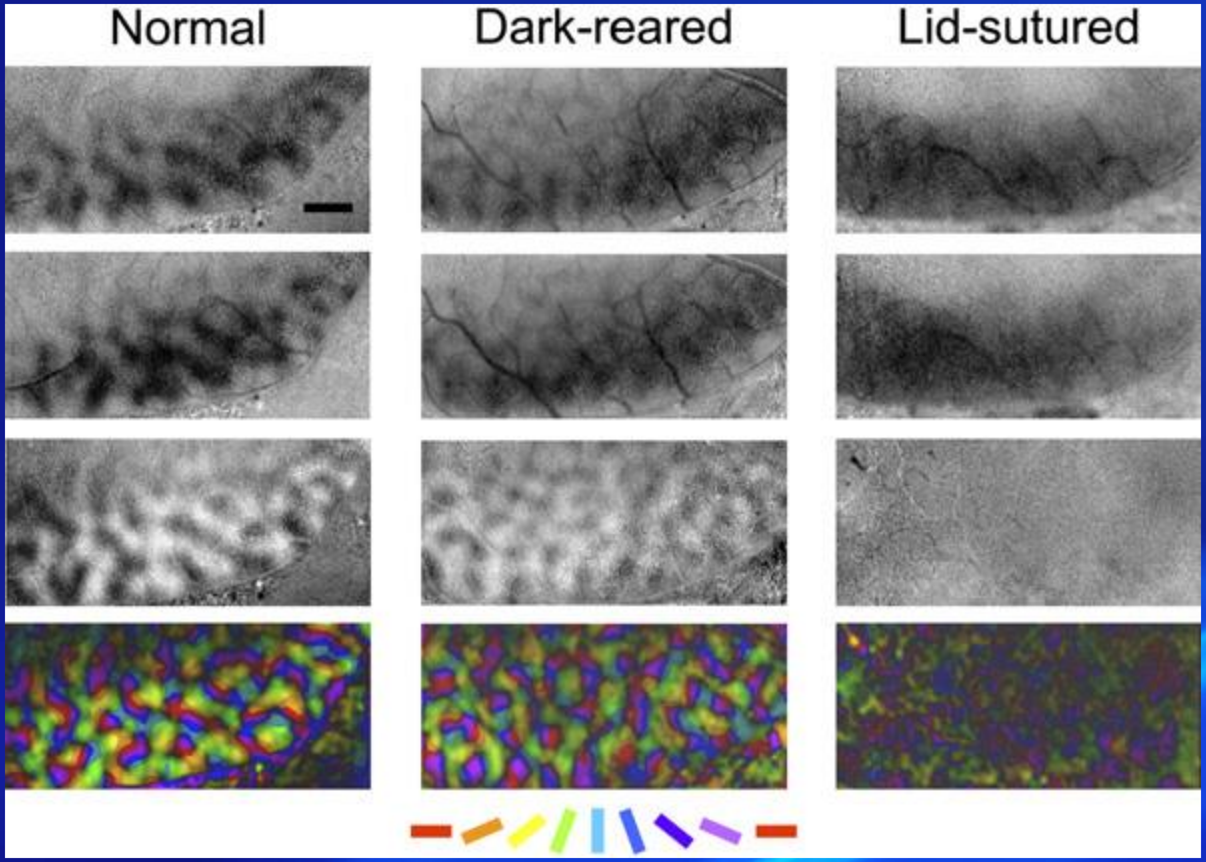
- Orientation Selectivity in V1
- Large cells and small cells – RF size





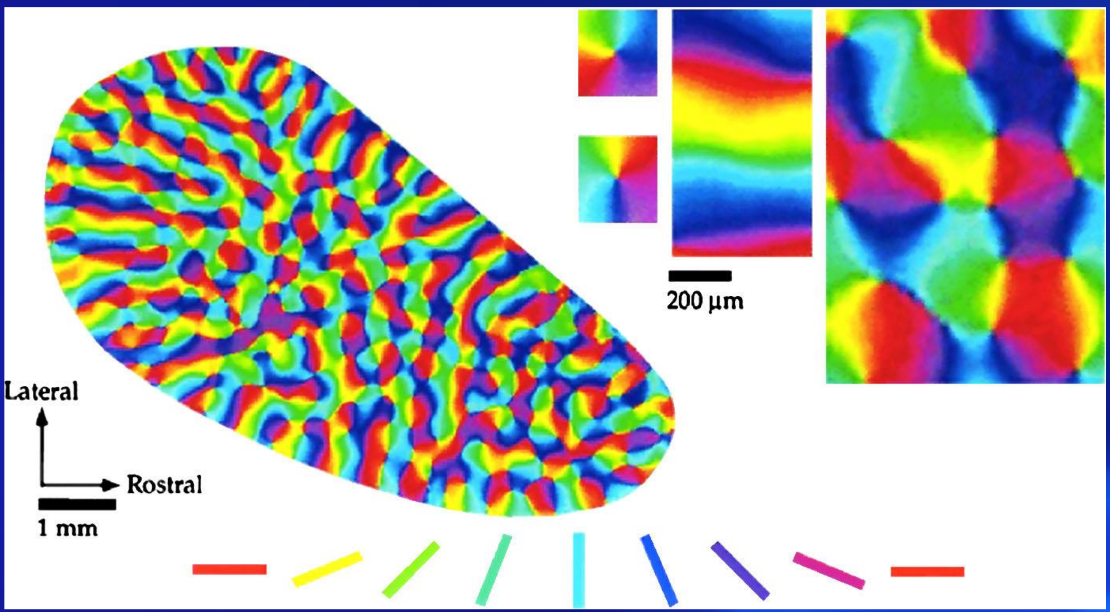
Convolution is auto-developed in Brain

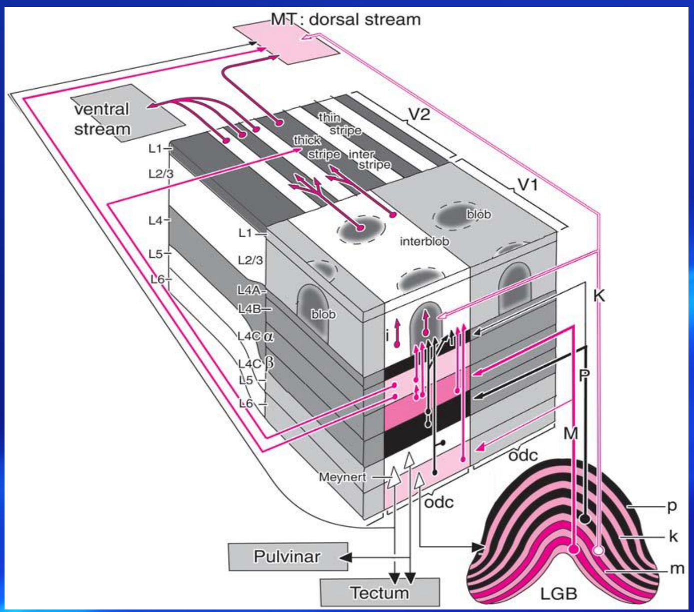
- Auto-developed under
  - Normal condition
  - With critical period



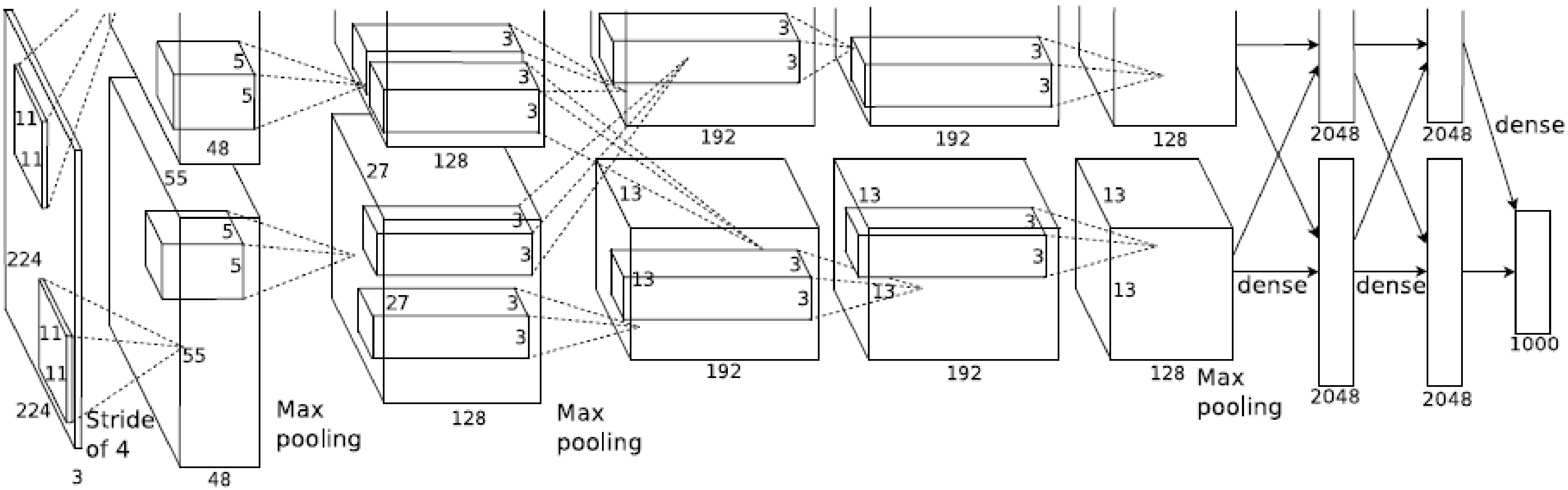
Neural Circuitry from LGN to V1

- How exactly the orientation map is developed?
- What’s the neural connection pattern from LGN to V1-Layer4-LayerX?
  - 从V1看，有Orientation Cells arranged in pinwheels. 但由于pinwheel的Size大约是0.5mm，整个V1能容纳的pinwheels数量有限，所以V1对方向的感受是有一定的区域，也就是感受野的大小。
  - V1中每个方向细胞的感受野相同吗？
  - 从V1细胞的反应能追溯LGN的细胞情况吗？是否可逆？





ConvNet or CNN - Basics



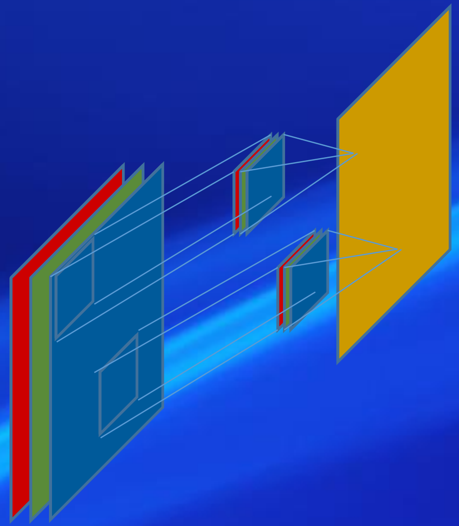
- Feed forward multi-layers
- A Layer can have depth – Anti biological intuition
- Convolve + Pooling
- FC towards the end

ConvNet for Image

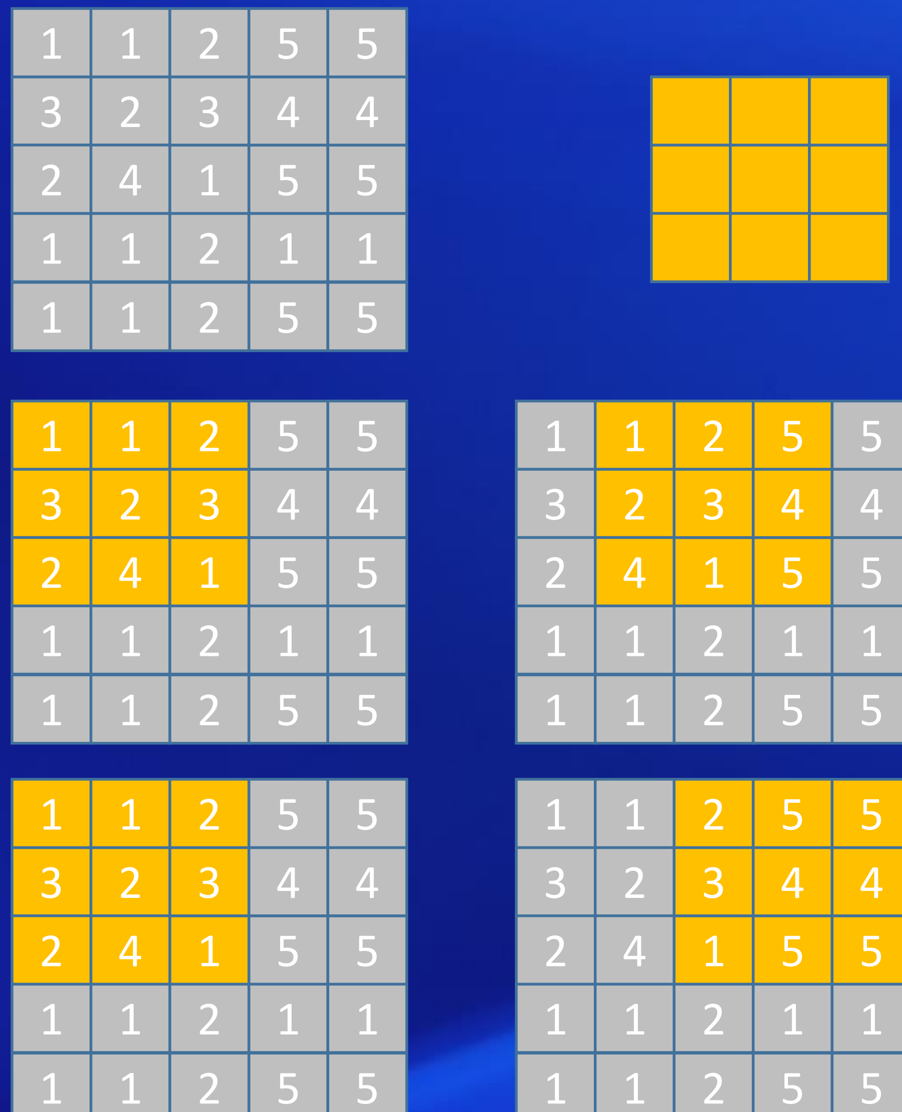
- Image RGB(i,j)-Input Layer depth=3
- Convolution filters
  - Depth of filters=Depth of Pre-Layer
  - # of filters is a hyperparameter
  - Depth of the Post-layer=# of filters

ConvNet-Filter Parameters

- Filter has its size as hyperparameter denoted by F
- The squared size of a filter is FxF
- # of filter’s parameters=WxHxD+1, D for Depth and 1 for bias.
- Every filter has it’s own parameters
- Filter’s weights are shared by all neurons in the filter’s Post-Layer.
- About weight sharing
  - Biological plausible
  - Saving a lot of weights
- Questions
  - Square?
  - Same size F – RF size?



ConvNet-Stride



- Stride=1
- Stride=2

ConvNet-Padding

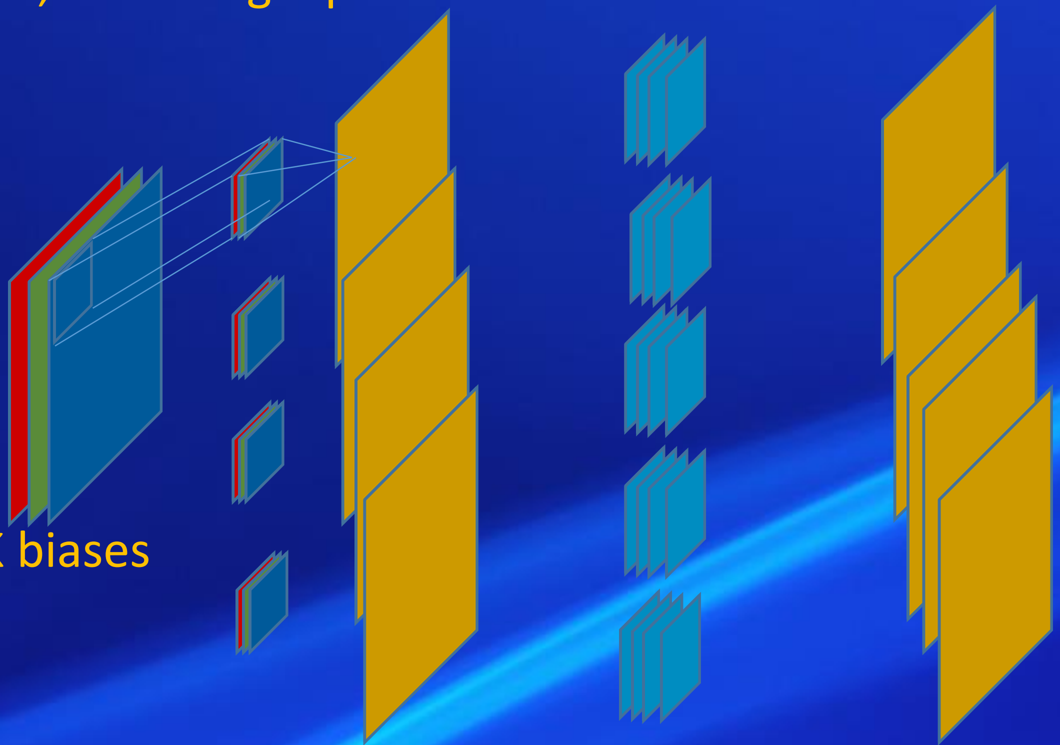
- Filter size can reduce the post-Layer's size
- Padding to prevent such reduction
- Padding size: (F-1)/2   F-Filter Size
- Zero: Best?

0	0	0	0	0	0	0
0	1	1	2	5	5	0
0	3	2	3	4	4	0
0	2	4	1	5	5	0
0	1	1	2	1	1	0
0	1	1	2	5	5	0
0	0	0	0	0	0	0

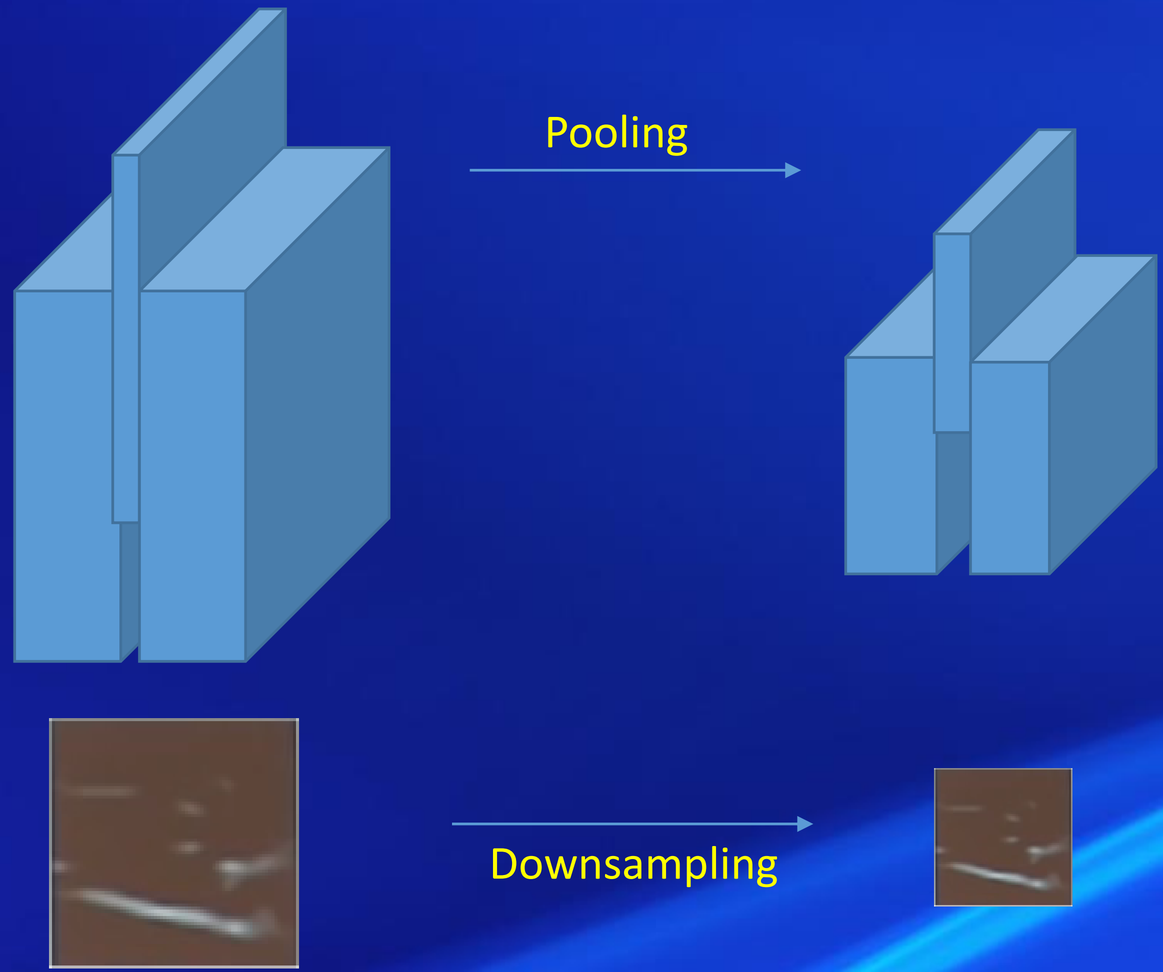
0	0	0	0	0	0	0
0	1	1	2	5	5	0
0	3	2	3	4	4	0
0	2	4	1	5	5	0
0	1	1	2	1	1	0
0	1	1	2	5	5	0
0	0	0	0	0	0	0

ConvNet - Build PostLayer

- Input Layer
  - W1, H1, D1
- Hyperparameters
  - # of filters K – Power of 2
  - Filter size F – Odd number; Assuming Square Filter
  - Stride size S
  - Pad Size P
- ConvLayer
  - W2=(W1-F+2P)/S+1
  - H2=(H1-F+2P)/S+1
  - D2=K
- Parameters
  - FxFxD1 for one filter
  - FxFxD1xK for K filters + K biases



ConvNet - Pooling



ConvNet - Max Pooling

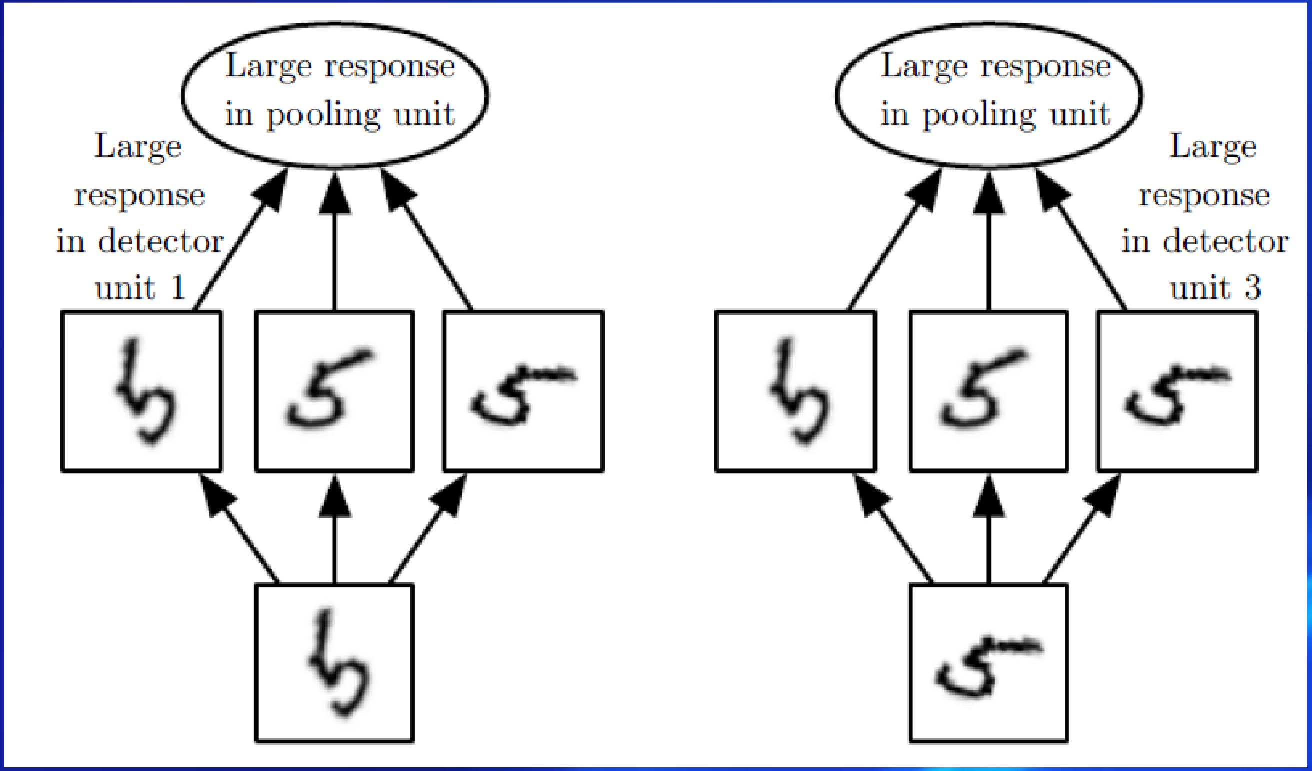
- Input Layer
  - W1, H1, D1
- Hyperparameters
  - Pooling size F
  - Stride size S
- PoolingLayer
  - W2=(W1-F)/S+1
  - H2=(H1-F)/S+1
  - D2=D1
- Parameters
  - None
- Other Pooling
  - Average – Not common
  - BP-derivative calculation

1	1	2	5
3	2	3	4
2	4	1	5
1	1	2	1

3	5
4	5

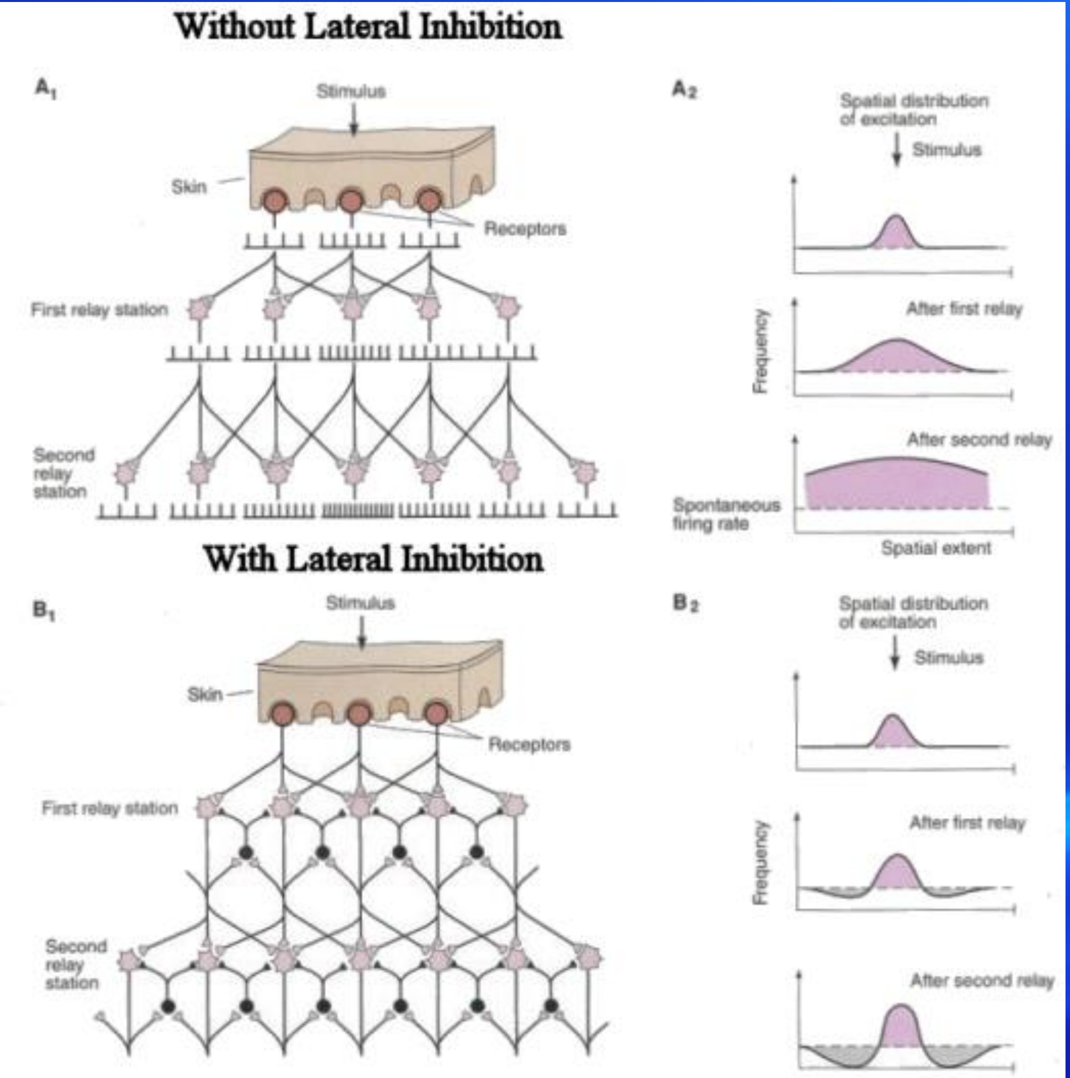
Why Pooling?

- Reduce space
- Get invariance – reducing noise



Why Pooling?

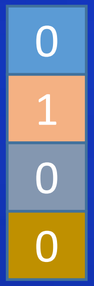
- Biological reason
  - Lateral Inhibition
  - WTA



Adapted from Kandel, E. R., Schwartz, J. H. , et al. (1995).
Essentials of neural science and behavior. Stamford, CT, Appleton & Lange.

Final Fully Connected Layer

- FC Layer
  - # of neurons=# of classes
  - One-hot encoding

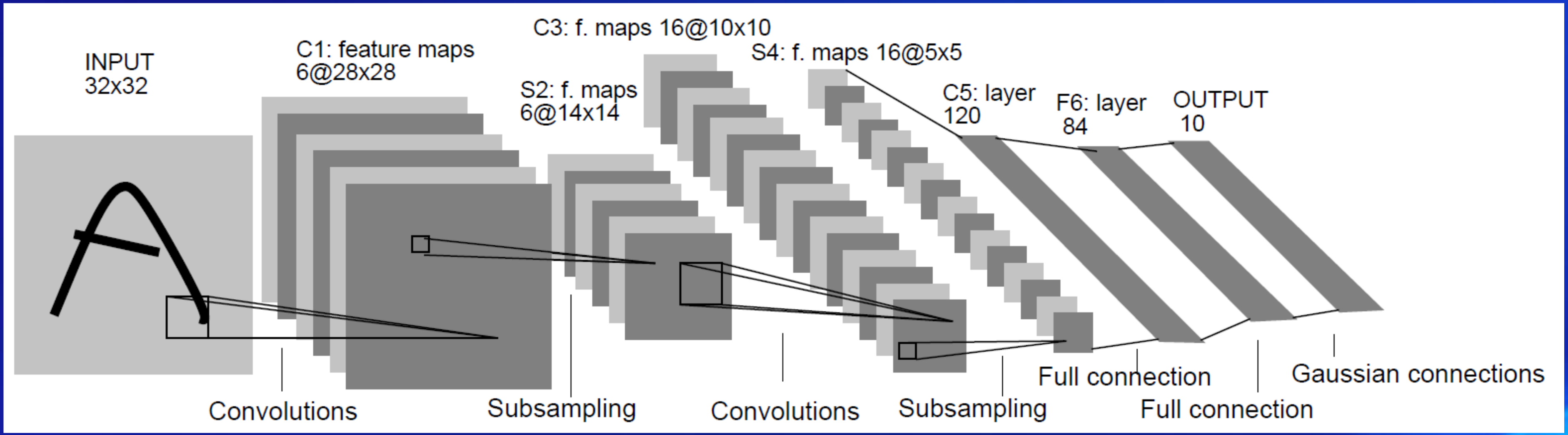


Training CNN

- BP method
- Different from MLP BP
  - Pooling
  - Parameter sharing
  - Auto derivative?
- Refs:
  - http://www.cnblogs.com/tornadomeet/p/3468450.html

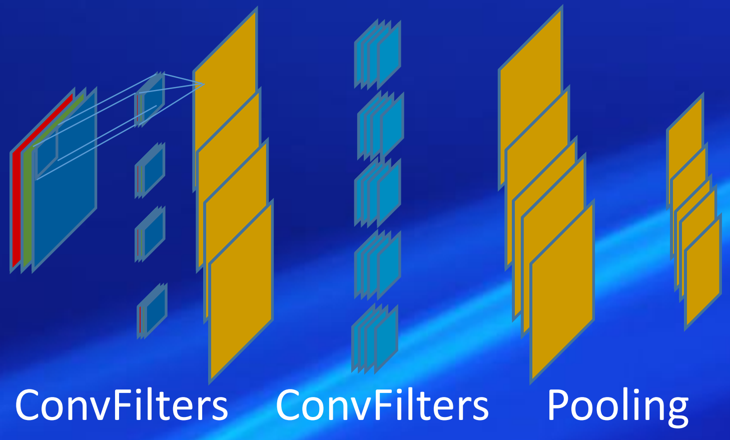
LeNet-5

- LeCun 1998
- 10 Classes



总结

- Kernel Method
  - Sample point can be viewed as a kernel convolved with delta-function on that point
    - Density estimation
    - Regression
  - Kernel can be viewed as dot product in Hilbert space
    - Kernel trick for SVM
- Convolution
  - Kernel as filter
    - Signal processing – Fourier Transformation
    - Image processing
    - Gabor filter
  - Convolution in brain
    - Orientation cell in V1
  - ConvNet
    - Layer with depth
    - Filter with depth (=PreLayer Depth)
    - # of filters (=Post-Layer depth)
    - Filter size – Stride – Padding
    - Max pooling: Pooling size - stride



Homework

- Theory
  - 证明 两个函数的卷积的F变换等于两个函数的F变换后的乘积
- Practice
  - 采用中传男女学生身高体重信息
    - 用Kernel方法分别得到男女生的身高体重1D分布密度函数估计
    - 用Kernel方法分别得到男女生的身高体重2D分布密度函数估计
    - 用Kernel方法分别得到男女生的体重y vs 身高x 的回归曲线
    - 用Kernel Trick方法依据身高体重数据做SVM分类
    - 以上所有结果请形成分析报告
  - 采用HTML5+JavaScript实现一个展示Gabor filter的交互网页
  - 采用MNST数据，建立一个ConvNet做分类训练学习，并形成分析报告
    - 建议参考LeNet-5的构架
    - Ref 1998，LeCun等，Gradient-based learning applied to document recognition