双月总结报告

1.学习
2.工作
3.讨论

学习

- 学习了linux操作系统及相关脚本的使用，从零开始到现在已经基本可以自己独立修改脚本，实现了maxout+dropout、dropconnect、softmaxout等脚本的修改或编写，能够自己提取fbank、mfcc等特征。

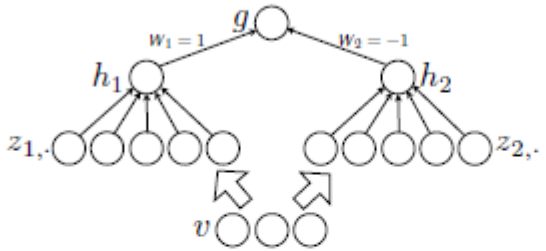
但仍然有很多需要学习的地方，对命令的掌握不够熟悉全面，不能用最简单有效的命令实现想要的功能

学习

- 学习了有关神经网络相关的知识，对常用的activation-function有了一定的了解，如

$$y = \max_{i=1}^{G} x_i$$

Maxout



$$y = \log \sum_{i=1}^{G} \exp(x_i)$$

SoftMaxout

$$y = ||\mathbf{x}||_p = \left(\sum_i |x_i \right.$$

p-Norm

sigmoid:

g(x) = 1 /(1+exp(-1)). g'(x) = (1-g(x))g(x).

tanh :

g(x) = sinh(x)/cosh(x) = (exp(x)- exp(-x)) / (exp(x) + exp(-x))

Rectifier (ReL):

- hard ReLU: g(x)=max(0,x)
- Noise ReLU max(0, x+N(0, σ(x)).

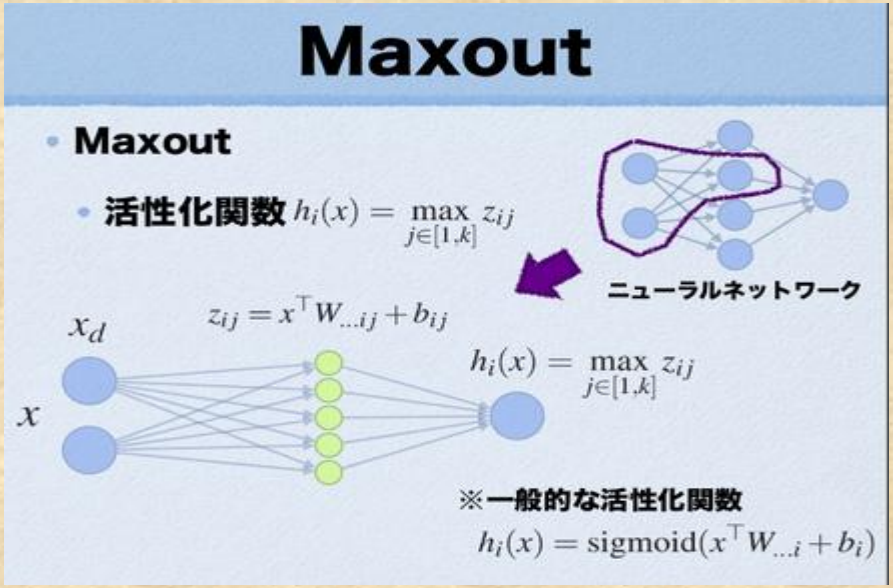
学习

- 学习dropout相关知识，了解其相关知识：
- 1. 由于每次用输入网络的样本进行权值更新时，隐含节点都是以一定概率随机出现，因此不能保证每2个隐含节点每次都同时出现，这样权值的更新不再依赖于有固定关系隐含节点的共同作用，阻止了某些特征仅仅在其它特定特征下才有效果的情况。
- 2. 可以将dropout看作是模型平均的一种。对于每次输入到网络中的样本（可能是一个样本，也可能是一个batch的样本），其对应的网络结构都是不同的，但所有的这些不同的网络结构又同时share隐含节点的权值。这样不同的样本就对应不同的模型
- 3. native bayes是dropout的一个特例。Native bayes有个错误的前提，即假设各个特征之间相互独立，这样在训练样本比较少的情况下，单独对每个特征进行学习，测试时将所有的特征都相乘，且在实际应用时效果还不错。而Droput每次不是训练一个特征，而是一部分隐含层特征。
- 4. 还有一个比较有意思的解释是，Dropout类似于性别在生物进化中的角色，物种为了使适应不断变化的环境，性别的出现有效的阻止了过拟合，即避免环境改变时物种可能面临的灭亡。

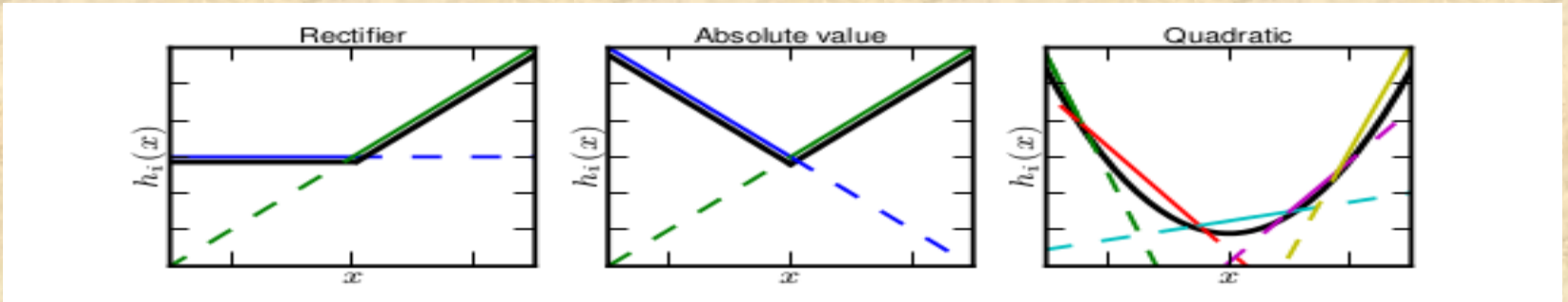
学习

- Maxout的学习理解：
- maxout其实一种激发函数形式。通常情况下，如果激发函数采用sigmoid函数的话，在前向传播过程中，隐含层节点的输出表达式为：
- $$h_i(x) = \text{sigmoid}(x^\top W_{\dots i} + b_i)$$
- 其中W一般是2维的，这里表示取出的是第i列，下标i前的省略号表示对应第i列中的所有行。但如果是maxout激发函数，则其隐含层节点的输出表达式为：
- $$h_i(x) = \max_{j \in [1,k]} z_{ij}$$
- $$\text{where } z_{ij} = x^T W_{\dots ij} + b_{ij}, \text{ and } W \in \mathbb{R}^{d \times m \times k}$$
- 这里的W是3维的，尺寸为d*m*k，其中d表示输入层节点的个数，m表示隐含层节点的个数，k表示每个隐含层节点对应了k个”隐隐含层”节点，这k个”隐隐含层”节点都是线性输出的，而maxout的每个节点就是取这k个”隐隐含层”节点输出值中最大的那个值。因为激发函数中有了max操作，所以整个maxout网络也是一种非线性的变换。因此当我们看到常规结构的神经网络时，如果它使用了maxout激发，则我们头脑中应该自动将这个”隐隐含层”节点加入。

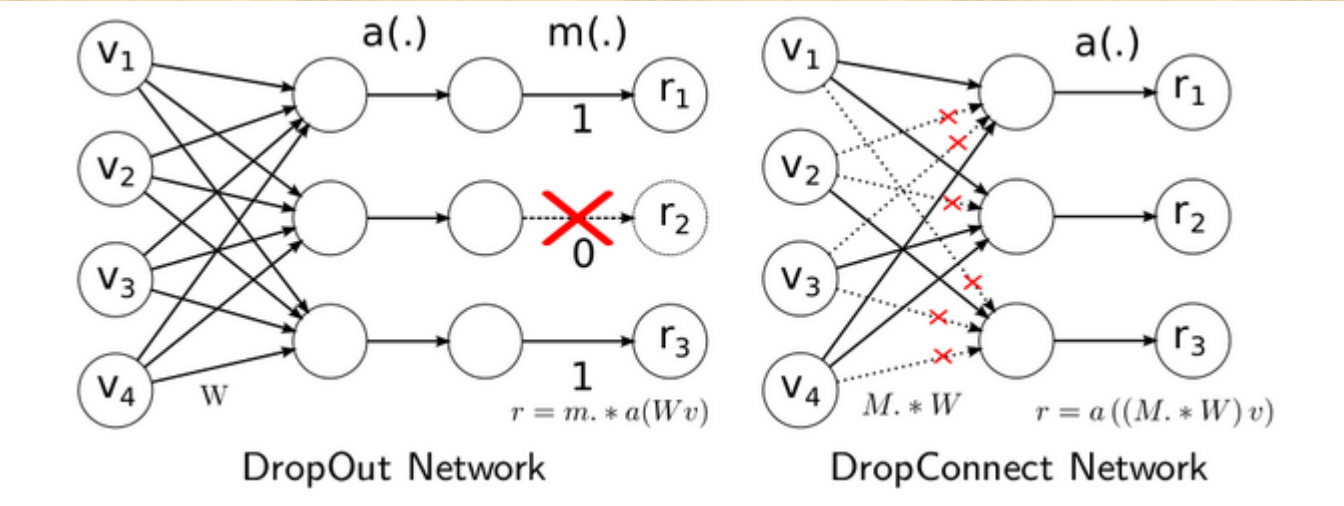
- Maxout的一个示意图（网上资料）



maxout的拟合能力是非常强的，它可以拟合任意的的凸函数。最直观的解释就是任意的凸函数都可以由分段线性函数以任意精度拟合，而maxout又是取k个隐隐含层节点的最大值，这些”隐隐含层“节点也是线性的，所以在不同的取值范围下，最大值也可以看做是分段线性的（分段的个数与k值有关）。论文中的图1如下（它表达的意思就是可以拟合任意凸函数）（在此感谢帮助我理解的老大、殷实、梦圆，好像最后是老大从王老师那得到领悟，所以还要感谢王老师）



- DropConnect的思想也很简单，与Dropout不同的是，它不是随机将隐含层节点的输出清0,而是将节点中的每个与其相连的输入权值以1-p的概率清0。（一个是输出，一个是输入）
- 其表达式如下：
- $r = a((M .* W)v)$
- 两者的区别可从下图看出



- Dropconnect我已初步完成，正在完善一些细节，尝试一些改变，具体后面再说

工作

- 工作应该是从dropout开始的，在WSJ和aurora4上都得到了不错的效果。
- 在aurora4上用clean（老大训的）、noisy、multi（基于老大给的clean，加入了噪声的训练集）等做了测试，得到了以下结果：
- 基于clean集，做了几个实验：
- 1.follow std lr
- 2.把已经训练好的std再通过有dropout的网络再次训练

工作

- 3.在clean与noisy集都验证了maxout+dropout的效果不错，结果如下：
- 4.测试了增加隐含层数量的maxout，得到如下结果：
- 5.训练了pnorm的模型，发现时间是个很大问题，效果也并非很好

13, nnet2
We test the Dan Povey codes of nnet2 to verify the parallelization of gpus using the P-norm activation-function.

	time/hours	clean	air	babble	car
pnorm-2_fast_job1	8h/08mins	6.74	29.19	28.54	16.05
pnorm-2_fast_job2	3h/38mins	5.85	27.61	26.39	17.94
pnorm-2_fast_job3	2h/31mins	6.02	29.19	27.36	16.98

8, Maxout + dropout

1) AURORA4

(1) train-noisy

drop-retention/testcase(WER)	test_clean_wv1	test_airport_wv1	test_babble_wv1	test_car_wv1
std-baseline	9.60	11.41	11.63	8.64
dp0.3_lr0.00025	28.05	40.58	38.50	26.94
dp0.4_lr0.0005	15.38	21.36	21.13	14.34
dp0.5_lr0.0005	11.88	17.02	16.66	12.01
dp0.6_lr0.001	9.90	14.01	13.94	10.00
dp0.7_lr0.001	8.95	12.45	12.60	9.14
dp0.8_lr0.001	8.59	11.44	11.69	8.70
dp0.8_lr0.001_only	9.63	11.81	11.71	8.66
maxout_lr0.001_gs6_only				

(2) train_clean (gs = 6)

drop-retention/testcase(WER)	test_clean_wv1	test_airport_wv1	test_babble_wv1	test_car_wv1
std-baseline	6.04	29.91	27.76	16.37
dp0.3_lr0.00025	11.77	41.62	42.33	27.65
dp0.4_lr0.0005	9.06	30.85	31.30	23.08
dp0.5_lr0.0005	8.02	29.17	28.37	19.08
dp0.6_lr0.001	7.18	27.02	24.50	16.39

- 6.自己编写了softmaxout的源程序和相关的代码，脚本，并训练了相关的模型（感谢老大的帮助），得到如下结果：

4, SoftMaxout
1) AURORA4 -15h

NOTE: gs==groupsize
(1) Train: train_clean

model/testcase(WER)	test_clean_wv1	test_airport_wv1	test_babble_wv1	test_car_wv1
lr0.001_gs5	6.13	29.3	27.97	14.81
lr0.001_gs6	5.92	27.08	25.15	15.56
lr0.001_gs8	6.00	29.74	26.98	16.89
lr0.001_gs10	6.04	29.09	28.27	18.41
lr0.001_gs12	6.21	28.35	26.2	16.96
lr0.001_gs15	6.26	29.78	29.72	16.91
lr0.001_gs20	5.94	30.56	29.44	16.53
lr0.0001_gs4	6.44	28.37	29.27	17.82
lr0.0001_gs5	6.23	29.78	27.55	16.05
lr0.0001_gs6	6.32	30.98	29.55	16.39
lr0.0008_gs6	5.90	27.30	24.35	16.18
lr0.00001_gs6	7.64	42.92	47.89	26.47

```
class SoftMaxout : public Component {
public:
   SoftMaxout(int32 dim_in, int32 dim_out ):
       Component(dim_in, dim_out), softmaxout_GroupSize_(10)
   { }
   ~SoftMaxout()
   { }

  Component* Copy() const { return new SoftMaxout(*this); }
  ComponentType GetType() const {return kSoftMaxout; }

  void InitData(std::istream &is)  {
    is >>std::ws;
    std::string token;
    while(!is.eof()){
      ReadToken(is, false, &token);
      /**/ if (token == "<SoftMaxoutGroupsize>") ReadBasicType(is, false, &softmaxout_GroupSize_);
      else KALDI_ERR << "Unknown token " << token << ", a typo in config?"
                     << " (SoftMaxoutGroupsize)";
    }
    KALDI_ASSERT(softmaxout_GroupSize_ > 0.0);
  }

  void PropagateFnc(const CuMatrixBase<BaseFloat> &in,
                    CuMatrixBase<BaseFloat> *out){
    //Gpu implementation of maxout
    out->SoftMaxout(in, softmaxout_GroupSize_);

//     std::cout<<"in: "<<std::endl;
//     for(int i=0;i<10;i++) std::cout<<in(1,i)<<" ";
//     std::cout<<std::endl;
//
//
//     std::cout<<"out: "<<std::endl;
//     for(int i=0;i<10;i++) std::cout<<(*out)(1,i)<<" ";
//     std::cout<<std::endl;
  }



  void BackpropagateFnc(const CuMatrixBase<BaseFloat> &in_value,
                        const CuMatrixBase<BaseFloat> &out_value,
                        const CuMatrixBase<BaseFloat> &out_deriv,
                        CuMatrixBase<BaseFloat> *in_deriv) {
```

工作

- 7.编写了dropconnect的源程序及相关代码、脚本，训练了几个模型，得到如下结果：

- 任然有些需要改进的地方，目前正在修改

```
void PropagateFnc(const CuMatrixBase<BaseFloat> &in, CuMatrixBase<BaseFloat> *out) {
  // precopy bias
  //out->AddVecToRows(1.0, bias_, 0.0);
  // multiply by weights^t
  //out->AddMatMat(1.0, in, kNoTrans, linearity_, kTrans, 1.0);
  //out->CopyFromMat(out);
  dropconnect_mask_.Resize(linearity_.NumRows(),linearity_.NumCols());
  dropconnect_mask_.Set(dropconnect_retention_);
  linear_.Resize(linearity_.NumRows(),linearity_.NumCols());
  linear_.CopyFromMat(linearity_);
  //dropconnect_vec_.Resize(bias_.dim_,kUndefined)
  //dropconnect_vec_.Set(dropconnect_retention_);
  //KALDI_LOG<<"dp= "<<dropconnect_retention_;
  //for(int i=0;i<10;i++) KALDI_LOG<<"dm= "<<dropconnect_mask_(1,i);
//    for(int i=0;i<dropconnect_mask_.NumRows();i++){
//       for(int j=0;j<dropconnect_mask_.NumCols();j++){
//        std::cout<<dropconnect_mask_(i,j)<<' ';
//     }
//  }

  rand_.BinarizeProbs(dropconnect_mask_,&dropconnect_mask_);
  linear_.MulElements(dropconnect_mask_);
 //for(int i=0;i<10;i++) KALDI_LOG<<"lin= "<<linearity_(1,i);
  //linearity_.Scale(1.0/dropconnect_retention_);

  //int i,j;
//    for(int i=0;i<dropconnect_mask_.NumRows();i++){
//       for(int j=0;j<dropconnect_mask_.NumCols();j++){
//        std::cout<<dropconnect_mask_(i,j)<<' ';
//     }
//  }
  // rescale to keep same dynamic range as w/o dropout
 // out->Scale(1.0/dropconnect_retention_);
  //dropconnect_vec_.CopyColFromMat(dropconnect_mask_,1);
  //dropconnect_vec_.Resize(bias_.Dim(),kUndefined);
  //dropconnect_vec_.set(dropconnect_retention_);
  //bia_.CopyFromVec(bias_);
  //rand_.BinarizeProbs(dropconnect_vec_,&dropconnect_vec_);
  //rand.BinarizeProbs(dropconnect_vec_,&dropconnect_vec_);
  //bia_.MulElements(dropconnect_vec_);
  out->AddVecToRows(1.0, bias_, 0.0);
   // multiply by weights^t
  out->AddMatMat(1.0, in, kNoTrans, linear_, kTrans, 1.0);
  //out->Scale(1.0/dropconnect_retention_);
```

讨论

- 参加了所有的小组讨论学习，了解了当前的一些语音识别的知识和论文，也了解了大家目前的工作
- 给大家讲了activation-function的相关内容，主要maxout及与其相关的几个函数

感谢

- 最后要感谢王老师能给我这个机会进入实验室学习；感谢老大一路过来给予我那么多的帮助和关心；感谢殷实、一叶、刘荣、梦圆等等在我刚进实验室，什么都不懂的情况下给了我很多的帮助。最后说一句，不管能否留在实验室，和大家相处的三个多月，真的收获很大，也和大家相处的很开心，非常谢谢大家给予的关心与帮助，谢谢！