CNN for CV

AI for CV Group
2020

Week 8.
CNN III. Other Parts

Augmentation

Data

Selection Strategy

Initialization Strategy

Learning Strategy

Optimization Strategy

Network

Framework

Modules

Layers

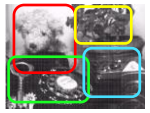
Loss Type

Loss Selection

Loss

Update Strategy

Evaluation

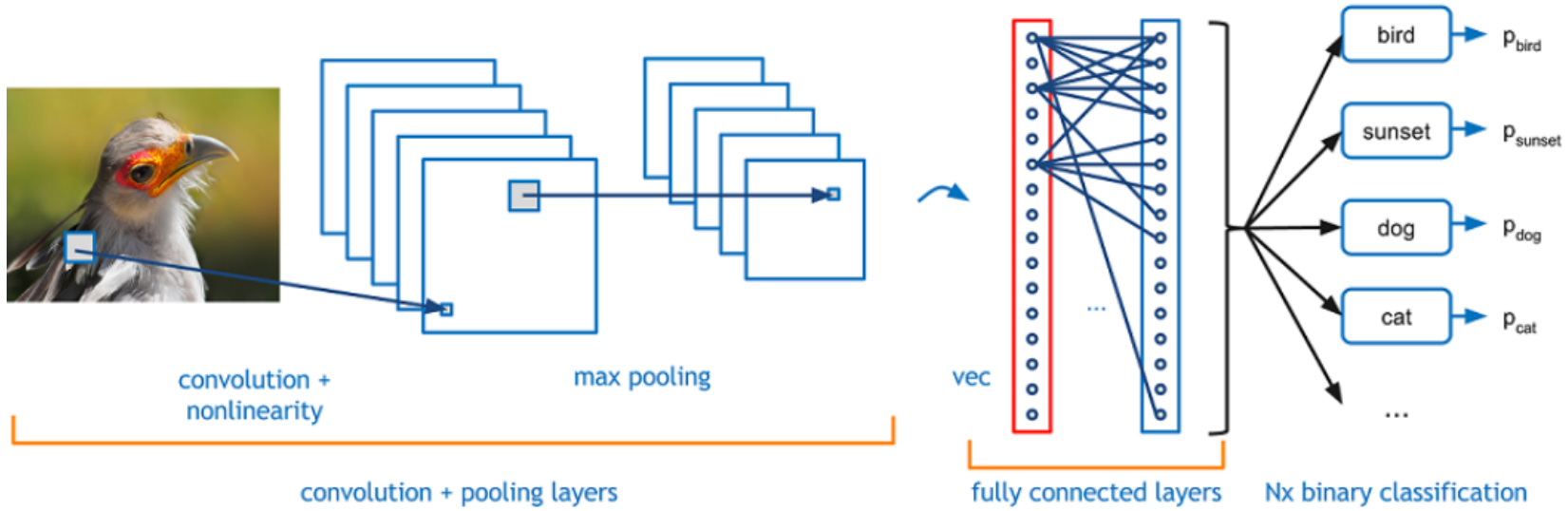


Contents:

Contents:

I. Initialization Methods

I. Initialization Methods



Model: Weights for each layer

Learn: The procedure to get our weights gradually

Q: How to start the procedure?

I. Initialization Methods

A. Gaussian / Xavier / Kaiming:

Gaussian:

Initialized each elements of weights observed by gaussian distribution

Xavier: [Bengio 2010]

We changed the std var to:

I. Initialization Methods

A. Gaussian / Xavier / Kaiming:

Kaiming (MSRA/He) [He, 2015]:

We changed the std var to:

I. Initialization Methods

A. Gaussian / Xavier / Kaiming:

Currently we use:

We changed the std dev to:

I. Initialization Methods

A. Gaussian / Xavier / Kaiming:

The reason:

Help to pass the gradient/less explosion & vanishing

Methods:

Maintain 0 mean & constant standard deviation

II. Image Preprocessing

II. Image Preprocessing

B. Traditional Ways:

1. Mean Subtraction:

```
X -= np.mean(X, axis = 0)       # 0: row; 1: col
```

2. Normalization:

```
X /= np.std(X, axis=0)
```

II. Image Preprocessing

B. Traditional Ways:

3. PCA (Comments can be found at *code* provided):

```
X = np.random.randn(1000, 500)
X -= np.mean(X, axis=0)
cov = np.dot(X.T, X) / (X.shape[0] - 1)
U, S, V = np.linalg.svd(cov)
#Xrot = np.dot(X, U)
Xrot_reduced = np.dot(X, U[:, :100])
```

4. Whitening (Seldom use):

```
Xwhite = Xrot / np.sqrt(S + 1e-5)
```

I. Image Preprocessing

B. Traditional Ways:

PCA !

I. Image Preprocessing

C. General Modern Ways:

When CNN comes:

Initially, people still tend to subtract mean first.

Now:

People find gradually that CNN's so powerful that we can actually forget about preprocessing

So:

The most important thing for data is: AUGMENTATION, which you've learnt at the very FIRST Homework!

III. Optimization

III. Optimization

D. Momentum Based

D0. SGD

III. Optimization

D. Momentum Based

D1. SGD + Momentum

III. Optimization

D. Momentum Based

D2. Nesterov

III. Optimization

E. Adaptive Methods

E1. Adagrad

III. Optimization

E. Adaptive Methods

E2. RMSProp

III. Optimization

E. Adaptive Methods

E3. Adam

III. Optimization

E. Adaptive Methods

E4. Post Adam Era

(AdaMax & Nadam)

III. Optimization

Summary for this part:

1. Adam is widely used today;
2. Adam is much faster than momentum based method;
3. However, momentum based method can usually provide us better result at final stage;
4. It's a trend that we initially use Adam then move to momentum after several training epochs;
5. One research topic is to combine features from both momentum and Adam. No winners yet.

IV. Evaluation

IV. Evaluation

Different tasks have different evaluations.

We just discuss in general way here.

IV. Evaluation

F. Recall / Precision / Accuracy

IV. Evaluation

G. AP / ROC

V. Learning Strategy

V. Learning Strategy

We have the way to update weight;

But that's not enough.

We also need to find a way to update our learning rate in order that it won't be out of work during the whole training procedure.

So HOW?

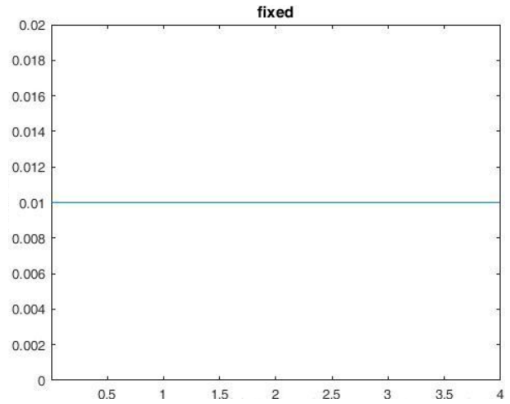
V. Learning Strategy

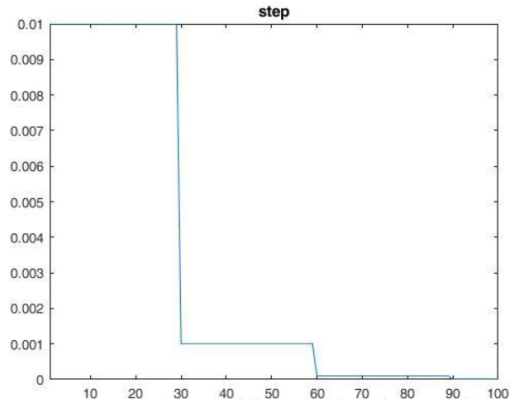
C++: Caffe [caffe.proto]

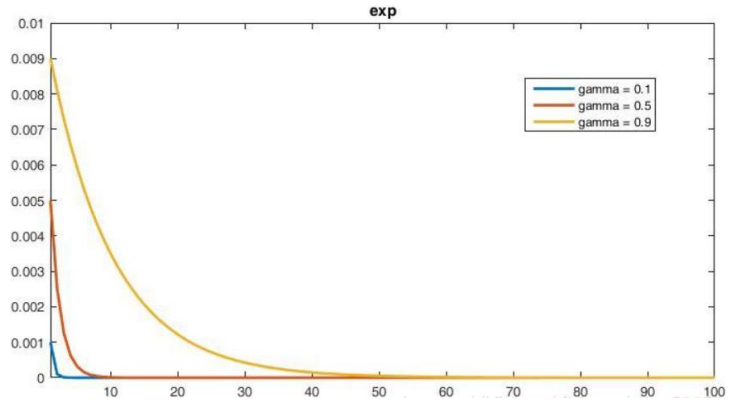
```
// The learning rate decay policy. The currently implemented learning rate
// policies are as follows:
//    - fixed: always return base_lr.
//    - step: return base_lr * gamma ^ (floor(iter / step))
//    - exp: return base_lr * gamma ^ iter
//    - inv: return base_lr * (1 + gamma * iter) ^ (- power)
//    - multistep: similar to step but it allows non uniform steps defined by
//      stepvalue
//    - poly: the effective learning rate follows a polynomial decay, to be
//      zero by the max_iter. return base_lr (1 - iter/max_iter) ^ (power)
//    - sigmoid: the effective learning rate follows a sigmod decay
//      return base_lr ( 1/(1 + exp(-gamma * (iter - stepsize))))
//
// where base_lr, max_iter, gamma, step, stepvalue and power are defined
// in the solver parameter protocol buffer, and iter is the current iteration.
```

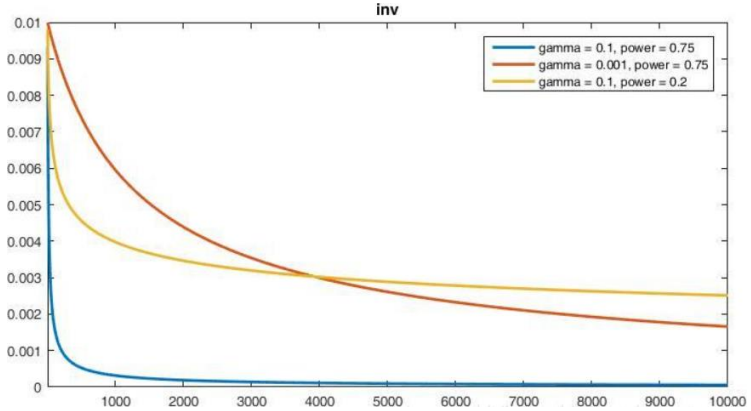
PyTorch: [torch.optim.lr_scheduler]

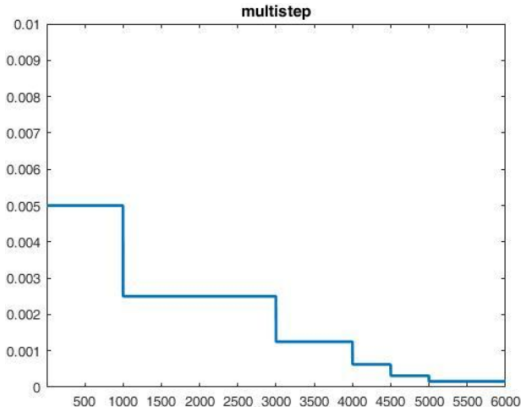
V. Learning Strategy

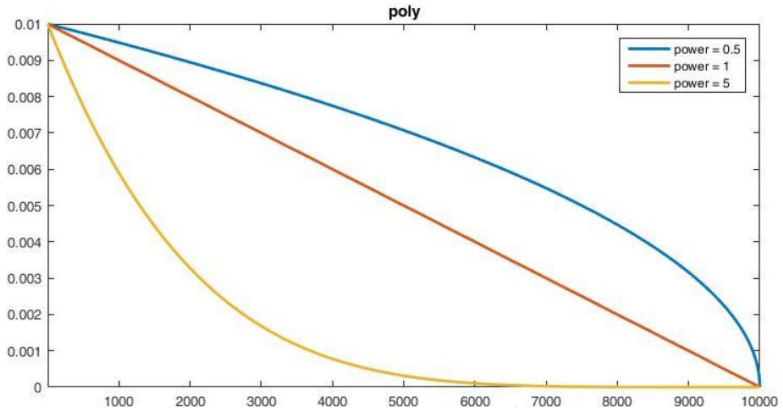












Interesting Thinking Questions for u!