Week1-1

- 3 hours class + 1 hour tutorial per week
- 2 assessment tasks (20%)-->in class test
- written examination (80%)
- *in-class material* is important!
- 100% content is ANN

Bio-computation vs Biologically-inspired computing

ANNs(Artificial Neural Networks)

- AI
  - Classic AI(logic system)
  - Modern AI
    - Machine Learning
      - ANN
      - Other
    - Bio-inspired
- ANN as a model of brain-like Computer
- recognize means classify
- Neural Network
  - Learning Process is the most important
  - Learning denotes *changes* in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population *more effectively the next time*.

Week1-2

M-P Neuron, Hebb Learning and Perceptron

- use vector as *input* in each neuron
- The McCulloch-Pitts Neuron (MP), does not involve history, it's static
  - $y=f(x^t*w)$
- Threshold activation function
  - 0, less than threshold
  - 1, larger than threshold

- MP made a base for a machine capable of
    - Storing information
    - Producing logical and arithmetical operations on it

- Brain
    - store knowledge through experiences, i.e., *learning*
    - apply the knowledge to solve problems

- *Learning*

    - In a newwork of MP-neurons, binary weights of connections and thresholds are fixed. The only change can be the change of pattern of connections, which is technically expensive.

- Two Questions:
    - What is the modle
    - How to change the connection

- Hebb Learning(learning rule)
    - Step 1(给输入, 得到输出, data 怎么transfer的)
        - 计算内积(w^Tx), use weight w to combine the input x.
        - apply the non-linear function(such as step function,像楼梯一样的那个方程)

    - Step 2
        - change connection weight, 根据那个公式 △wji ∝ CXiYi (这里是*无监督*的)

ANN learning rules

- what kind of criteria?
- most use mathmatic tools called *optimization*, 比如梯度下降
- learning ruls: how to adjust the weights of connections to get desirable output
- *Hebb learning rule*:

    - to increase weight of connection at every next instatnt in the way

Week1-3

no lecture

Week2-1

Perceptron

- what the relationship between AI, ML, NN? (*this question will appear in the test!!!*)
- supervised learning
    - (x,y) 这里的y就是label, 好比一张图片, 对应的标签是什么. *mapping*

- Step 1(计算内积, 应用方程)
    - same as MP neuron, 只不过多了一些output nueron

- Step 2(计算error, change weight,实际就是training或者learning)
    - *new* change the connection weight.这里是*有监督*的
    - 每一次输入一个input, 都会产生相应的label与之对应. 任何有监督的学习, 都必须有这样的配对
    - x输入---neuron处理(期初都是randomly给定weight值)---y输出, 这里的y会形成一个vector, 然后这个vector和之前的label对比, 看相似度.
    - 用error(误差值 = target - real output)来change connection weight, 然后再循环之前的步骤, 可能会减少误差, 这就是监督学习.
    - 这里的target是人为定义的, 是常识.
    - j 代表有多少neuron; i 代表第几个input; p代表第几个pattern
    - tjp is the target value for output unit j after presentation of pattern p
    - Xjp is the instant output produced by output unit j after presentation of pattern p
    - ejp = (tjp - Xjp)

- percetpron learning rule (error correction)
    - wji(new) = wji(old) + △wji
    - where △wji = Cejai = C(tj - Xj)ai
        - e is error; a is input; w is weight; C is *learning rate*--> how much you change each time.

Week3-1

Perceptron Rule

- The algorithm **converges** to the correct classification, if

    - the training data is *linearly separable* or the w*(solution) exist.
    - the learning rate is sufficiently small, usually set below 1.

- When the RMS error is very low, training stop.
- Why we use *bias*: to shift the line, not pass the origin=data point(就是让线不要穿过数据)
- For 2 classes, view net output as a *discriminant function y(x,w), 判别式*
- For m classes, a classifier should partition the feature space into *m decision regions*

    - the line or curve separating the classed in the *decision boundary*, 线条就是decision boundary, 被线条分开的区域叫 decision region

- x = (1,x1,x2), w = (w0,w1,w2), xwT = wxT

- $w0 + w1x1 + w2x2$
- if we can use a line to separe the two classes, we call it *Linearly separable*, in that condition, perceptron will work, but in the non-linearly separable, doesn't work.
- The weight vector should be orthogonal(正交) to the decision boundary.wTx=0
- The weight vector should point in the direction of the vector which should produce ans output of 1.
- The bias determines the position of the boundary. if we don't have the bias, the boundary will go through the origin.

Linear Separability Problem

- b + ∑(i=1-->n)xiwi = 0; 这里的b也就是w0

Week3-2

Perceptron Rule - *gradient descent*

- How the perceptron rule proposed?
- J = 1/2∑e(target-output)^2 **Important!**
- △w = -η ∂J/∂w ,j是纵坐标, w是横坐标
- To derive the learning rule, first the problem, in the perceptron, we can not derive the function, 因为它没有导数.
- 练习复合函数的导数求法

```
oe = ∑wixi  ----> 这里的i,e是下标
J= ∑e(ye-oe)^2
∂J/∂wi = ∂(1/2 ∑e(ye-oe))^2/∂wi
= 1/2∑e2(ye-oe) * 2(ye-oe)/∂wi ----> 这里的1/2就是人为定的去消掉平方2的.
= ∑(ye-oe) * ∂(ye-wixie)/∂wi
= ∑e(ye-oe)(-xie)
△w = -η∑e(ye-oe)(-xie)
accumulate
```

- *batch learning*

```
Initialization(这里的e,i,d均是下标)
data:{(xe,ye)}, e=1到N;w,η(learning rate)
Repeat:
-   for each traning example(xe,ye)
        oe=∑wixi, i=1到d, d是说明有多少个input
        if there is error
            △wi=△wi+η(ye-oe)xie 也叫*delta rule*
            update the weight with accumulate error(累加错误)
            Wnew=Wold+△w
```

Week4-1

Perceptron Learning -- 随机梯度下降, online learning, Incremental Gradient

- approximates the gradient descent error decrease by updating the weights after each training example.

```
Initialization:
{(xe,ye)}, e=1到N; wi,
Repeat:
    for each training
        calculate the output
        if the perceptron does not respond correctly update the weights:
            wi=wi+η(ye-oe)xi
    end
```

Sigmoidal Perceptrons

- 因为我们想随处可导, 所以引进这个函数(相比与step function)
- output = σ(S) = 1/(1+e^-S)
- S = ∑wixi, i=0到d
- 类似的还有tanh(S); (1-e^-s)/(1+e^-s)

```
wi ∝ -η ∂E/∂wi            这里的E就是之前的J
oe = σ(S) = 1/(1+e^-S)
E = 1/2∑e(ye-oe)^2          这里的1/2还是为了消掉那个指数2
∂E/∂wi = ∂(1/2∑(ye-oe)^2)/∂wi
       =1/2∑e (∂(ye-oe)^2/∂wi)
       =1/2∑e 2(ye-oe) * 2(ye-oe)/∂wi
       =∑e (ye-oe) * ∂(ye-oe)/∂wi
       =∑e (ye-oe) * σ'(S)(-xie)

       其中 ∂(ye-oe)/∂wi = σ'(S)(-xie)
       其中 σ'(S) = σ(s)*(1-σ(s))
△wi = △wi + η(ye-oe)σ(S)(1-σ(s))xie
```

- Perceptron
    1. Rosenblat's Perceptron rule
        - △w = η * error * input
    2. Principle objective function
        - △w = -η∂E/∂w; E = (target-out)^2 ---> SSE:Sum Squared Error
    3. Non-linear Function
        - Sigmod function
- Perceptron Vs. Delta Rule
    - **补足笔记**

Week5-1

MultipleLayer Perceptron

- Step1
    - Forward propagation, from input to output, is same like single perceptron
- Step2
    - learning, base on the algorithm.
- IMPORTATN!
    - 在hidden-output 层, 可以用线性方程, 但是在input-hidden 层, 必须用非线性方程.
- hidden layer中 neurons个数的选择:基本上是 (input*neurons + output*neurons)/2

- 但是hidden neurons太多, 也会造成overfitting(过拟合)
- 什么时候结束BP?
    - The error at the end of an epoch is below a threshold. 可以看成MSE 和 iteration 的图像, 到后面就趋于稳定.
    - All training examples are propagated and the mean error is calculated.
    - 等等
- Learning rate, 一般一个模型里用相同的, 要小. varied η: 比如在开始的时候, 用大一点的, 然后开始逐渐变小. 这个rate change with iteration.就像二次函数, 一开始很大, 后面变小.
- local minimum
    - 这就是为什么收敛的模型不一定就是好模型, 因为有可能陷入局部最优, 无法到达全局最优
- Momentum
    - **这个很重要, 要理解, 去年出了题目**
    - $\triangle w(t) = - \partial Ee/\partial w(t) + \alpha \triangle w(t-1)$, t is the index of the current weight change.
    - Momentum term: $w(k+1) = w(k) + (1-\alpha)\triangle w(k) + \alpha \triangle w(k-1)$
        - $\alpha = 0$:same as the regular BP
        - $\alpha = 1$:
        - effects: 逃离局部最优, 让训练更快, 更快收敛
- Generalization & Overfitting
- *overfitting*, 虽然在training data上表现良好, 但是在test data上就会很差. **这个overfitting也会考**
    - 为什么会发生?
        - Number of the free parameters(conncetion weight) is bigger than the number of training examples.就是说hidden unit 太多.
    - 阻止方法?
        - Ockham's Razor theorom: 不要用更大的network如果小的可行的话.
        - free parameter 不要超过training example!
        - trial-and-error, 累试法, 一遍遍试, 看哪个模型最好
        - *early stopping*, 在过拟合之前提前结束训练. 在testing时候, 记录上次的error, 每次与现在的相比, 如果现在的比以前的大了, 那就停止.不考虑之后会更好的情况.

Week5-2

Techniques to solve *overcome overfitting*

- *weight decay*: decrease each weight by some small fator during each iteration. 因为大的weight会影响

泛化, 或推广(generalization).

  - MSEreg = MSE + γ*MSW, MSW:penalises large weight
  - Large weights can hurt generalisation in two different ways:

- *cross-validation*: a set of validation data in addition to the training data. 训练的数据留一些出来不用做训练而是用作测试, 这里的测试是validation, 在训练时进行

  - 一般都有两种数据, 一个training 一个 testing.
  - 用training data中的一部分来训练, 然后用剩下的检测.
  - Holdout Valisation:random sub-sampling validation. 从training data 中随机挑选来训练.
    - A typical application the holdout method is determining a stopping point for the BP error.
    - *disadvantage* 不能保证所有的数据都被选中, 有一些数据被选很多次

      - Not enough data may be available to provide a validation set

  - *K-fold cross validation*

    - 总数据集被随机分成K份, 整个循环k次. k-1份用来training, 1份用来validate (其实也就是testing, 但是这里的testing 数据是来自training data的, 所以叫validate)
    - 分成多少份就会有多少个model, 最后选择一个最好的model.

  - *Leave-one-out cross-validation*

    - using a single observation from the original sample as the validation data......

Limitations & Capabilities of MLP

- MLP with BP can perform function approximation and pattern classifcation
- Theoretically they can
  - perform any linear and non-linear mapping
  - can approximate any reasonable function arbitrary well
  - are able to overcome the limitations of perceptrons

- In practice:
  - may not always find a solution, 陷入局部最优
  - the performance is sensitive to the starting condtinos
  - 对于数据集很敏感

Week6-1

Further Discusstion of MLP

- MLP as a Classifier

- Weakness
    - Long training time
    - require a number of parameters typically best determined empirically, eg, the network structure
    - poor interpretabilitu: difficult to interpret the symbolic meaning behind the learned weights and of 'hidden units'

- Strength
    - high tolerance to noisy data
    - ability to classify untrained patterns
    - well-suited fo continuous-valued inputs and outputs
    - successful on a wide array of real-world data
    - algorithms are inherently parallel
    - techniques have recently been developed for the extraction of rules from trained neural networks.

- Example: classification of postal codes **这种问题会在考试中出现, 设计一个MLP解决问题**

    - segment the digit as image
    - change the image into vector(一维)
    - specifiy the structure, 比如数字是20 * 20的, 那么拓展成一维的就是400 * 1, 就有401个input(加上bias)
    - 考虑output, 有10个数字. 最常用的, one perclass coding.
        - 1 2 3 4 5 6 7 8 9 10(也就是对应的0-9)
        - 1对应的vector是{1,0,0,0,0,0,0,0,0,0}
        - 10对应的是{0,0,0,0,0,0,0,0,0,1}
        - 多少hidden unit? 基于经验来说, 小于 1/2(input + output), 但对于这道题, 200依旧很大,很难拟合, 所以估计150好一点.

- Deign MLP 的步骤:
    1. Preparation of data.
    2. specify the model structure
    3. specify the leaning parameters, η, α之类
    4. specify the training procedural.**可以写代码, 或者伪代码, 或者直接语言描述都行, 别忘了说怎么避免过拟合这些问题**

- 为什么用input当target?(有些训练会这样)
    - 这叫无监督学习, identity mapping.
    - x → x
    - 目的是, 发现一些信息藏在input中.

Convolutional Neural Network

- MLP 涉及到的operation: dot product & sigmod
- CNN: 卷积操作

Week7-1

Basic Concept of CNN

- Sparse(稀疏) Connectivity
  - 不是接受所有的input, 而是这个接受几个, 那个接受几个,如此
  - Reduce parameter
  - *Share weight*

- 一般不用全部的图像, 就用local regions(部分)
- 在这里, 每个unit叫feature map, 也就是mlp里面的output, mlp里面是一个数字, 这里是一个图片, 或者说是矩阵
- mlp里的weight在这里是filter
- 过程:
  - Input Image -> Convolution -> Non-linearity -> Spatial pooling -> Normalization -> Feature maps. 然后再repeat这个循环.

- 不用sigmod的原因, 因为作用范围太小了, 就是变化的那部分太短, 之后就过饱和.
- 用什么函数呢?
  - Rectified Linear Unit(*ReLU*)

- Local Pooling Operation
  - in order to reduce variance, pooling layers compute the *max* or *average* value of a particualr feature over a region of the image. 导致参数减半甚至更多.
  - *max-pooling*: is a non-linear down-sampling. max is better than average

- Local Contrast Normalization
- end-to-end: input data --- output data

- CNN更多的是representation learning

Training of CNN

- Forward Propagation
- Autoencoders
  - copy input to output(非监督学习)

- Deep learning can jointly optimize key components in vision systems
- Challenging prediction tasks can make better use the large learning capacity and avoid overfitting

Week8-1

Task of supervisor training

- classification & prediction
- *curve-fitting*(or *function approximation*)

Radial-Basis Functions

- interpolation functinos F
    - $F(x) = \sum_{i=1}^{n} w_i\Phi(\|x - x_i\|)$

- How do we find the wi?
    - w is the weight vector, y is the response vector, and the Φ is the design matrix
    - $\Phi = \{\Phi_{ei} | \Phi_{ei} = \Phi(\|x_e-x_i\|, e,i = 1,2....n)\}$

- Gaussian Function
    - $G(x,x_i) = \exp(-(\|x - x_i\|)^2/2\sigma^2)$
    - $F(x) = \sum_{i=1}^{N} w_iG(x,x_i)$

Regularization Network

- Φi-Φn, Φ的数量和input data的数量一样多
- $w = (\Phi^T\Phi)^{-1}\Phi^Ty$
- 不需要这个Φ方程穿过每一个数据点

RBF Network Structure

- Hidden layer: applies a non-linear transformation from the input space to the hidden space.
- Output layer: applies a linear transformation from the hidden space to the output space.
- Reduce the Φ 数量
- Two stages of operation
    1. what is the center ti(就是从很多数据里面取出来的一些), input->hidden上的东西, 就是center的分量(component)
    2. what is the wi btw the hidden->output

Week8-2

RBF network

- input -> hidden: 只是transform, center
- hidden -> output: mapping

- 怎么找到center? 用 **K-Means**

- Training RBF network

    1. 选择center unit 从input 中, 以此来减少输入的个数

Week8-3

RBF

- 分差法: 高斯函数的线性叠加能逼近任意非线性函数.
- **pseudo-inverse**: 回传过程.

    - 效果和逆相似. (逆: XY=I, Y = X的逆.)

Week10-1

Time Series

- most time series can be modeled mathematically as a **wide-sense stationary(WSS) random process**
- some ts exhibits *chaotic nature*.
- t is real-valued, and x(t) is a continuous signal.
- to get a series {x[t]}, must sample the signal at discrete points.
- {x[t]} = {x(0), x(1), ... x(n)}

Problem of Predicting the Future

- x[t+s] = f(x[t], x[t-1], ...)
- s is called the *horizon of prediction*

TS Prediction

- Difficult

    - Limited quantity of data
    - Noise
    - Non-stationarity

- Steps(Moving-Window)

    - 比如用数据(天), 1到4天作为输入, 然后第五天的数据作为target, 用之前学的MLP进行.
    - 接着用2到5天作为输入, 然后第六天的数据作为target来train.

- 用这种方法迭代.

Time-series and Elman Network

- Motivation for dynamic networks
  - our requirement is not met by a function (no matter how complex it may be) and demands *a model which maintains a state, or memory*, over several presentations.
  - MLP & RBF are *static networks*: they learn a mapping from a single input signal to a single output response for an arbitrary large number of pairs
  - **Dynamic network** learn a mapping from a single input signal to a sequence of response signals, for an arbitrary number of pairs. (short trem memory)
- Sequence Learning II
  - Implicit
  - Explicit
- Time Delayed Networks
  - illustrate the implicit representation approach for sequence learning, which combine a static network with a memory structure.(e.g. tapped delay line延时线寄存器)

Dynamical Neural Nets 预测的时候, 比MLP 的结果要好一些

- Introducing time
  - Recurrent nets can explicitly deal with inputs that are presented sequentially, as they would almost always be in real problem.
- Structure of ELman Network
  - output layer
  - hidden layer
  - context layer
- Learning
  - 就像BP一样
- Matlab Implementation
  - newelm(): create an Emlan BP network(old)
  - elmannet: create an Emlan BP network(new)
  - trains()

- learngdm()

Unsupervised Learning

Introduction

- Unsupervised learning discovers significant features or patterns in the input data through general rules that operate locally
- UL networks typically consist of two layers with feed-forward connections and elements to facilitate 'local' learning

Henbbian Learning (1949)

- w(n+1) = w(n) + ηx(n)y(n); 这里的n是循环次数
- for linear activation function
  - w(n+1) = w(n)[1+ηx(n)^Tx(n)]
  - weights increase without bounds.

- Hebb learning is intrinsically *unstable*, *不拟合*. unlike error-correction learning with BP algorithm.
- Geometric Interpretation of Hebbian Learning
  - consider a single linear neuron with p inputs
  - y = w^Tx = x^Tw

- △w = [∑n=1->N x(n)^Tx(n)]w(0)
- R(x) - 1/n ∑n=1->N x(n)^Tx(n)
- 虽然不拟合, 但是方向是朝着数据最大变化的方向. ....(补上笔记)..

Week10-2 (PCA)

Oja's Rule(1985) 奥雅

- wji(n+1) = (wji(n) + ηxi(n)yj(n)) / (∑i [wji(n) + ηxi(n)yj(n)]^2); apply normalization(范化), 确保w在1以下; 说白了就是w=W/lwl
- 不是局部的, 是全局的. no local.
- Oja approximated the normalizatino (for small η) as:
- **wji(n+1) = wji(n) + ηyi(n)[xi(n) - yj(n)wji(n)]**
- 上面的式子也就是wji(n+1) = wji(n) + ηyi(n)xi(n) - ηyi(n)^2*wji(n); 最后一部分是新的.

Oja's rule: Geometric Interpretation, 关心 data structure

- The Hebbian rule finds *the weight vector with the largest variance with the input data*

- But the weight vector increases without bounds
- Oja's rule has a similar interpretation:
  - normalization only changes the magnitude while the direction of the weight vector is same
  - Magnitude is equal to one

- Oja’s rule converges asymptotically
- PCA -> transform

The Maximum Eigenfilter

- $Re1 = \lambda 1e1$; $i = 1...N$
  - R: auto-correlation matrix of input data
  - e1: largest eigenvector which corresponds to the weight vector w obtained by Oja's rule, 这个代表最大的数据方向
  - $\lambda 1$: largest eigenvalue, which corresponds to the network's output

Summary of Oja's rule

- $\Delta w(t) = \eta\ [x(t)y(t) - w(t)y(t)^2]$
- Oja learning Rule is equivalent to a normalized Hebbian rule.
- By the Oja learning rule, w(t) converges asymptotically to
  - $w = w(\infty) = e1$; where e1 is the principal eigenvector of Rx.

- 可以缩小数据的维度, 用投影的方式.二维的投影到坐标轴上就成了一维的.
- 只能找到一个最大的eignvector. 你会说再加一个neuron, 但是用的方法都是一样的, 所以两个没有区别.
  ## Week11-1
- Deflation Method: subtract the principal component from the input
  - Assume that the first component is already obtained; then the output value can be "deflated" by the following transformation:
  - $Xnew = (I - W1*W1^T)Xold$; no local, 就是数据用的不是直接的input和output

- Sanger's Rule - Generalized Hebbian Algorithm(GHA), 是local的
  - consider p inputs and m outputs, where $m < p$
    - $yj(n) = \sum_{i=1->p} wji(n)xi(n)$, $j = 1,...,m$

  - and the update(Sanger's rule)
    - $\triangle wji(n) = \eta[yj(n)xi(n) - yj(n)\sum_{k=1->j} wki(n)yk(n)]$,这是标量

- Further explanation of Sanger's Rule
  - $wjnew = wjold + \eta[yjx-yj\sum_{k=1->j}wkyk]$, $j=1,...m$, 这是矢量

- 考虑降维之后能不能恢复
- 最大化数据变化方向

- Summary of GHA
    - Oja's rule + Deflation = Sanger's rule

Dimensionality Reduction

- PCA
- Other PCA neural networks
    - Multi-layer networks:auto-encoder
        - 只用线性函数
        - wij和wjk是大约重复的

K-Means Algorithm

参考CSE315 Machine Learning里的笔记, 用excel表格做的那个K-means计算.

Week11-2

Competitive Learning

- Competitive learning means that *only a single neuron from each group fires at each time step*
- Output units compete with one another.
- These are *winner takes all*(WTA) units 在每次的循环中.

- how can we decide which neuron win?
    - 用相似度. 用input和neuron 的output相比, 相似度最高的胜出, 换句话说, 就是距离最近的胜出.

- after decided the winner, how to learn???
    - move the winner even closer towards the example.
    - △wj*i = ηxi; 这里的 j*是winner neuron, wj*i是input和这个winner neuron之间的weight.
    - only the incoming weights of the winner node are modified
    - 其他的式子: △wj*i = ηyj(xi-wj*i)
        - yj* = 1
        - yj = 0 if j ≠ j*

- 结论: The clusters formed are *highly sensitive* to the initial data.

- 有些unit永远都是loss的, 没有学习, 没有激活, 叫 *dead unit*

- 怎么解决?

- leak leaning

- 鼓励loser学习一点点

Week12-1

Vector Quantization(VQ)

- idea: categorize a given set of input vectors into M classes using CL algorithms, and then represent any vector just by the class into which it falls.
- divides entire pattern space into a number of separate subspaces= set of M units represent set of prototype vectors: *CODEBOOK*码本
- new pattern x is assigned to a class based on its closeness to a prototype vector using Euclidean distance.
- 每个neuron产生一个codebook.

NN with Competition Mechanism

- 怎么设计一个网络让他自动实现wta而不是用程序算出来?----用生物学的方法.
- Lateral inhibition: output of each node feeds to others through inhibitory connections(with negative weights)
- a specific competitive net that performs wta competition is *Maxnet*
- Maxnet.
    - we define *fix negative connection weight*
    - wji = o (if i = j); or = -ς
    - 一个神经元激活了, 就会给别的神经元传递负值, 就抑制了别的神经元.
- *Mexican Hat Network*
    - For a given node
        - close neighbors: cooperative(mutually *excitatory*, w > 0), 近的给大的weight
        - farther away neighbors: competitive(mutually *inhibitory*, w < 0)
        - too far away neighbors: irrelevant(w = 0)
    - Need a definition of *distance(neighborhood)*:
        - one dimensional: ordering by index(1,2,...,n)
        - two dimensional: lattice

Self-Organizing Map(SOM) - Biological Motivation

- Topographic maps
    - we want nonlinear transformation(PCA 是linear的) of input onto output feature space *which preserves neighbourhood relationship between the inputs*, a feature map where nearby neurons

respond to similar input.

小总结

- Unsupervised learning
    - PCA
        - Hebbian learning
    - Clustering
    - SOM(Kohonen)

SOM 这个期末考试会考, 很大比重

- why SOM?
    - SOM is to *transform on input of arbitrary dimension into a 1 or 2 dimensional discrete map*
- neurons are arranged in *geometric shape*(grid栅格)
- 为什么可以看成二维的? 因为人的大脑皮层是一个曲面, 但是一小部分可以看成一个平面, 每个平面掌管不同的功能.

SOM Training Algorithm

1. Initialization
    - between each neuron, there are weight.
2. Competition, 先在output层找出最大的
    - after have input x, we can calculate the ∑wx, 然后看哪个neuron是最大的. pick up the winner, 也就是|w-x| 最小的那个.
3. Cooperation, 然后再根据远近来分配不同的weight
    - 用Mexican Hat 的方法, 选出close neighbors让它们加入训练过程. topological neighborhood hj,i
    - 也可以用高斯方程.hj,i = exp(-(dj,i)^2/2∂^2)
    - 其他的neuron就不参加训练
    - learn means change connection weight
    - wj(t+1) --> 第二层上面的点 = wj(t) + η(t)hj,i(t)(x-wj(t)); j是neighborhood的index
        - η(t) = η0*exp(-t/τ)
        - hj,i(t) = exp(-(dj,i)^2/2∂(t)^2) 这个h是新的
    - 让close neighbor 学习但是不同的neighbor用不同的学习方法
4. Synaptic Adaptation

- 随着时间的变化, 范围越来越小.

topological preservation(拓扑保留)

- SOM 产生了这个东西.
- 就那个很多人头的图片, 有印象吧?
- 每个图片是一个neuron的weight, 相似的weight在一起.
- manifold

Week13-1

recall unsupervised competitive learning

1. hebbien learning
   - $\triangle w = \eta xy$

LVQ - learning Vector Quantization

- 原先的VQ是用来classification.
- 现在要用VQ在有监督学习上.

- 传统的VQ

  - $\triangle wi = \eta hj,i(x * wj)$, 每次迭代这个
  - 第一步, 先初始化, 随机的
  - 第二部步, 看input 和output多近

- LVQ

  - 用targeted information to change the protype w
  - if the winner belongs to the right class
    - wnew = wold + η(x - w) -if the winner belongs to the wrong class
    - wnew = wold - η(x - w)

- 先用无监督(SOM)找到protype, 然后再用有监督学习, 应用LVQ to change the weight

- 确保Protype能正确表达class

- LVQ用targeted information

- input → SOM → LVQ

Associative Memories

- An associative memory is a *content-addressable structure* that maps a set of input patterns to a set of output patterns.
- that is, memory can be directly accessed by the cotent rather than the physical address in the computer memory.

- two types associative memory:
  - autoassociative
    - retrieves a previously stored pattern that most *closely resembles* the current pattern, 就是之前见过, 再检索一模一样的

  - heteroassociative
    - 对于每个image, 都有一个与之有关联的对应, 但不一定就是原来的image, 有点像联想

  - outer product (x*x^T)
    - 是一个matrix
    - [1,2,3] * [4,5,6]^T = [4,5,6],下一行[8,10,12], 下一行[12,15,18]

Week13-2

The Hopfield NNs

- used for
  - Associated memories, 我们只说这个
  - Combinatorial optimization

- contribution
  - treating a network as a *dynamic system*
  - introduced the notion of *energy function* and attractors into NN research

- 每个neuron的input是别人的output(很奇怪)
  - 也就是说 a *fully connected, symmetrically weighted* network where each node functions both as input and output node.

- single layer
- each node as both input and output units
- 不包括self-connection
- node values are iteratively updated, based on the weighed inputs from all other nodes, until stabilized
- It is usually *initialized* with appropriate weights instead of being trained.

  - **不一样的是, 我们自己design weight**
  - **因为是动态的, 所以我们想要它稳定, 而不是震荡**

- 我们关注discrete Hopfield model. 比如时间是1,2,3,4,5....,而不是1.0,1.1,...
- 我们用sign function, 大于0的就是1, 小于0的就是-1.
- weights:
  - 是对称的, wij=wji
  - 对于自己的weight, 没有! wii=0

- wij = 1/N * ∑ p=1->P (xi)^p*(xj)^p

- *wii=0 is important, 也就是说矩阵对角线上的数字都是0*
    - 怎么减掉? W - I, 其实是下面的

```
X^k = (x1^k, x2^k,...,xn^k)^T, k = 1,2,3,...,p
xi^k =∊{+1,-1}, i =1,2,...,n
W = ∑k=1->p x^k(x^k)^T -pI
也就是
wij=∑k=1->p xi^k*xj^k if i≠j; or =0 if i=j
```

- 这种learning 是一次的, 所以是design出来的而不是学习来的.
- 也就是说, Hopfield mode 的weight 不是random出来的, 是design 出来的
- 检查是否能够胜任
- 昨天的, 提供一个key, 是否能够找到相应的pattern在那个matrix里面
- 这里的weight就是memory, 只是对于神经网络来说是weight
- Wx = [∑k=1->p x^k(x^k)^T -pI]x
    - 这里的x就是一个key或者query, 可以和原图一样, 也可以有噪音的那种
- 假设x≈x^i, x^i是原图, 已经参与matrix的运算
    - 就有Wx ≈ nx^i-px^i = (n-p)x^i
- 这里的x就是一个外部输入, 之前不是说了, 这个网络是接受其他的神经元的输出的, 这里再加一个x当做额外的输入
- the output of a neuron i at time t is:
    - vi(t) = sig(∑wijvi(t-1))
- state update rule
    - Asynchronous mode, 一个一个的更新, 而不是全部一起更新
    - update rule
    - Hi(t+1) = ∑j=1-> n wijvj(t) + Ii
- 怎么判断stable或者是拟合?
    - vk(prevous) = vk(current)

```
例子
a 4 node network, stores 2 patterns(1 1 1 1) and (-1 -1 -1 -1)
weights: wij=1, for i≠j, and wii=o for all i
W = (1 1 1 1)外积(1 1 1 1)^T + (-1 -1 -1 -1) 外积(-1 -1 -1 -1)^T
current input pattern: (1 1 1 -1)
node 2: w21*x1 + w23 *x3 + w24*x4 + I2 = 1 + 1 - 1 + 1 = 2
node 4: w
```

Week14-1

Hopp

- Design connection 在各个neuron之间, 但自己没有self-connection
- recall something(dynamics)
- 确保stable
- why can do this?
- will Hopfield AM converge with any given recall input?
  - by introducing an energy function to this mosdel.
  - 只要证明这个能量方程式单调递减的, 就说明稳定

- $E = -1/2 * \sum_{i=1 \to n} \sum_{i=1 \to n} w_{ij} v_i v_j - \sum_{i=1 \to n} I_i v_i$
  - $v_i = sgn(\sum_{j=1 \to N, j \neq i} w_j i v_i)$

- stable states(attractor) : the number of state = 2^(number of neuron)

Storage capacity

- C = 0.15n, n是neuron的个数

Review

- Perceptron learning
  - △w = learning rate•(target - output)•input

Hint

- PPT **most important**
- Tutorial **important**
- Lab **not required**, 有些问题用matlab问的.
- 今年的试卷: 4个大问题
  1. Answer questions: 5 * 5 = 25 分, 5个问题. 不需要写很长的答案
  2. Solve the following problems using some sepcific model : 3 + 5 + 10 + 7 = 25, 4个问题, 偏实践.

- 先计算一些东西
- 再include matlab, **SOM**, 在ppt上有相应的代码

3. CNN相关问题, 20分, 分成两部分10+10.
    - part1: 一些小问题
        - model structure
        - size change (image ->feature map), parameters

    - part2: 一些实际问题
        - 不是让你design
        - CNN 和MLP一样吗? 有什么区别?
        - 等...

4. Specific Design 对于现实问题, 30分, 两道题15+15