

浙大机器学习课程: 14, 16, 24 (83: 00), 28, 30 (10: 27),
35 (12: 00), 37, 38, 39 (60: 00), 41

机器学习笔记:

人工神经网络网络训练:

训练建议

(1) 一般情况下，在训练集上的目标函数的平均值 (cost) 会随着训练的深入而不断减小，如果这个指标有增大情况，停下来。有两种情况：第一是采用的模型不够复杂，以至于不能在训练集上完全拟合；第二是已经训练很好了。

(2) 分出一些验证集 (Validation Set)，训练的本质目标是在验证集上获取最大的识别率。因此训练一段时间后，必须在验证集上测试识别率，保存使验证集上识别率最大的模型参数，作为最后结果。

(3) 注意调整学习率 (Learning Rate)，如果刚训练几步 cost 就增加，一般来说是学习率太高了；如果每次 cost 变化很小，说明学习率太低。

激活 Windows
转到“设置”以激活 Windows。

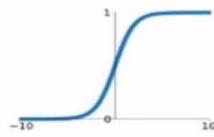
参数设置

1. 随机梯度下降
(Stochastic Gradient Descent, SGD)
2. 激活函数选择
3. 训练数据初始化
4. (W , b) 的初始化
5. Batch normalization
6. 目标函数选择
7. 参数更新策略
8. 训练建议

激活函数

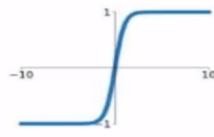
Sigmoid

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



tanh

$$\tanh(x)$$



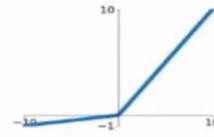
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



激活 Windows
转到“设置”以激活 Windows。



Batch Normalization

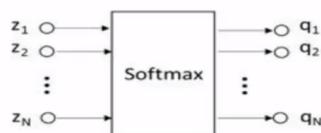
论文：Batch normalization accelerating deep network training by reducing internal covariate shift (2015)

基本思想：既然我们希望每一层获得的值都在0附近，从而避免梯度消失现象，那么我们为什么不直接把每一层的值做基于均值和方差的归一化呢？

目标函数选择

2. 如果是分类问题， $F(W)$ 可以采用SOFTMAX函数和交叉熵的组合。

(a) SOFTMAX函数：



$$q_i = \frac{\exp(z_i)}{\sum_{j=1}^N \exp(z_j)}$$

我们希望通过此网络学习由 $Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}$ 到 $P = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}$ 的映射。其中 $\sum_{i=1}^N p_i = 1$

(b) 交叉熵 (Cross Entropy) :

$$\text{定义目标函数为 } E = - \sum_{i=1}^N p_i * \log(q_i)$$

激活 Windows
转到“设置”以激活 Windows。

参数更新策略

SGD的问题

(1) (W, b) 的每一个分量获得的梯度绝对值有大有小，一些情况下，将会迫使优化路径变成Z字形状。

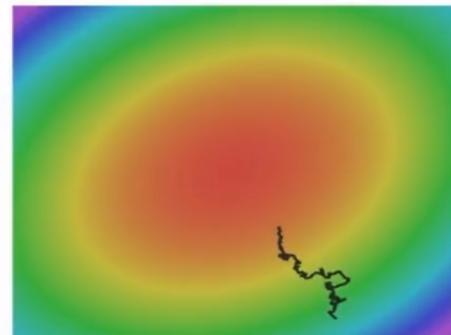


参数更新策略

SGD的问题

(2) SGD求梯度的策略过于随机，由于上一次和下一次用的是完全不同的BATCH数据，将会出现优化的方向随机的情况。

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



参数更新策略

同时两个问题：

(4) Adam

Algorithm 8.7 The Adam algorithm

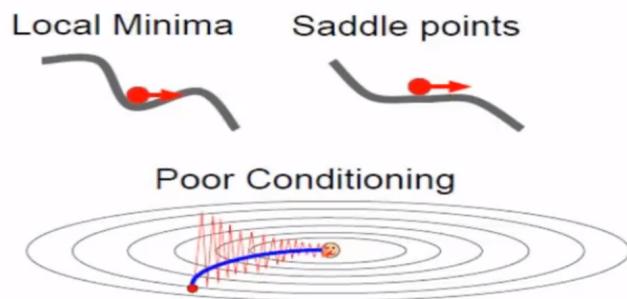
Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1]$.
(Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default:
 10^{-8})
Require: Initial parameters θ
Initialize 1st and 2nd moment variables $s = \mathbf{0}$, $r = \mathbf{0}$
Initialize time step $t = 0$
while stopping criterion not met **do**
 Sample a minibatch of m examples from the training set $\{x^{(1)}, \dots, x^{(m)}\}$ with
 corresponding targets $y^{(i)}$.
 Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
 $t \leftarrow t + 1$
 Update biased first moment estimate: $s \leftarrow \rho_1 s + (1 - \rho_1) \mathbf{g}$
 Update biased second moment estimate: $r \leftarrow \rho_2 r + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}$
 Correct bias in first moment: $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$
 Correct bias in second moment: $\hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$
 Compute update: $\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r} + \delta}}$ (operations applied element-wise)
 Apply update: $\theta \leftarrow \theta + \Delta\theta$
end while

激活 Windows
转到“设置”以激活 Windows,

参数更新策略

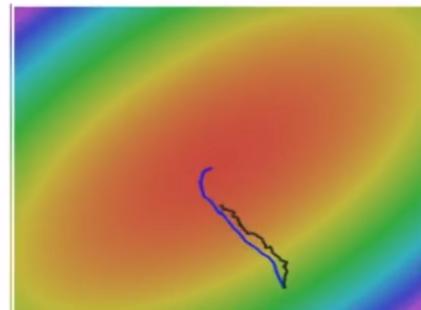
解决梯度随机性问题：

(3) Momentum



SGD+Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$



训练建议

(4) Batch Normalization 比较好用，用了这个后，对学习率、参数更新策略等不敏感。建议如果用Batch Normalization，更新策略用最简单的SGD即可，我的经验是加上其他反而不好。

(5) 如果不用Batch Normalization，我的经验是，合理变换其他参数组合，也可以达到目的。

(6) 由于梯度累积效应，AdaGrad, RMSProp, Adam三种更新策略到了训练的后期会很慢，可以采用提高学习率的策略来补偿这一效应。

深度学习介绍

1. 数据库介绍
2. 自编码器 (Auto encoder)
3. 卷积神经网络 (Convolutional Neural Networks, CNN)
4. 深度学习工具 (Tensorflow和Caffe)
5. 流行的卷积神经网络结构 (LeNet, AlexNet, VGGNet, GoogLeNet, ResNet)
6. 我们实验室的工作

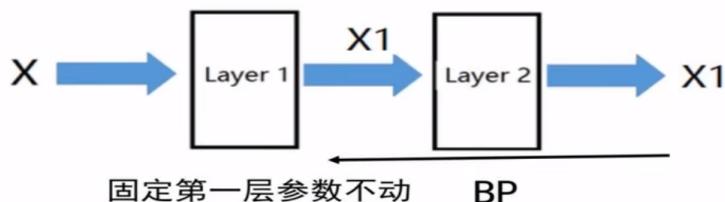
深度学习：

数据库：手写数字识别，ImageNet

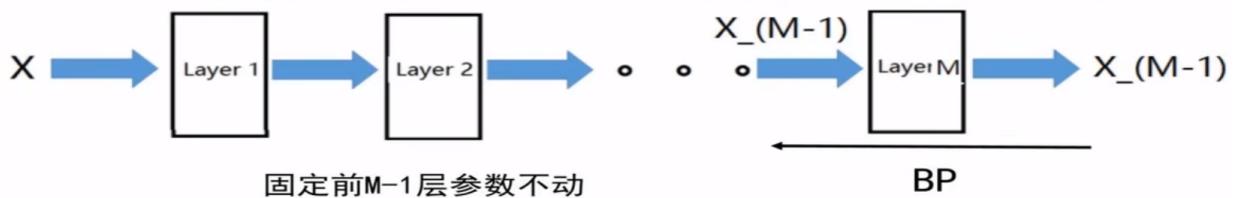
自编码器

自编码器 (Auto-encoder)

步骤2：训练好第1层后，接着训练第二层：

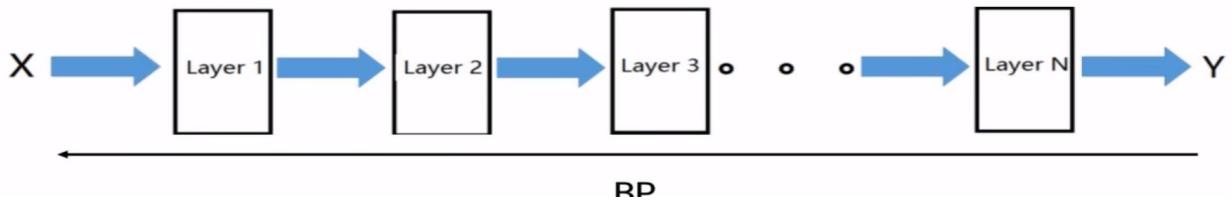


步骤M：以此类推，训练好第M-1层后，接着训练第M层。



自编码器 (Auto-encoder)

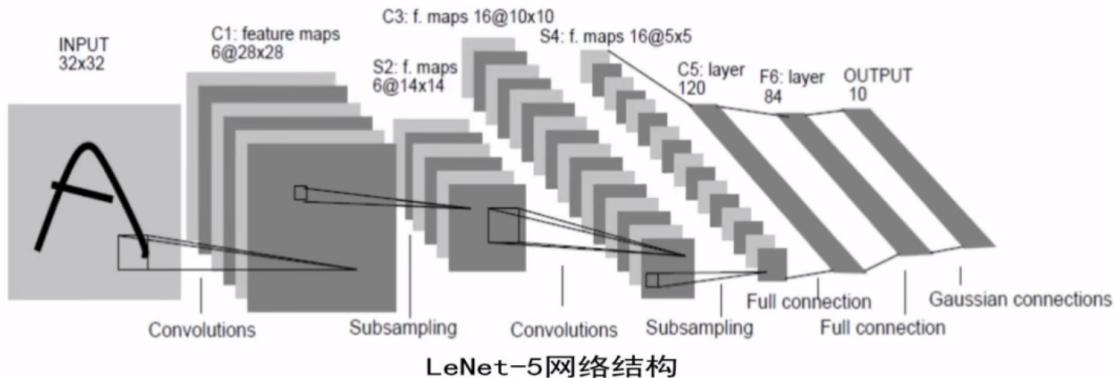
最后用BP对网络进行微调



深度学习 - 卷积神经网络



卷积神经网络 (Convolutional Neural Network, CNN) 由 LeCun 在上世纪90年代提出。



LeNet-5 网络结构

1. LeCun Y., Bottou L., Bengio Y., and Haffner P., Gradient-based learning applied to document recognition, Proceedings of the IEEE, pp. 1-7, 1998.

CNN的特征图计算公式：

$$\begin{aligned} \text{点 1: } | &\sim m \\ \text{点 2: } (|+u) &\sim m+u \\ &\vdots \\ \text{点 } K: |+(K-1)u &\sim m+(K-1)u \\ &\leq M \\ K &\leq \frac{M-m}{u} + 1 \end{aligned}$$

点1: $| \sim m$

点2: $(|+u) \sim m+u$

:

,

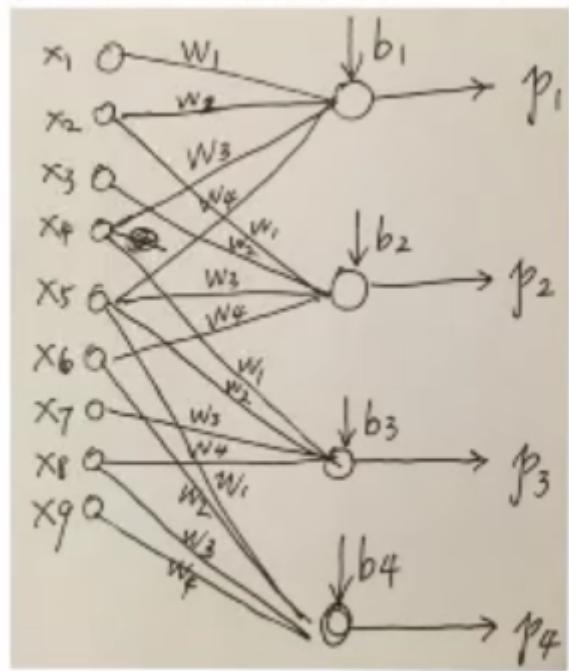
点K: $|+(K-1)u \sim m+(K-1)u \leq M$

$$K \leq \frac{M-m}{u} + 1$$

$$L \leq \frac{N-n}{v} + 1$$

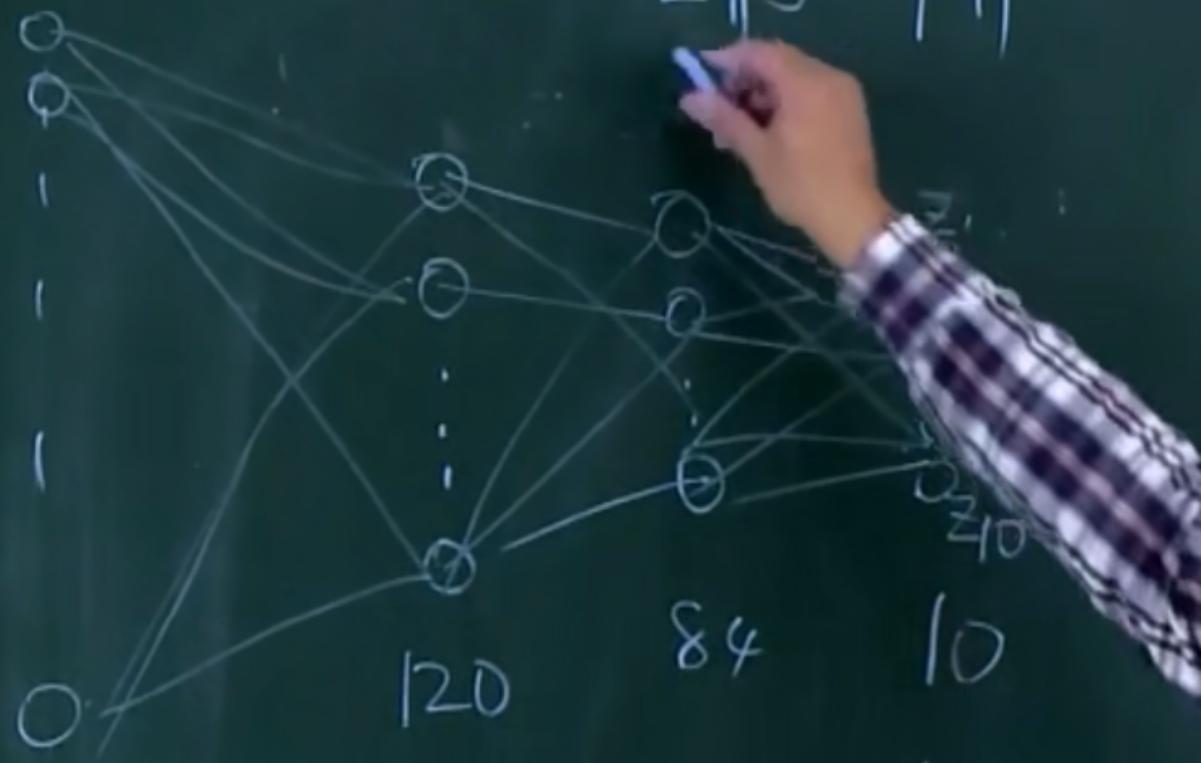
权重共享:

卷积神经网络，用于图像识别的卷积神经网络：



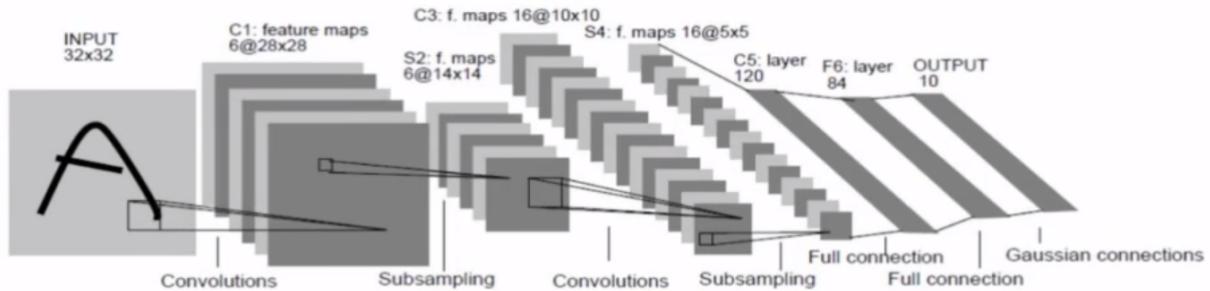
交叉熵计算公式：

$$E = \frac{1}{2} \|y - Y\|^2$$



$$\text{Softmax}(\mathbf{z}) = p$$

$$E = - \sum_{i=1}^{10} y_i \log(p_i)$$



参数个数计算：

第1层(convolutional layer): $(5*5+1)*6=156$

第2层(subsampling layer): 0

第3层(convolutional layer): $(5*5*6+1)*16=2416$

第4层(subsampling layer): 0

第5层(fully connected layer): $(5*5*16+1)*120=48120$

第6层(fully connected layer): $(120+1)*84=10164$

第7层(fully connected layer): $(84+1)*10=850$

激活
转至

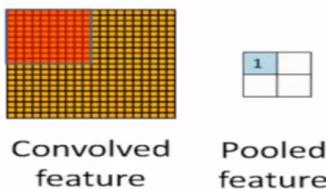
参数总数: 61,706

网络：

LeNet, Alexnet

2013 AlexNet 改进

(2) 为降采样操作起了一个新的名字—池化（Pooling），意思是把邻近的像素作为一个“池子”来重新考虑。如图3.31所示，左边所有红色的像素值可以看做是一个“池子”，经过池化操作后，变成右边的一个蓝色像素。

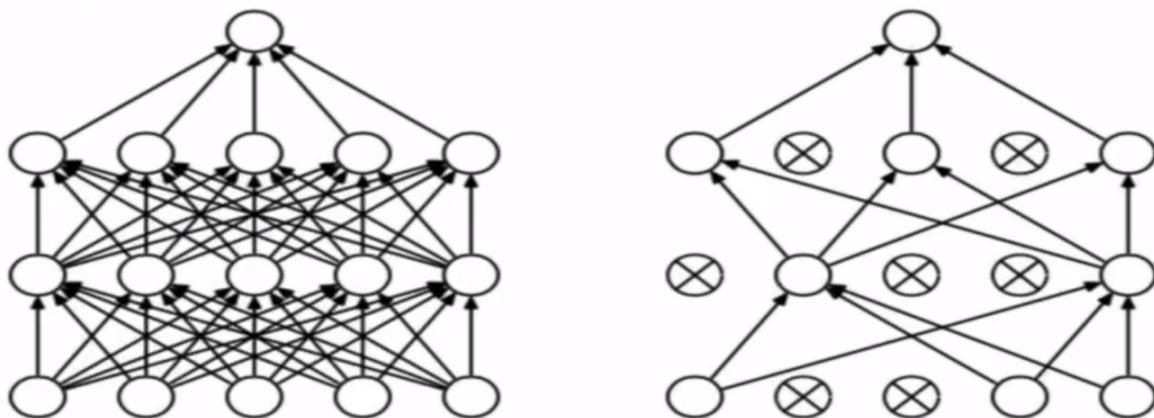


在AlexNet中，提出了最大池化(Max Pooling)的概念，即对每一个邻近像素组成的“池子”，选取像素最大值作为输出。在LeNet中，池化的像素是不重叠的；而在AlexNet中进行的是有重叠的池化。实践表明，有重叠的最大池化能够很好地克服过拟合问题，提升系统性能。

2013 AlexNet 改进

(3) 随机丢弃(Dropout)。为了避免系统参数更新过快导致过拟合，每次利用训练样本更新参数时候，随机的“丢弃”一定比例的神经元，被丢弃的神经元将不参加训练过程，输入和输出该神经元的权重系数也不做更新。这样每次训练时，训练的网络架构都不一样，而这些不同的网络架构却分享共同的权重系数。实验表明，随机丢弃技术减缓了网络收敛速度，也以大概率避免了过拟合的发生。

2013 AlexNet 改进



Dropout做法是，对每一层，每次训练时以概率 p 丢弃一些神经元，这样每次训练的网络都不一样。

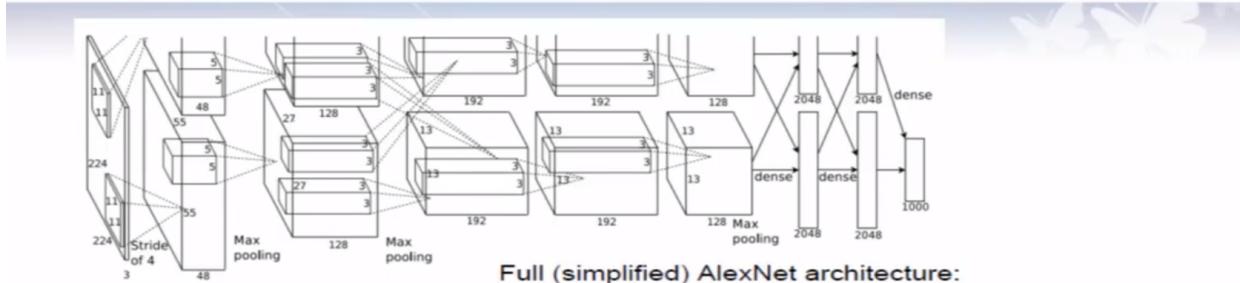
训练结束后的测试流程，要用完整的网络结构，同时对该层的所有参数(W, b)都要乘以 $(1-p)$ 。

2/2

2013 AlexNet 改进

(4) 增加训练样本。尽管ImageNet的训练样本数量有超过120万幅图片，但相对于6亿待估计参数来说，训练图像仍然不够。Alex等人采用了多种方法增加训练样本，包括：1. 将原图水平翻转；2. 将 256×256 的图像随机选取 224×224 的片段作为输入图像。运用上面两种方法的组合可以将一幅图像变为2048幅图像。还可以对每幅图片引入一定的噪声，构成新的图像。这样做可以较大规模增加训练样本，避免由于训练样本不够造成的性能损失

(5) 用GPU加速训练过程。采用2片GTX 580 GPU对训练过程进行加速，由于GPU强大的并行计算能力，使得训练过程的时间缩短数十倍，哪怕这样，训练时间仍然用了六天。



思考题：请计算一下
ALEXNET的参数个数

Full (simplified) AlexNet architecture:

[227x227x3] INPUT
 [55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
 [27x27x96] MAX POOL1: 3x3 filters at stride 2
 [27x27x96] NORM1: Normalization layer
 [27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
 [13x13x256] MAX POOL2: 3x3 filters at stride 2
 [13x13x256] NORM2: Normalization layer
 [13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
 [13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
 [13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
 [6x6x256] MAX POOL3: 3x3 filters at stride 2
 [4096] FC6: 4096 neurons 激活 Windows
 [4096] FC7: 4096 neurons 转到“设置”以激活 Windows.
 [1000] FC8: 1000 neurons (class scores)

利用两个3*3的卷积核代替一个7*7的卷积核：

VGGnet-16

VGGNet: (Simonyan and Zisserman, 2014)

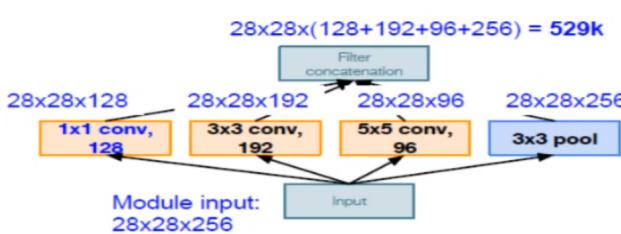


问题：为何要将2个3*3卷积核叠到一起？

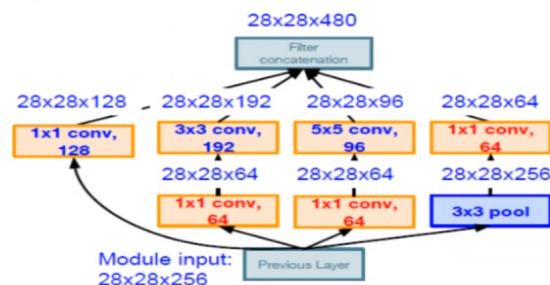
因为2个叠到一起的3*3卷积核，感受野（Receptive Field）是7*7，大致可以替代7*7卷积核的作用。但这样做可以使参数更少，参数比例大致为18:49

GOOgleNet创造的Inception结构

GoogLeNet: (Szegedy, 2014)



最初的Inception结构

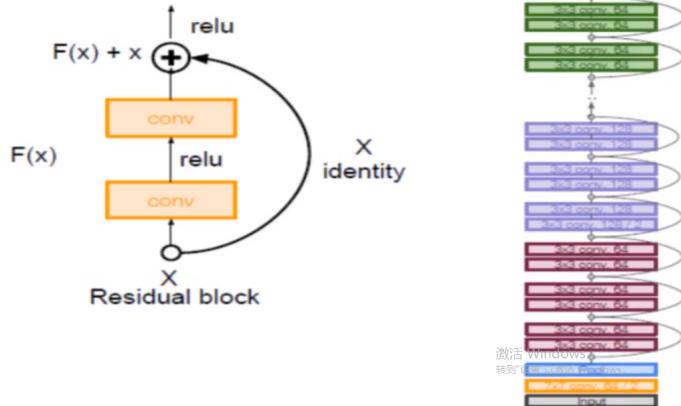


改进的Inception结构

将前层的结果输入后面

ResNet: (He et al, 2015)

- (1) 152层
- (2) ILSVRC'15冠军，
(3.57 TOP 5 ERROR)
- (3) 加入了前向输入机制，将前面层获得的特征图作为监督项输入到后面层。用这样的方法使深层网络训练能够收敛。



ResNet: (He et al, 2015)



(1) 作者首先发现了，训练一个浅层的网络，无论在训练集还是在测试集上，都比深层网络表现好，而且是在训练各个阶段持续的表现好。

(2) 在这个例子中，既然**20**层的网络表现**比56层的好**，那么大不了另外**36**层什么事情也不做，于是产生了将浅层网络的输出直接加到后面层的想法。

人脸识别

(二) 优化目标

设计我们的目标函数。参照文献，我们把目标函数确定为最小化下式：

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= -\sum_{i=1}^m \log \frac{e^{W_{y_i}^T \mathbf{x}_i + b_{y_i}}}{\sum_{j=1}^n e^{W_j^T \mathbf{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^m \|\mathbf{x}_i - \mathbf{c}_{y_i}\|_2^2\end{aligned}$$

第一项是softmax loss；第二项是center loss。

AlphaGo介绍

Mastering the game of Go with deep neural networks and tree search

2016.10.20

开源代码：<https://github.com/Rochester-NRT/RocAlphaGo>

迁移学习在水果识别上的应用

强化学习与监督学习的区别：

- (1) 训练数据中没有标签，只有奖励函数（Reward Function）。
- (2) 训练数据不是现成给定，而是由行为（Action）获得。
- (3) 现在的行为（Action）不仅影响后续训练数据的获得，也影响奖励函数（Reward Function）的取值。
- (4) 训练的目的是构建一个“状态->行为”的函数，其中状态（State）描述了目前内部和外部的环境，在此情况下，要使一个智能体（Agent）在某个特定的状态下，通过这个函数，决定此时应该采取的行为。希望采取这些行为后，最终获得最大的奖励函数值。

参考书：

(1) Reinforcement Learning an introduction. R. S. Sutton and A. G. Barto, 2005

(2) UCL Course on RL,
<http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>

(3) 例程程序
http://karpathy.github.io/2016/05/31/r1/?utm_source=1-2-2
<https://gym.openai.com/> <https://github.com/openai/gym>

一些假设

(1) 马尔可夫假设: $\mathbb{P}[S_{t+1} | S_t] = \mathbb{P}[S_{t+1} | S_1, \dots, S_t]$

(2) 下一个时刻的状态只与这一时刻的状态以及这一时刻的行为有关:

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

(3) 下一个时刻的奖励函数值只与这一时刻的状态及这一时刻的行为有关:

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

MARKOV DECISION PROCESS (MDP)

(1) 在 $t=0$ 时候, 环境给出一个初始状态 $s_0 \sim p(s_0)$

(2) for $t=0:\text{end}$

-- 智能体选择行为 a_t

-- 环境采样奖励函数 $r_t \sim R(\cdot | s_t, a_t)$

-- 环境产生下一个状态: $s_{t+1} \sim P(\cdot | s_t, a_t)$

-- 智能体获得奖励函数 r_t 和下一个状态 s_{t+1}

我们需要学习一个策略 (Policy) π^* , 这是一个从状态到行为的映射函数, 使得最大化累积的奖励。

强化学习的问题

强化学习与监督学习的区别：

- (1) 训练数据中没有标签，只有奖励函数 (Reward Function)。
- (2) 训练数据不是现成给定，而是由行为 (Action) 获得。
- (3) 现在的行为 (Action) 不仅影响后续训练数据的获得，也影响奖励函数 (Reward Function) 的取值。
- (4) 训练的目的是构建一个“状态->行为”的函数，其中状态 (State) 描述了目前内部和外部的环境，在此情况下，要使一个智能体 (Agent) 在某个特定的状态下，通过这个函数，决定此时应该采取的行为。希望采取这些行为后，最终获得最大的奖励函数值。

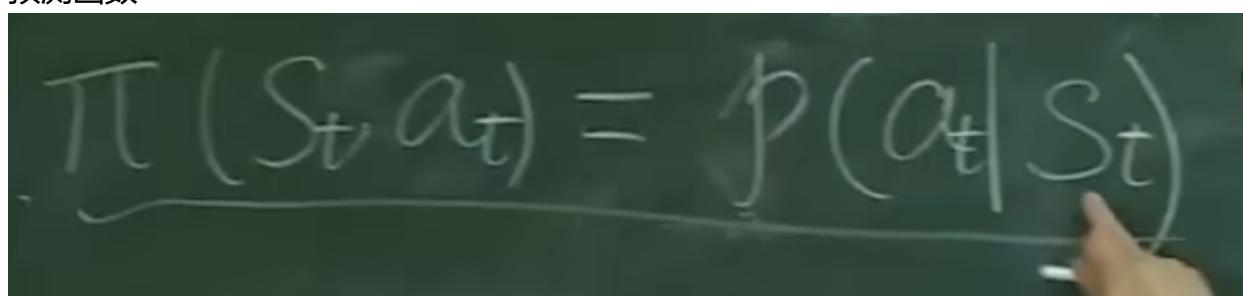
待优化目标函数

增强学习中的待优化目标函数是累积奖励，即一段时间内的奖励函数加权平均值：

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

在这里，GAMMA是一个衰减项。

预测函数


$$\pi(s_t | a_t) = P(a_t | s_t)$$

Q-learning

根据一个决策机制 (Policy) , 我们可以获得一条路径:

$$s_0, a_0, r_0, s_1, a_1, r_1, \dots$$

定义1: 估值函数 (Value Function) 是衡量某个状态最终能获得多少累积奖励的函数:

$$V^\pi(s) = \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi \right]$$

定义2: Q函数是衡量某个状态下采取某个行为后, 最终能获得多少累积奖励的函数:

$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

$$\begin{aligned} V^\pi(s) &= E_\pi \left(\sum_{t=0}^{+\infty} \underline{\gamma^t r_t} \mid S_0 = s, \pi \right) \\ &= E_\pi \left(r_0 + \gamma \sum_{t=0}^{+\infty} (\underline{\gamma^t r_{t+1}}) \mid S_0 = s, \pi \right) \\ &= \sum_{a \in A} P(a|s) \sum_{s' \in S} P_{ss'}^a (R_s^a + \gamma \underline{V^\pi(s')}) \end{aligned}$$

Q-learning

求最佳策略的迭代算法：

输入 $E = \langle S, A, P, R \rangle$

过程

① 初值 $\forall s \in S, V(s) = 0, \pi(s, a) = \frac{1}{|A|}$

② while(1)

{

 while(1)

{

$\forall s \in S$, 计算

$$V'(s) = \sum_{a \in A} \pi(s, a) \sum_{s' \in S} P_{ss'}^a (R_s^a + \gamma V^\pi(s'))$$

 if $||V - V'|| < \varepsilon$

 break;

 else

$V = V'$

}

$\forall s \in S, \forall a \in A$, 计算

$$Q(s, a) = \sum_{s' \in S} P_{ss'}^a (R_s^a + \gamma V^\pi(s'))$$

$$\pi(s, a) = \begin{cases} 1 & \text{若 } a = \operatorname{argmax} Q(s, a) \\ 0 & \text{其他} \end{cases}$$

if $||\pi - \pi'|| < \delta$

break;

else

$\pi = \pi'$

}

输出最优策略 π

Q-learning

求最佳策略的迭代算法：

这一算法的劣势：

(1) 对于状态数和行为数很多时，这种做法不现实。

例如：对一个ATARI游戏，状态数是相邻几帧所有像素的取值组合，这是一个天文数字！
ACTION数量从6到20不等

$$V'(s) = \sum_{a \in A} \pi(s, a) \sum_{s' \in S} P_{ss'}^a (R_s^a + \gamma V^\pi(s'))$$

定义

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

则有 Bellman Equation:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

Q-learning

由于: $Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$

(1) 前向计算:

Loss function: $L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)} [(y_i - Q(s, a; \theta_i))^2]$

其中: $y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a \right]$

(2) 后向传播:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right] \nabla_{\theta_i} Q(s, a; \theta_i)$$

Q-learning

DQN算法流程：

```
初始化  $Q(S, a, \theta)$  初始化记忆内存  $D$ 
for episode = 1~M do (打了M次游戏)
    初始化  $S_1$  (即进入游戏初始画面)
    for t = 1~T do (依时间采样)
        if random() < given_probability
            采取随机行为  $a_t$ 
        else
             $a_t = \max Q(S_t, a_t, \theta)$ 
        设置  $S_{t+1} \sim P_{S_t S_{t+1}}^{a_t}$ ,  $r_t = R_{S_t}^{a_t}$  □
        将  $(S_t, a_t, r_t, S_{t+1})$  存入记忆内存  $D$ 
    随机从  $D$  中取出一个batch的  $(S_t, a_t, r_t, S_{t+1})$  数据  $(S_j, a_j, r_j, S_{j+1})_{j=1 \sim p}$ 
    设置  $y_j = \begin{cases} r_j & \text{如果 } S_{j+1} \text{ 是终止状态} \\ r_j + \gamma \max Q(S_{j+1}, a', \theta) & \text{其他} \end{cases}$ 
     $\theta = \theta - [y_j - Q(S_j, a_j, \theta)] \nabla_\theta Q(S_j, a_j, \theta)]$ 
输出深度神经网络  $Q(S, a, \theta)$ 
```

Policy Gradient主要思想：在每一个状态下，根据现有的 $P(a_t|s_t)$ 采样 a_t ，如此往复，获得一组状态-行为对： $s_1, a_1, s_2, a_2, \dots, s_T$ ，此时获得最终的奖励函数 r_T ，这里我们假设 r_T 可取正负值，其中正值表示获得奖励，负值表示获得惩罚。最终我们根据 r_T 去修改每一步的 $P(a_t|s_t)$ ：

$$P(a_t|s_t) = P(a_t|s_t) + \alpha r_T$$

如果 $P(a_t|s_t) \sim Q(s_t, a_t, \theta)$ ，则有：

$$\theta = \theta + \alpha r_t \nabla_\theta Q(s_t, a_t, \theta)$$

Policy Gradient 的改进：上述算法的缺点是，我们需要非常精确的设置 r_T 的值，否则很可能出现P一直上涨或一直下降。一个主要的改进是如下：

$$P(a_t|s_t) = P(a_t|s_t) + \alpha(r_T - V(s_t))$$

$$\theta = \theta + \alpha(r_T - V(s_t))\nabla_{\theta}Q(s_t, a_t, \theta)$$

请注意 $V(s)$ 叫做估值函数：

$$V(s) = \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi \right]$$

它代表了在t时刻我们对最终REWARD的估计。

如何求 $V(s)$ 呢？也可以采用深度神经网络， $V(s) = V(s, \theta)$

POLICY GRADIENT

Actor-Critic算法：

初始化 $Q(S, a, \theta)$ 初始化 $V(S, \phi)$ 初始 $p_{\theta}(a|S) = softmax[Q(S, a, \theta)]$

for episode = 1~M do

 在现有策略 $p_{\theta}(a|S)$ 下采样m个轨迹

$\Delta\theta \leftarrow 0$

 for i = 1~m do

 for t = 1~T do

$$A_t^i = \sum_{t' \geq t} \gamma^{t'-t} r_t^i - V(S_t^i, \phi)$$

$$\Delta\theta \leftarrow \Delta\theta + A_t \nabla_{\theta} \log[p_{\theta}(a_t^i | S_t^i)]$$

$$\Delta\phi = \sum_i \sum_t \nabla_{\phi} ||A_t^i||^2$$

$$\theta = \theta + \alpha \Delta\theta$$

$$\phi = \phi + \beta \Delta\phi$$

输出 $Q(S, a, \theta)$, $V(S, \phi)$

总结

- (1) 目前强化学习的发展状况：在一些特定的任务上达到人的水平或胜过人，但在一些相对复杂的任务上，例如自动驾驶等，和人存在差距。
- (2) 和真人的差距，可能不完全归咎于算法，传感器、机械的物理限制等，也是决定性因素。
- (3) 机器和人的另一差距是：人有一些基本的概念，依据这些概念，人能只需要很少的训练就能学会很多，但机器只有通过大规模数据，才能学会。
- (4) 但是，机器速度快，机器永不疲倦，只要有源源不断的数据，在特定的任务上，机器做得比人好，是可以期待的

特征提取与特征选择
(Feature Extraction and Selection)

特征提取：主成分分析
(Principle Component)

特征选择：自适应提升算法
(AdaBoost)

特征选择问题

有一串向量 $\{x_1, x_2 \dots x_p\}$

其中 $x_i = \{x_{i1}, x_{i2} \dots x_{iN}\}$

每一个 x_i 属于 C_1 或 C_2

问题： N 个维度有冗余

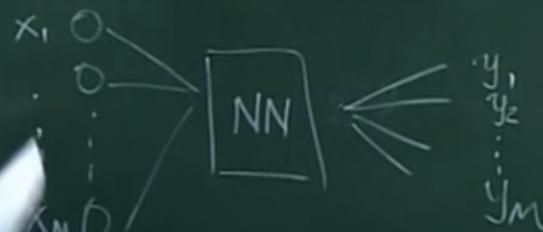
如何从 N 个维度中选取 M 个
维度 ($M < N$)，使识别率最高？
(特征选择问题)

特征提取问题

N 个维度 $\{x_{i1}, x_{i2} \dots x_{iN}\}$

构造 $\{f_1(x_{i1} \sim x_{iN}), f_2(x_{i1} \sim x_{iN}), \dots, f_M(x_{i1} \sim x_{iN})\}$

特征提取



主成分分析问题解析

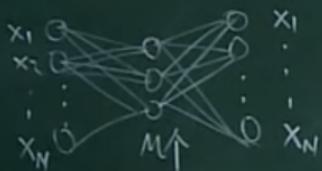
主成分分析：构造一个 A, b , 使

$$Y = A X + b$$

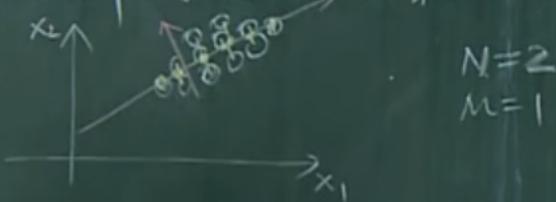
$(M \times 1) \quad (M \times N) (N \times 1) \quad (M \times 1)$

$$Y = W^T X + b$$

主成分分析可以看成是一个一层的，有 M 个神经元的神经网络。
 X 是没有标签 (Label) 的。



主成分分析：寻找使方差最大的方向，并在该方向投影。



$$\begin{aligned} N &= 2 \\ M &= 1 \end{aligned}$$

$$\left\{ \begin{array}{l} Y = A(X - \bar{X}) \\ \bar{X} = E(X) \\ \bar{X} = \frac{1}{P} \sum_{p=1}^P X_p \end{array} \right.$$

$$Y = A(X - \bar{X})$$

$(M \times 1) \quad (M \times N) \quad (N \times 1) \quad (N \times 1)$

$$A = \begin{bmatrix} \quad a_1 \quad \\ \quad a_2 \quad \\ \vdots \\ \quad a_n \quad \end{bmatrix}$$

a_i 代表一个提取方向

PCA问题最大化的函数

$$Y_i = \begin{bmatrix} \alpha_1(x_i - \bar{x}) \\ \alpha_2(x_i - \bar{x}) \\ \vdots \\ \alpha_M(x_i - \bar{x}) \end{bmatrix} = \begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{iM} \end{bmatrix} \quad (i=1 \sim P)$$

最大化: $\sum_{i=1}^P (y_{i1} - \bar{y}_{i1})^2$
 $= \sum_{i=1}^P |y_{i1}|^2 = \sum_{i=1}^P [\alpha_1(x_i - \bar{x})]^2$

最大化: $\sum_{i=1}^P (y_{i1} - \bar{y}_{i1})^2$
 $= \sum_{i=1}^P |y_{i1}|^2 = \sum_{i=1}^P [\alpha_1(x_i - \bar{x})]^2$
 $= \sum_{i=1}^P [\alpha_1(x_i - \bar{x})] [\alpha_1(x_i - \bar{x})]^T$

$$= \sum_{i=1}^P [a_1(x_i - \bar{x})] [a_1(x_i - \bar{x})]^T$$

$$\begin{aligned}
 &= \sum_{i=1}^P a_1 \left[(x_i - \bar{x})(x_i - \bar{x})^T \right] a_1^T \\
 &= a_1 \left[\underbrace{\sum_{i=1}^P}_{(I \times N)} \underbrace{(x_i - \bar{x})(x_i - \bar{x})^T}_{(N \times N)} \right] a_1^T \\
 &\quad \underbrace{\qquad\qquad\qquad}_{(N \times N)} \qquad\qquad\qquad \underbrace{\qquad\qquad\qquad}_{(N \times 1)}
 \end{aligned}$$

$$= a_1 \sum a_1^T \quad \left(\Sigma = \sum_{i=1}^P (x_i - \bar{x})(x_i - \bar{x})^T \right) \quad \text{协方差矩阵 Covariant Matrix}$$

λ 是最大的特征值, a_1 是最大值对应的特征向量

最大化: $a_1 \Sigma a_1^T$

限制条件 $a_1 a_1^T = \|a_1\|^2 = 1$

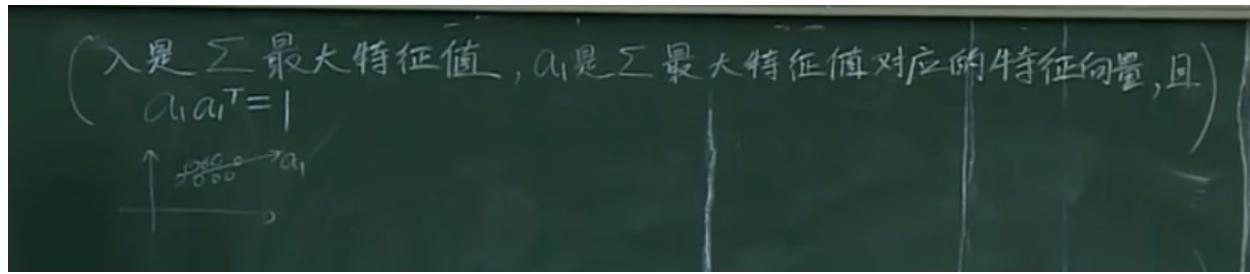
拉格朗日乘子法:

$$E(a_1) = a_1 \Sigma a_1^T - \lambda (a_1 a_1^T - 1)$$

$$\frac{\partial E}{\partial a_1} = (\sum a_1^T - \lambda a_1^T)^T = 0$$

因此

$$\sum a_1^T = \lambda a_1^T \quad (a_1^T \text{是 } \Sigma \text{ 的特征向量}, \lambda \text{ 是 } \Sigma \text{ 的特征值})$$



多维的优化问题，限制 a_2 与 a_1 正交

$$\text{最大化: } a_2^T \Sigma a_2^T$$

$$\text{限制条件} \quad a_2^T a_2 = \|a_2\|^2 = 1$$

$$a_2^T a_1^T = a_1^T a_2 = 0 \quad (\text{即 } a_1 \text{ 与 } a_2 \text{ 正交})$$

$$\text{最大化: } a_2^T \Sigma a_2^T$$

$$\text{限制条件} \quad a_2^T a_2 = \|a_2\|^2 = 1$$

$$a_2^T a_1^T = a_1^T a_2^T = 0 \quad (\text{即 } a_1 \text{ 与 } a_2 \text{ 正交})$$

拉格朗日乘子法

$$E(a_2) = a_2^T \Sigma a_2^T - \lambda (a_2^T a_2 - 1) - \beta a_1^T a_2^T$$

$$\frac{\partial E}{\partial a_2} = (\Sigma a_2^T - \lambda a_2^T - \beta a_1^T)^T = 0$$

$$\Sigma a_2^T - \lambda a_2^T - \beta a_1^T = 0$$

证明 $\beta = 0$

$$(\Sigma a_2^T - \lambda a_2^T - \beta a_1^T)^T = 0$$

$$a_2^T \Sigma^T - \lambda a_2^T - \beta a_1^T = 0$$

证明 $\beta = 0$

$$(\Sigma a_2^T - \lambda a_2^T - \beta a_1^T)^T = 0$$

$$a_2^T \Sigma^T - \lambda a_2^T - \beta a_1^T = 0$$

Σ 是一个对称矩阵，即 $\Sigma = \Sigma^T$

$$a_2^T \Sigma - \lambda a_2^T - \beta a_1^T = 0$$

$$\Sigma^T = \left(\sum_{i=1}^p (x_i - \bar{x})(x_i - \bar{x})^T \right)^T$$

$$= \sum_{i=1}^p (x_i - \bar{x})(x_i - \bar{x})^T$$

$$= \Sigma$$

证明 $\beta = 0$

$$\begin{aligned} & (\sum a_2^T - \lambda a_2^T - \beta a_1^T)^T = 0 \\ & a_2 \Sigma^T - \lambda a_2 - \beta a_1 = 0 \\ & \Sigma \text{ 是一个对称阵, 即 } \Sigma = \Sigma^T \\ & (a_2 \Sigma - \lambda a_2 - \beta a_1) a_1^T = 0 \\ & a_2 (\Sigma a_1^T) - \lambda (a_2 a_1^T) - \beta a_1 a_1^T = 0 \\ & \lambda a_2 a_1^T = 0 \quad -\frac{\beta a_1 a_1^T}{\beta} = 0 \end{aligned}$$

$$\begin{aligned} \Sigma^T &= \left(\sum_{i=1}^P (x_i - \bar{x})(x_i - \bar{x})^T \right)^T \\ &= \sum_{i=1}^P (x_i - \bar{x})(x_i - \bar{x})^T \\ &= \Sigma \end{aligned}$$

$$\begin{cases} \Sigma a_2^T = \lambda a_2^T \\ a_2 a_2^T = 1 \\ a_2 \Sigma a_2^T = \lambda \end{cases}$$

a_2 是 Σ 的特征向量, λ 是 Σ 第二大特征值

$$\begin{cases} \Sigma a_2^T = \lambda a_2^T \\ a_2 a_2^T = 1 \\ a_2 \Sigma a_2^T = \lambda \end{cases}$$

a_2 是 Σ 的特征向量, λ 是 Σ 第二大特征值

最大化: $a_3 \Sigma a_3^T$

限制条件: $\begin{cases} a_3 a_3^T = 1 \\ a_3 a_2^T = 0 \\ a_3 a_1^T = 0 \end{cases}$

a_3 是 Σ 第三大特征值对应的特征向量

主成分分析算法流程:

$$Y_i = \begin{bmatrix} a_1(x_i - \bar{x}) \\ a_2(x_i - \bar{x}) \\ \vdots \\ a_m(x_i - \bar{x}) \end{bmatrix} = \begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{im} \end{bmatrix} \quad (i=1 \sim p)$$

PCA 算法：

① 求 $\Sigma = \sum_{i=1}^p (x_i - \bar{x})(x_i - \bar{x})^\top$

② 求 Σ 的 特征值并按从大到小排序
 $[\lambda_1, \lambda_2, \dots, \lambda_m, \lambda_{m+1}, \dots]$

PCA 算法：

① 求 $\Sigma = \sum_{i=1}^p (x_i - \bar{x})(x_i - \bar{x})^\top$

② 求 Σ 的 特征值并按从大到小排序
 $[\lambda_1, \lambda_2, \dots, \lambda_m, \lambda_{m+1}, \dots]$

对应特征向量 $[a_1, a_2, \dots, a_m, a_{m+1}, \dots]$

$[\lambda_1, \lambda_2, \dots, \lambda_m, \lambda_{m+1}, \dots]$
 对应特征向量 $[a_1^\top, a_2^\top, \dots, a_m^\top, a_{m+1}^\top, \dots]$

③ 归一化所有 a_i , 使 $a_i a_i^\top = 1$

③ 归一化所有 a_i , 使 $a_i a_i^\top = 1$

④ $A = \begin{bmatrix} -a_1- \\ -a_2- \\ \vdots \\ -a_m- \end{bmatrix}$

⑤ $Y_i = A(x_i - \bar{x}) \quad | \quad (i=1 \sim P)$

求解PCA算法的特征值，用SVD算法

$$\left\{ \begin{array}{l} \frac{\partial(a_1 \Sigma a_1^\top)}{\partial a_1} = 2(\Sigma a_1^\top)^\top \\ \frac{\partial(a_1 a_1^\top)}{\partial a_1} = 2 \cdot a_1 \\ \frac{\partial(a_2 a_1^\top)}{\partial a_2} = a_1 \end{array} \right.$$

PCA在人脸识别的应用，XM2VTS数据库

PCA在人脸识别的应用

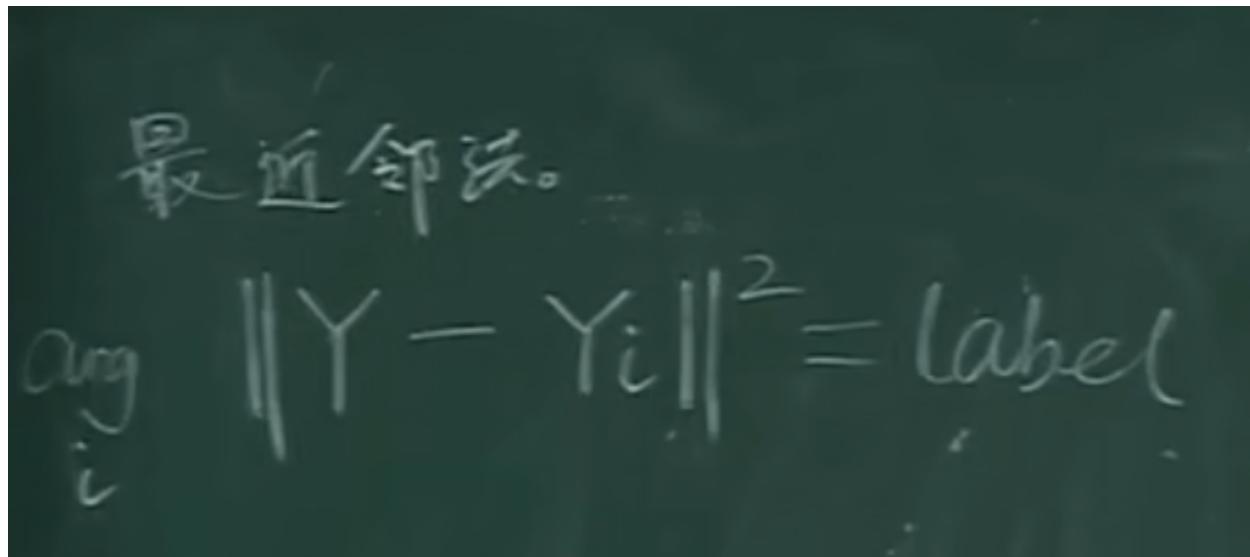
对每一个人，用前两次拍摄4张图片训练，用后两次拍摄4张图片测试。

训练数据：295*4=1180张， 测试数据：1180张

做PCA后，取前100维向量进行测试

	最近邻法	SVM
识别率	76.02%	80.25%

最近邻算法示意图



\bar{x} : 平均脸

$a_1, a_2 \dots a_m$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_1(x - \bar{x}) \\ a_2(x - \bar{x}) \\ \vdots \\ a_m(x - \bar{x}) \end{bmatrix}$$

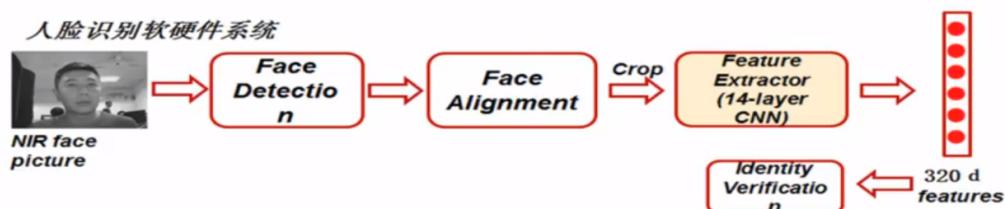
$a_1, a_2 \dots a_m$: EigenFace

特征脸

PCA在人脸识别的应用

识别器：深度神经网络，用WEBFACE数据集训练，最后一层产生320维特征向量。

识别算法：最近邻法



在瑞星微RK3288上硬件实现



特征选择算法AdaBoost：自适应提升算法（分类学习中应用）

特征选择

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

从 N 个特征中选 M 个使识别率最高。

选法数: $C_N^M = \frac{N!}{M! (N-M)!}$

启发式方法:

递增法: 首先随机选择 x_1 , 进行训练得到一个网络识别率, 接着固定 x_1 , 再加入 x_2 , 再训练得到识别率, 如率增加, 就再接着增加 x_3 , 若率减小, 就舍去增加的 x_i

递减法: 先用整体 x_1, x_2, \dots , 去训练网络得到识别率, 接着每次减去一个 x_i , 观察识别率的增减, 若减小则不舍弃, 若增加就接着舍弃 x_i

特征选择

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

从 N 个特征中选 M 个使识别率最高

选法数: $C_N^M = \frac{N!}{M! (N-M)!}$

① 递增法 $\{x_1, x_2, \dots, x_M\}$

② 递减法 $\{x_1, x_2, \dots, x_M\}$

自适应提升算法 (AdaBoost)

数据集 $T = \{(x_1, y_1), \dots, (x_N, y_N)\}$

二分类问题: $y_i \in \{-1, +1\}$

算法流程:

输入: $T = \{(x_i, y_i)\}_{i=1-N}$

输出: 分类器 $G(x) = \pm 1$

自适应提升算法 (AdaBoost)

数据集 $T = \{(x_1, y_1), \dots, (x_N, y_N)\}$

二分类问题: $y_i \in \{-1, +1\}$

算法流程:

输入: $T = \{(x_i, y_i)\}_{i=1-N}$

输出: 分类器 $G(x) = \pm 1$

① 初始化采样权值

$$D_1 = (w_{11}, w_{12}, w_{13}, \dots, w_{1N}) \quad w_{1i} = \frac{1}{N} \quad (i=1-N)$$

② 对 $m=1, 2, \dots, M$ (M 是弱分器个数)

用 D_m 采样训练样本, 在训练样本上获得弱分类器 $G_m(x) = \pm 1$

有放回的采样, 采样 N 个样本

① 初始化采样权值

$$D_1 = (w_{11}, w_{12}, w_{13}, \dots, w_{1N}) \quad w_{1i} = \frac{1}{N} \quad (i=1-N)$$

② 对 $m=1, 2, \dots, M$ (M 是弱分器个数)

用 D_m 采样 N 个样本, 在训练样本上获得弱分类器 $G_m(x) = \pm 1$

③ 计算加权错误率

$$\epsilon_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i)$$

④ 更新权值分布
 $\alpha_m = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$ (识别器 $G_m(x_i)$ 的权重)

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2}, \dots, w_{m+1,N})$$

$$w_{m+1,i} = \frac{w_{mi}}{\Sigma_m} \exp \left\{ -\alpha_m y_i G_m(x_i) \right\}$$

$$\Sigma_m = \sum_{i=1}^N w_{mi} \exp \left\{ -\alpha_m y_i G_m(x_i) \right\}$$

③ 计算加权错误率

$$\epsilon_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i) (e_m < \frac{1}{2})$$

④ 更新权值分布 (识别器 $G_m(x_i)$ 的权重) ($\alpha_m > 0$)

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2}, \dots, w_{m+1,N})$$

$$\begin{cases} w_{m+1,i} = \frac{w_{mi}}{\sum_m} \exp\left(-\alpha_m y_i G_m(x_i)\right) \\ \sum_m = \sum_{i=1}^N w_{mi} \exp\left(-\alpha_m y_i G_m(x_i)\right) \end{cases}$$

分类器 $G(X_i) \neq y$ 时取 1, 等于 y 时取 0

$$\sum_m = \sum_{i=1}^N w_{mi} \exp\left(-\alpha_m y_i G_m(x_i)\right)$$

适应提升算法 (AdaBoost)

数据集 $T = \{(x_1, y_1), \dots, (x_N, y_N)\}$

分类问题: $y_i = \{-1, +1\}$

流程:

$\therefore T = \{(x_i, y_i)\}$

⑥ 最终识别器 $G(x)$

$$f(x) = \sum_{m=1}^M \alpha_m G_m(x)$$

$$G(x) = \text{sign}(f(x)) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$$

对定理的证明, 假设 $G(X_i) \neq y$ 时, $I=1$, 不等号右边 f 是 G 的符号函数, 取值正负相同, 则 \exp 的次方为正, 右边大于 1; 若 $G(X_i) = y$ 时, $I=0$, 不等号右边 f 与 y 取相同的符号, 则 \exp 的次方为负, 但是 \exp 函数仍大于 0 小于 1, 而左边等于 0。

定理: 随着 M 增加, AdaBoost 最终分类器 $G(x)$ 在训练集上错误率将会越来越小。

$$\text{错误率 } E = \frac{1}{N} \sum_{i=1}^N I(G(x_i) \neq y_i) \leq \frac{1}{N} \sum_{i=1}^N \exp\left(-y_i f(x_i)\right) \cdot e^t > 1 \quad (t > 0)$$

若 $(G(x_i) \neq y_i)$
则 $I(G(x_i) \neq y_i) = 1$

定理: 随着 M 增加, AdaBoost 最终分类器 $G(x)$ 在训练集上错误率将会越来越小。

$$\text{错误率 } E = \frac{1}{N} \sum_{i=1}^N I(G(x_i) \neq y_i) \leq \frac{1}{N} \sum_{i=1}^N \exp\left(-y_i f(x_i)\right)$$

$$= \prod_{m=1}^M \sum_m$$

$$E \leq \left(\frac{1}{N} \sum_{i=1}^N \exp\left(-\sum_{m=1}^M \alpha_m y_i G_m(x_i)\right)\right)$$

$$= \sum_{i=1}^N (W_i) \prod_{m=1}^M \exp\left(-\alpha_m y_i G_m(x_i)\right)$$

$$\begin{aligned}
 E &\leq \left(\frac{1}{N} \sum_{i=1}^N \exp \left\{ - \sum_{m=1}^M \alpha_m y_i G_m(x_i) \right\} \right)^{-1} \\
 &= \sum_{i=1}^N \left(W_{2i} \prod_{m=1}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right) \\
 &= \sum_{i=1}^N \left[W_{2i} \exp \left\{ - \sum_{m=1}^M \alpha_m y_i G_m(x_i) \right\} \right] \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right]
 \end{aligned}$$

$$= \sum_{i=1}^N \left[W_{2i} \Xi_1 \right] \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right]$$

$$\begin{aligned}
 E &\leq \left(\frac{1}{N} \sum_{i=1}^N \exp \left\{ - \sum_{m=1}^M \alpha_m y_i G_m(x_i) \right\} \right)^{-1} \\
 &= \sum_{i=1}^N \left(W_{2i} \prod_{m=1}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right) \\
 &= \sum_{i=1}^N \left[W_{2i} \exp \left\{ - \sum_{m=1}^M \alpha_m y_i G_m(x_i) \right\} \right] \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right]
 \end{aligned}$$

$$\begin{aligned}
 &= \sum_{i=1}^N \left[W_{2i} \Xi_1 \right] \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right] \\
 &= \Xi_1 \sum_{i=1}^N W_{2i} \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right] \\
 &= \frac{1}{\Xi_1} \sum_{i=1}^N W_{2i}
 \end{aligned}$$

$$\begin{aligned}
 &= \sum_{i=1}^N \left[W_{2i} \Xi_1 \right] \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right] \quad \left| \begin{array}{l} \text{将 } \alpha_m = \frac{1}{2} \log \frac{1-e_m}{e_m} \\ \text{代入:} \end{array} \right. \\
 &= \Xi_1 \sum_{i=1}^N W_{2i} \left[\prod_{m=2}^M \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \right] \quad \left| \begin{array}{l} \Xi_1 \leq 2 \sqrt{e_m(1-e_m)} \\ \text{若 } e_m < \frac{1}{2}, \text{ 则} \end{array} \right. \\
 &= \frac{1}{\Xi_1} \sum_{i=1}^N W_{2i} \quad \Xi_m < 1 \\
 &: \quad \Xi_m = 2 \sqrt{e_m(1-e_m)} \quad \pm \quad \pm \\
 & \Xi_m = \sum_{i=1}^N W_{2ii} \exp \left\{ - \alpha_m y_i G_m(x_i) \right\} \\
 &= \sum_{i=1}^N W_{2ii} e^{-\alpha_m} + \sum_{i=1}^N W_{2ii} e^{\alpha_m} = (1-e_m) e^{-\alpha_m} + e_m e^{\alpha_m}
 \end{aligned}$$

$y_i = G_m(x_i)$

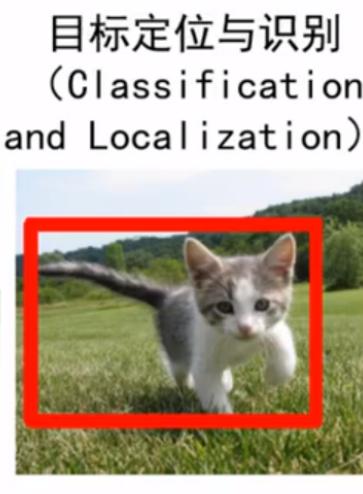
$$\begin{aligned}
 &= \sum_{i=1}^M \left[w_{2i} \Xi_1 \right] \left[\prod_{m=2}^M \exp[-\alpha_m y_i G_m(x_i)] \right] \\
 &= \Xi_1 \sum_{i=1}^M w_{2i} \left[\prod_{m=2}^M \exp[-\alpha_m y_i G_m(x_i)] \right] \\
 &= \prod_{m=1}^M \Xi_m
 \end{aligned}$$

将 $\alpha_m = \frac{1}{2} \log \frac{1-e_m}{e_m}$
 代入:
 $\Xi_m \leq 2 \sqrt{e_m(1-e_m)}$
 若 $e_m < \frac{1}{2}$, 则
 $\Xi_m < 1$

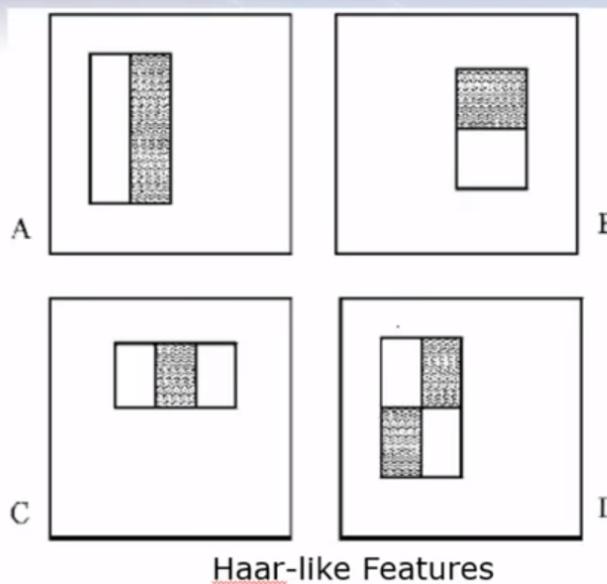
证明: $\Xi_m = 2 \sqrt{e_m(1-e_m)} \leq 1$
 $\Xi_m = \sum_{i=1}^N w_{mi} \exp[-\alpha_m y_i G_m(x_i)]$
 $= \sum_{i=1}^N w_{mi} e^{-\alpha_m} + \sum_{i=1}^N w_{mi} e^{\alpha_m} = (1-e_m) e^{-\alpha_m} + e_m e^{\alpha_m}$
 $y_i = G_m(x_i)$

AdaBoost的应用，目标检测

目标检测与分割



目标检测 (AdaBoost)



(1) 白色区域的像素值，减去黑色区域的像素值。

(2) 每一个FEATURE所有区域长度和宽度一致。

(3) FEATURE可以在整幅图上平移，只需要满足(1)和(2)即可。

(4) 可以取左图这四种形式。

对于一个 $24*24$ 的图像，所有的
Haar-like feature 个数为20万左右。

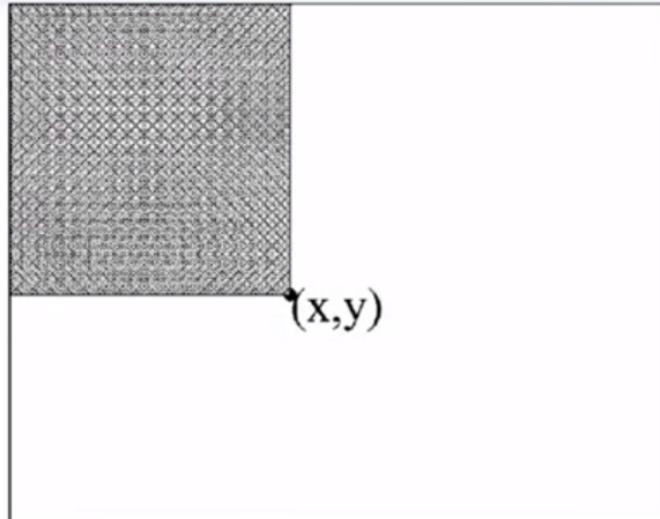
Viola and Jones, Robust Real-Time Face Detection, International Journal of Computer Vision 57(2), 137–154, 2004

目标检测 (AdaBoost)



检测效果

目标检测 (AdaBoost)



$$\text{Integral}(x, y) = \sum_{u \leq x} \sum_{v \leq y} \text{Image}(u, v)$$

目标检测 (AdaBoost)

分类器构造：取一些人脸（6000张左右）和一些非人脸（7万张）作为训练样本。

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{otherwise} \end{cases}$$

x: 图像。

f: 某一个 Haar-like feature。

θ : 阈值。

p: POLARITY, 只能取+1或-1。

对某个特定的 f , 求最佳的 θ, p 取值, 使 h 在训练样本上识别率最高。

ADABOOST算法流程

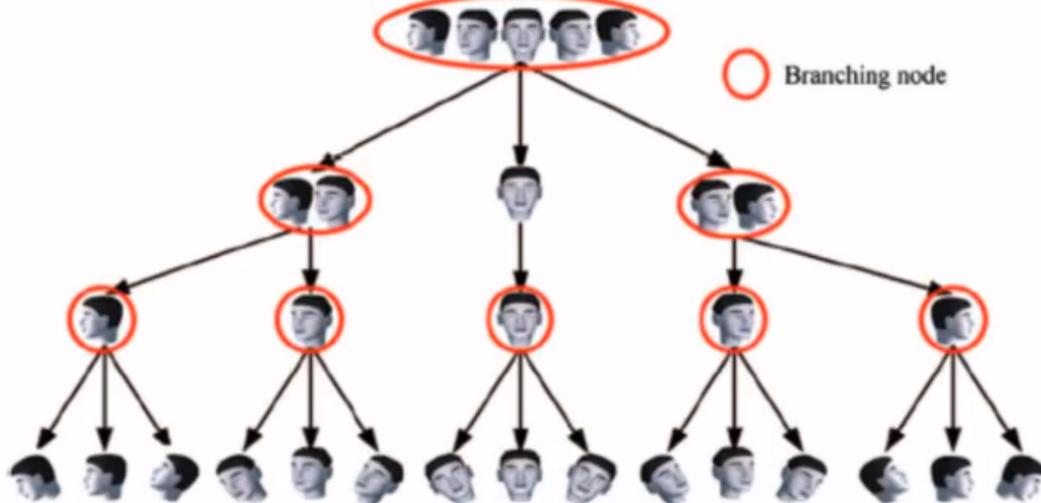
1. 首先在数据集D中选取正确率最高的特征，用F1表示。
2. 将数据集D分为两类，{F1分对的数据} 和 {F1分错的数据}。
3. 以较大概率取F1分错的数据，以较小概率去F1分对的数据，形成新的集合D2。
4. 在D2中选取正确率最高的特征，用F2表示。

目标检测（AdaBoost）

AdaBoost人脸检测流程：

1. 在图像中，对每一个24*24的格子遍历使用分类器，如果是人脸，则输出。
2. 将图像缩小，长宽同时除以1.2，在用分类器遍历每一个24*24的格子。如果是人脸，将该处位置坐标乘以1.2，等比例放大到原图。
3. 重复2，直到图像长或宽小于24个像素为止。

目标检测 (AdaBoost)



多个姿态人脸都可以构建AdaBoost分类器，树状级联后可以获得多姿态人脸检测器

Huang et al. High-performance rotation invariant Multiview face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(4), 671-686 (2007)

目标检测 (RCNN)

如何将卷积神经网络 (CNN) 用在目标检测上？

主要问题：

用大大小小的方框遍历所有图像不现实，如何快速挑出可能有物体的区域（Region of Interest, ROI）。我们需要一个计算量不那么大的算法，提出ROI的候选区域（Region of Proposals, or Proposals）



目标检测 (RCNN)

2014-R-CNN, a naïve deep detection model

Basic Ideas:

1. Use selective search to generate proposals
2. Scale and resize proposals to fit the CNN
3. SVM for final decisions

Main Problems:

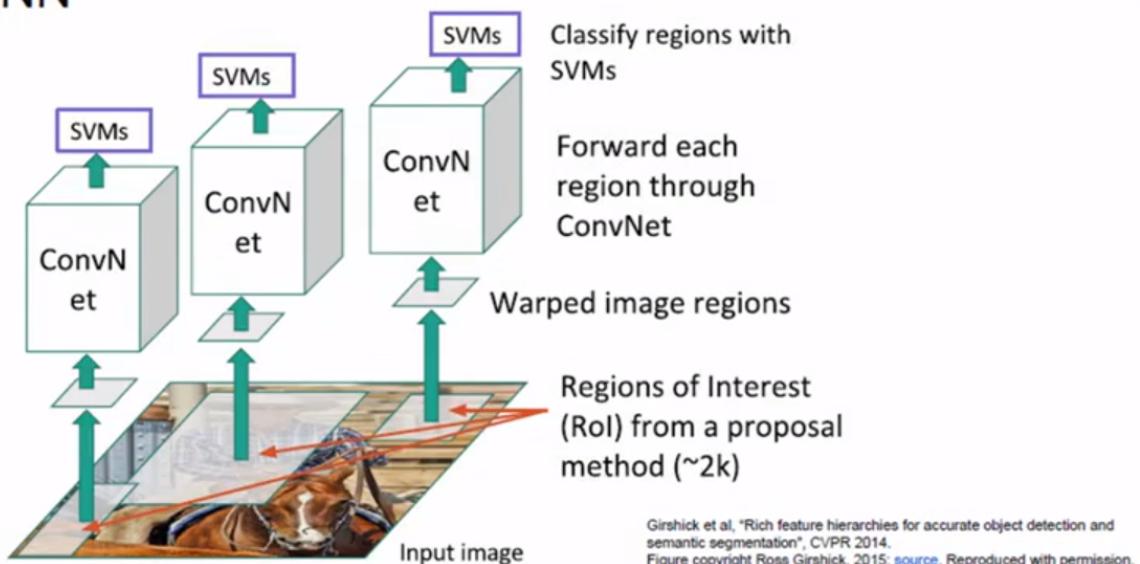
1. High cost to perform Selective Search (~5s per image)
2. Too many passes to CNN (~2000 proposals per image)
3. Lead to unacceptable test time (~50s per image)
4. High space cost to train SVM (millions of 1024-d features)



Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." CVPR. 2014.

目标检测 (RCNN)

R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." CVPR. 2014.

目标检测 (RCNN)

Region Proposals (Selective Search, SS)

给定一张图片，首先使用 Efficient Graph-Based Image Segmentation 算法，将图片进行过分割 (Over-Segmentation)

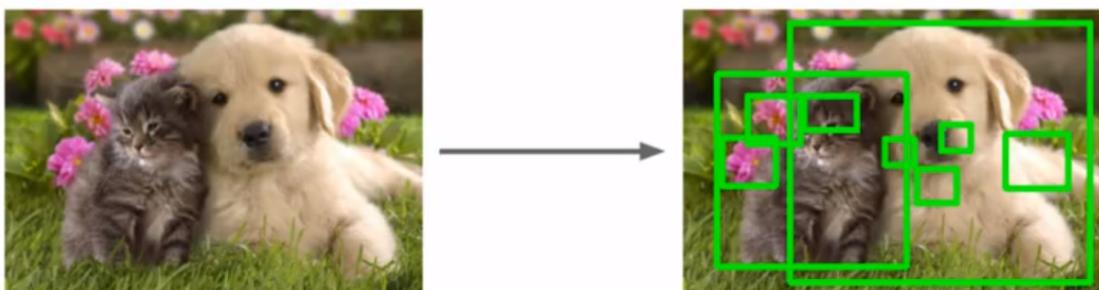


如图所示，过分割后的每个region非常小，以此为基础，对相邻的region进行相似度判断并融合，形成不同尺度下的region。每个region对应一个bounding

目标检测 (RCNN)

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU

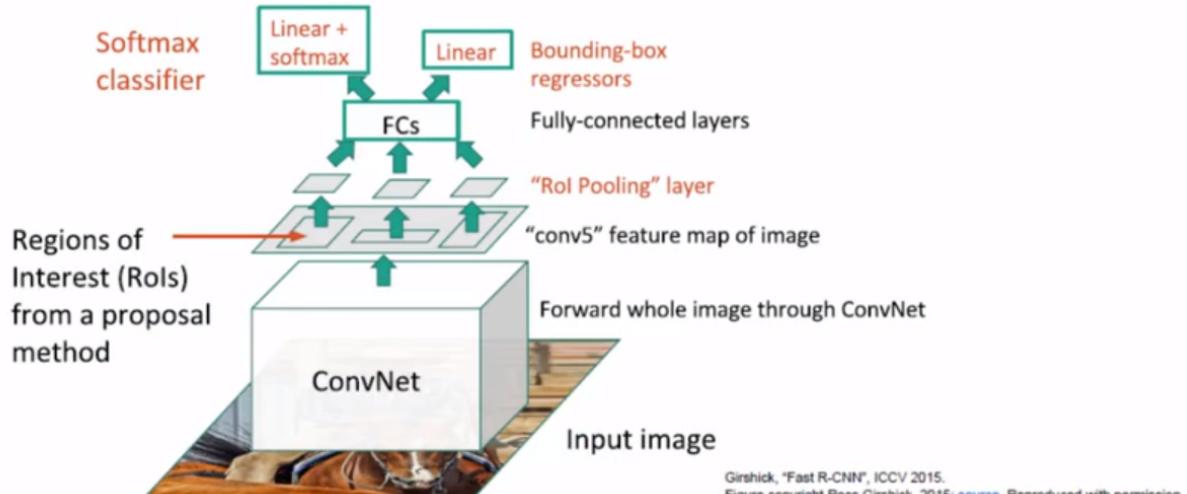


Alexe et al., "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al., "Selective Search for Object Recognition", IJCV 2013
Cheng et al., "BiNG: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

2015-Fast, RCNN解决重复计算的问题

目标检测 (RCNN)

Fast R-CNN



Girshick, Ross. "Fast r-cnn." CVPR. 2015.

目标检测 (RCNN)

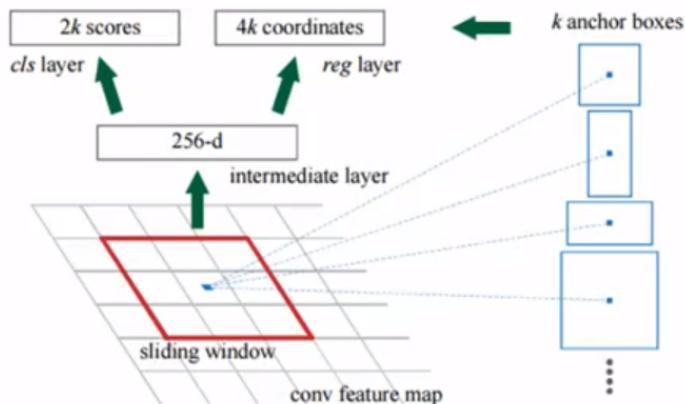
2015-faster R-CNN, RPN

Basic Ideas:

Reduce the time of generating region proposals

Main Contributions:

1. Region Proposal Network (RPN)
2. An end to end model finally!

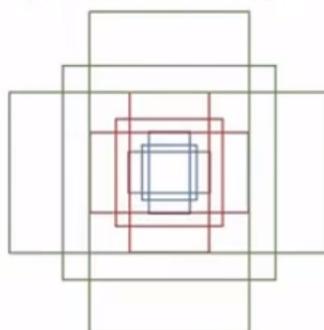


Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

目标检测 (RCNN)

2015-faster R-CNN, RPN

对于特征图某个固定点，ANCHOR 生成9个矩形，共有3种形状，长宽比为大约为：width:height = [1:1, 1:2, 2:1]三种，实际上通过anchors就引入了检测中常用到的多尺度方法。



把任意大小的输入图像reshape成800x600（即图2中的M=800，N=600）。再回头来看anchors的大小，anchors中长宽1:2中最大为352x704，长宽2:1中最大736x384，基本是cover了800x600的各个尺度和形状。

目标检测 (RCNN)

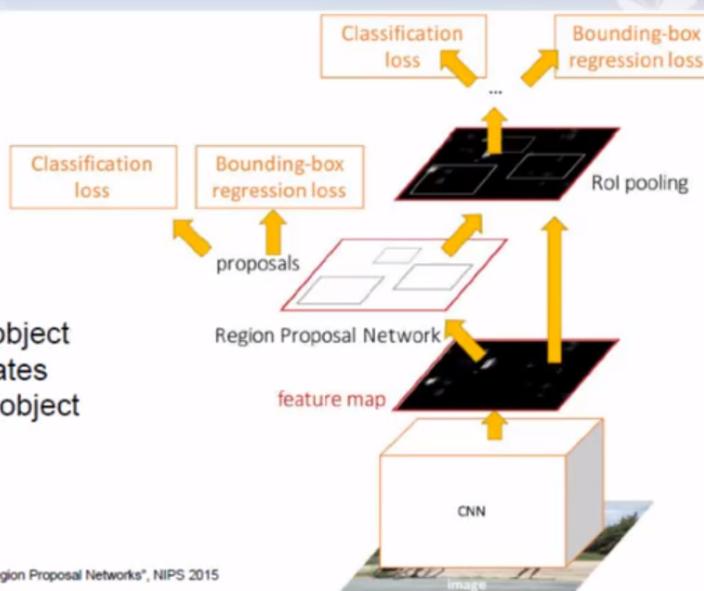
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

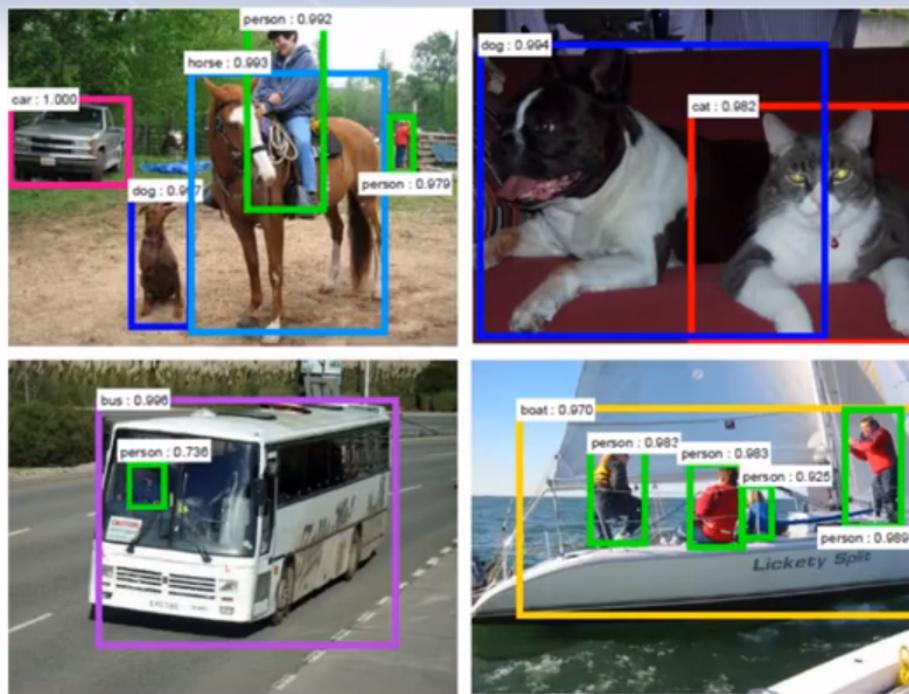
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

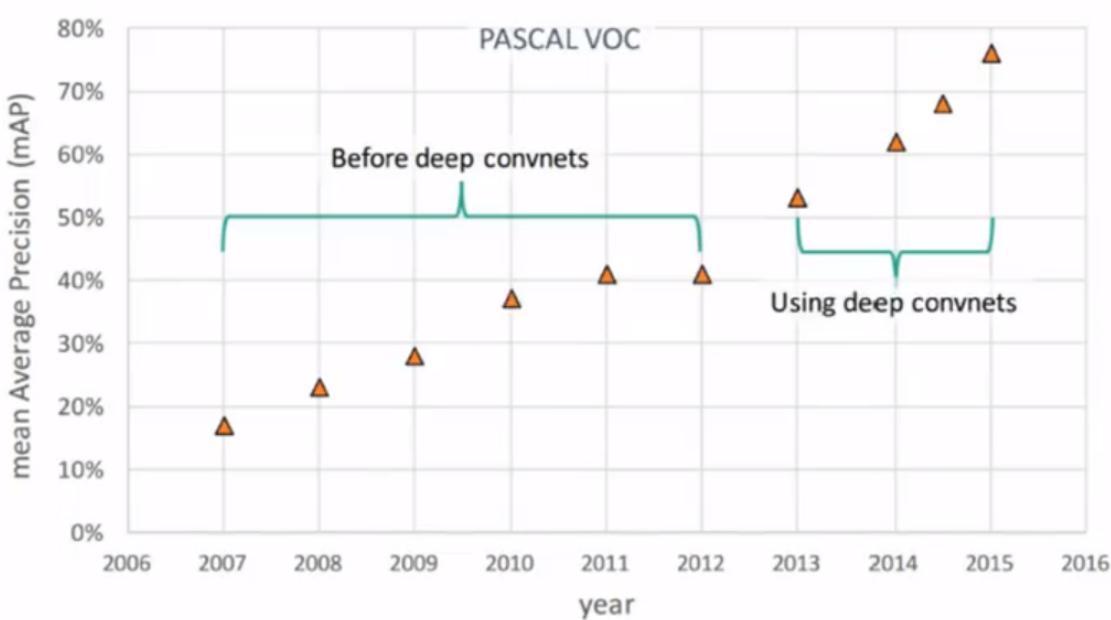
目标检测 (RCNN)



Faster R-CNN检测结果

Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." NIPS. 2015.

目标检测 (RCNN)



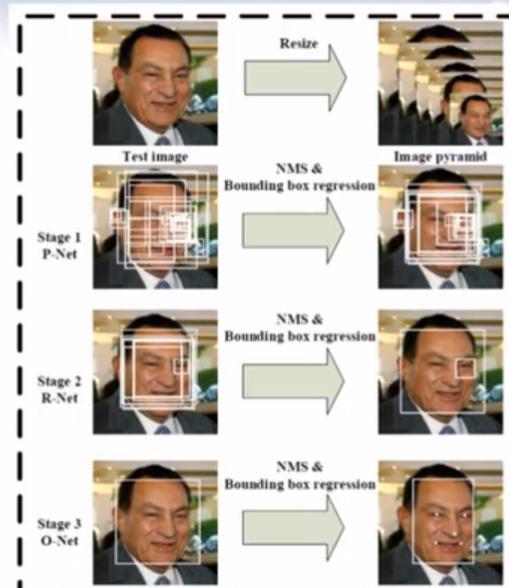
在PASCAL VOC上的性能对比

目标检测 – 以人脸检测为例

MTCNN

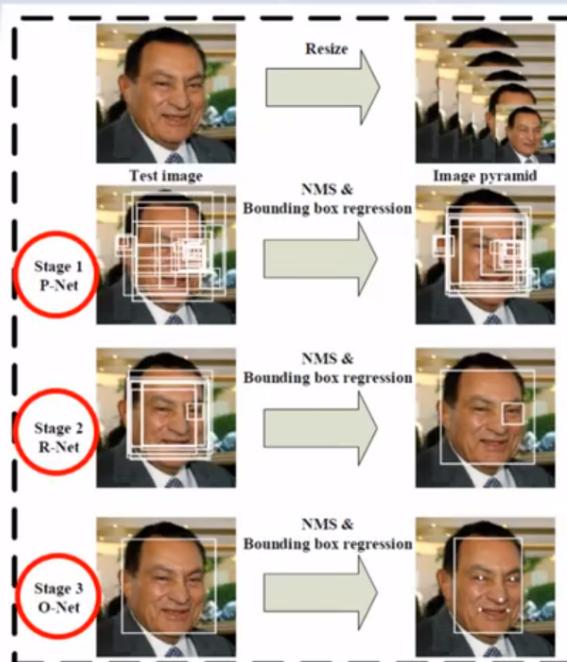
Multitask:

- 1 Face detection
- 2 Facial landmarks localization



Zhang K, Zhang Z, Li Z, et al. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks.

目标检测 – 以人脸检测为例



P-Net (Proposal Network):

该网络主要是检测图中人脸，产生多个人脸候选框和回归向量，再用回归向量对候选窗口进行校准，最后通过非极大值抑制NMS来合并高度重叠的候选框。

R-Net (Refine Network):

该网络同样输出候选框置信度（根据置信度削减候选框数量）和回归向量，通过边界框回归和NMS精调候选框的位置。

O-Net (Output Network):

比R-Net层又多了一层卷积层，处理结果更加精细，作用和R-Net层作用一样（削减框数量同时精调回归框）。再者，该层对人脸区域进行了更多的监督，最后输出5个人脸关键点坐标。

目标检测 – 以人脸检测为例



以Onet的关键点训练为例
(实际mtcnn训练label应有人脸框坐标)



FCNN的语义分割

目标检测 – 语义分割

Pooling 层的上采样 (Upsampling)

(b) Max pooling

Max Pooling

Remember which element was max!

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

Input: 4 x 4

Output: 2 x 2

Max Unpooling

Use positions from
pooling layer

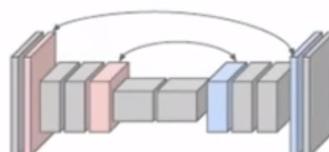
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

Corresponding pairs of
downsampling and
upsampling layers



目标检测 – 语义分割

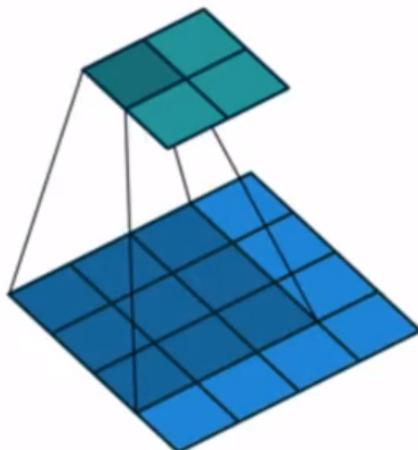
卷积层的上采样 (Upsampling)

也叫反卷积 (Deconvolution) 或 转置
卷积 (Transpose Convolution)

目标检测 – 语义分割

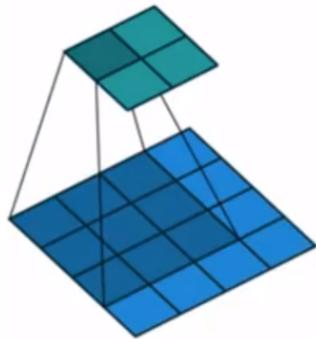
卷积层的上采样 (Upsampling)

考虑如下一个卷积层，输入特征图 $4*4$ ，卷积核 $3*3$ ，步长 1 ，
卷积后获得特征图维度为 $2*2$ ：



目标检测 – 语义分割

卷积流程



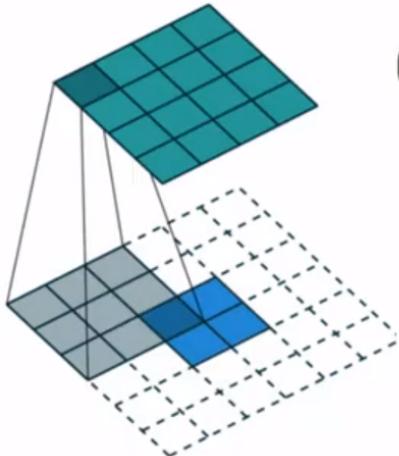
$$C = \begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \end{pmatrix}$$

$$X = [x_1, x_2, \dots, x_{16}]^T$$

$$Y = CX$$

目标检测 – 语义分割

反卷积流程



$$C = \begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \end{pmatrix}$$

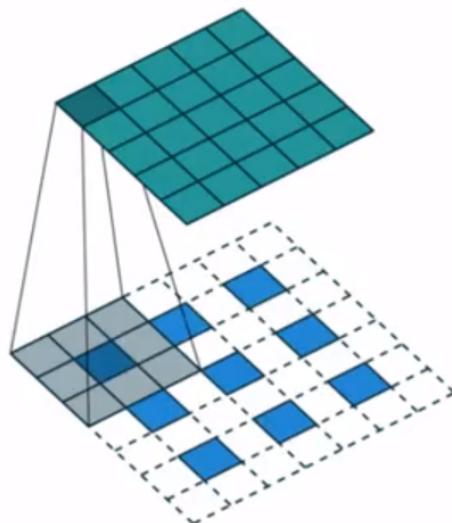
$$X = [x_1, x_2, \dots, x_{16}]^T$$

$$X = C^T Y$$

目标检测 – 语义分割

卷积层上采样另一个例子：

考虑一个卷积层，输入特征图 $5*5$ ，卷积核 $3*3$ ，步长 2 ，补零 1 ，卷积后获得特征图维度为 $3*3$ ，其反卷积示意图如下：



目标检测 – 语义分割

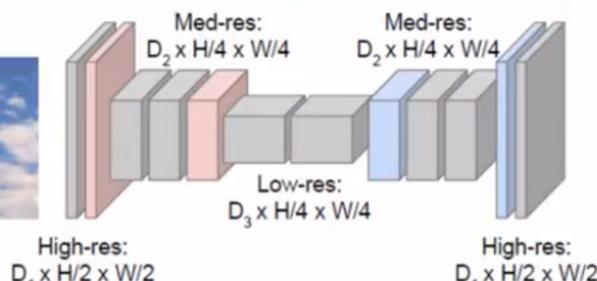
全卷积网络 – 边缘提取

Downsampling:
Pooling, stride
convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with
downsampling and upsampling inside the network!



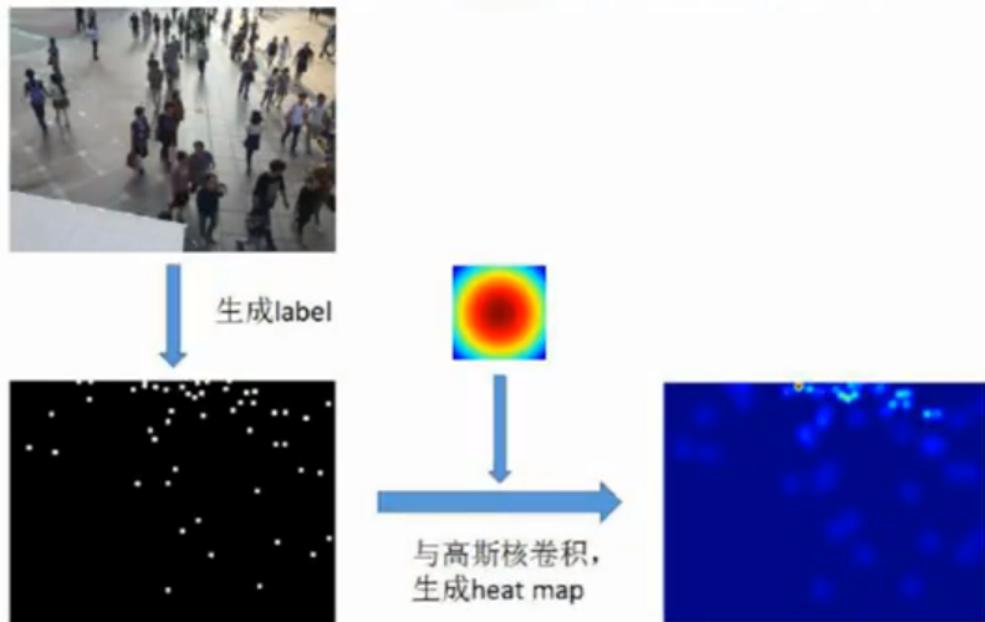
Upsampling:
???



Predictions:
 $H \times W$

目标检测 – 语义分割

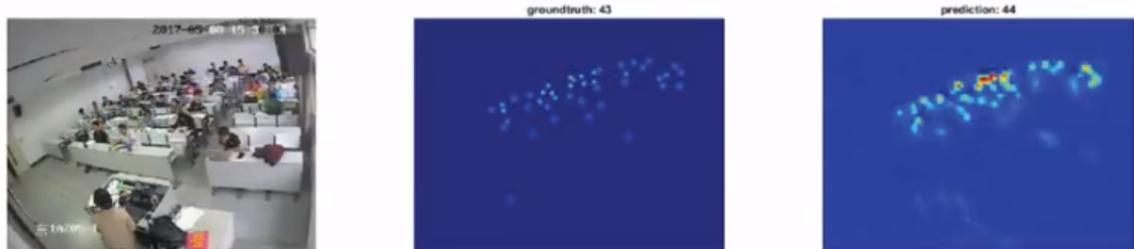
全卷积网络 – 视频场景人数估计



训练流程: Heat map 生成

目标检测 – 语义分割

全卷积网络 – 视频场景人数估计



图像

Ground Truth = 43

Prediction= 44

测试流程: 数据的Ground Truth 与 Prediction