2022年1月29日 星期六

Mentor session 1

RL: Reinforcement learning

-区别在于你不用一次就给出最好的决策，而是在每一次的reward function的评价中，逐渐训练出，就像人训练动物一样。中间的实现过程并不知道。

Model Building Block：从概率的角度

-Probabilistic Model

- A *probabilistic model* is a set of probability distributions, $p(x|\theta)$.
- We pick the *distribution family* $p(\cdot)$, but don't know the parameter θ.

Example: Model data with a Gaussian distribution $p(x|\theta)$, $\theta = \{\mu, \Sigma\}$.

• p(.)是概率模型，θ是该概率模型的一些参数，x指当前概率模型分布中的一个值。

-The i.i.d. assumption：独立同分布假设

Assume data is *independent and identically distributed (iid)*. This is written

$$x_i \overset{iid}{\sim} p(x|\theta), \quad i = 1, \ldots, n.$$

Writing the density as $p(x|\theta)$, then the *joint* density decomposes as

$$p(x_1, \ldots, x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta).$$

Model Training Block

-极大似然估计：这里的p(·)是概率分布模型

Maximum Likelihood approach

We now need to find θ. *Maximum likelihood* seeks the value of θ that maximizes the likelihood function:

$$\hat{\theta}_{\text{ML}} := \arg\max_{\theta} \; p(x_1, \ldots, x_n|\theta),$$

- This value best explains the data according to the chosen distribution family.

Maximum Likelihood equation

The analytic criterion for this maximum likelihood estimator is:

$$\nabla_\theta \prod_{i=1}^{n} p(x_i|\theta) = 0.$$

- 实际上，我理解的，就是p是一个确定的function，然后θ是之中未确定的参数。x_i是现有的确定的观测值。然后我们的目的是，通过训练得到θ使得p的函数值达到最大。但我们并不关心最大函数值是多少，我们只关心此时的θ是多少。

Recall: Gaussian density in n dimensions

Assume a diagonal covariance matrix $\Sigma = \sigma^2 I$. The density is

$$p(y|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\Big(-\frac{1}{2\sigma^2}(y-\mu)^T(y-\mu)\Big).$$

Maximum likelihood for Gaussian linear regression

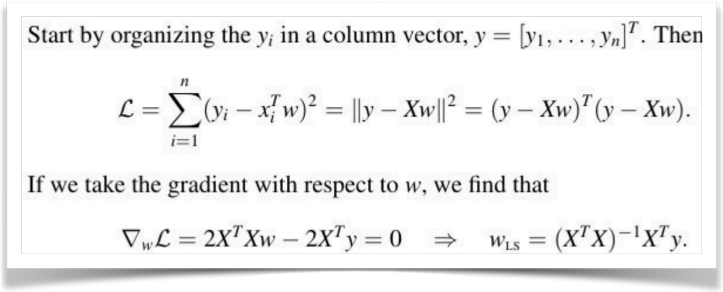
Plug $\mu = Xw$ into the multivariate Gaussian distribution and solve for w using maximum likelihood.

$$\begin{aligned} w_{\text{ML}} &= \arg\max_w \ln p(y|\mu = Xw, \sigma^2) \\ &= \arg\max_w -\frac{1}{2\sigma^2}\|y - Xw\|^2 - \frac{n}{2}\ln(2\pi\sigma^2). \end{aligned}$$

Least squares (LS) and maximum likelihood (ML) share the same solution:

$$\text{LS: } \arg\min_w \|y - Xw\|^2 \quad \Leftrightarrow \quad \text{ML: } \arg\max_w -\frac{1}{2\sigma^2}\|y - Xw\|^2$$

-使用极大似然估计和最小二乘法计算结果是一致的：计算出的w期望相同。



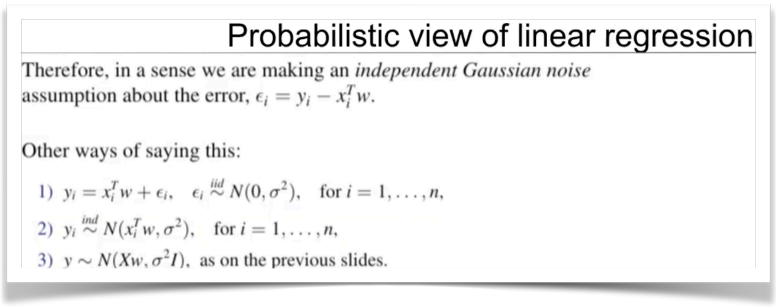
Start by organizing the y_i in a column vector, $y = [y_1, \ldots, y_n]^T$. Then

$$\mathcal{L} = \sum_{i=1}^{n} (y_i - x_i^T w)^2 = \|y - Xw\|^2 = (y - Xw)^T (y - Xw).$$

If we take the gradient with respect to w, we find that

$$\nabla_w \mathcal{L} = 2X^T Xw - 2X^T y = 0 \quad \Rightarrow \quad w_{\text{LS}} = (X^T X)^{-1} X^T y.$$

线性回归的概率角度



Probabilistic view of linear regression

Therefore, in a sense we are making an *independent Gaussian noise* assumption about the error, $\epsilon_i = y_i - x_i^T w$.

Other ways of saying this:

1) $y_i = x_i^T w + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2), \quad \text{for } i = 1, \ldots, n,$
2) $y_i \overset{ind}{\sim} N(x_i^T w, \sigma^2), \quad \text{for } i = 1, \ldots, n,$
3) $y \sim N(Xw, \sigma^2 I)$, as on the previous slides.

-使用极大似然估计和最小二乘法计算结果是一致的：计算出的w期望相同。

- 从概率的角度看线性回归：可以说每一个y_i满足独立的相同的高斯分布；也可以说所有的y满足一个n维的高斯分布。

Matrix：矩阵运算

-*Matrix Cookbook* :https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

-中文博客：https://blog.csdn.net/lijiayu2015/article/details/52611841

- 写的太棒了

-矩阵的行列式

- 行列式的理解：行列式就是关于“面积”的在欧几里得空间推广；
- 线性无关的几何意义：方阵的每一行（或者每一列）都可以看作一个向量，一组矢量的线性相关性本质上，是描述他们所张成的广义平行四边形体积是否为NULL（零）。
  - N个向量线性无关 == 他们所张成的N维体体积不为零

-矩阵的秩

- 有时候，虽然A并不能保持把空间一组最大数目矢量的线性无关性，但它能保证一组更少数目矢量的线性无关性。这个数目往往少于A的维度（或者说，线性空间的维度），这个数目就叫做线性变换A的秩。
  - 例如，一个秩为2的三乘三矩阵A。因为秩小于3，那么任何一个3维六面体经过他的变换后，体积都为零（退化一个面）；但存在一个面积不为零的面，在变换之后还可以是一个非零面积的面。

贝叶斯定理：Bayes rule

Bayes rule lets us quantify what we don't know. Imagine we want to say something about the probability of B given that A happened.

Bayes rule says that the probability of B after knowing A is:

$$\underbrace{P(B|A)}_{posterior} = \underbrace{P(A|B)}_{likelihood}\underbrace{P(B)}_{prior} / \underbrace{P(A)}_{marginal}$$

-这里出现了先验概率和后验概率。

- Let θ be a continuous-valued model parameter.
- Let X be data we possess. Then by Bayes rule,

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{\int p(X|\theta)p(\theta)d\theta} = \frac{p(X|\theta)p(\theta)}{p(X)}$$

In this equation,

- $p(X|\theta)$ is the likelihood, known from the model definition.
- $p(\theta)$ is a prior distribution that we define.
- Given these two, we can (in principle) calculate $p(\theta|X)$.

- 从这个角度看之前线性回归的参数估计。假设需要估计的参数服从一个先验概率分布；

Mentor session 2

MAP: Maximum a Posteriori

The likelihood model is $y \sim N(Xw, \sigma^2 I)$. What about a prior for w?

Let us assume that the prior for w is Gaussian, $w \sim N(0, \lambda^{-1} I)$. Then

$$p(w) = \left(\frac{\lambda}{2\pi}\right)^{\frac{d}{2}} e^{-\frac{\lambda}{2} w^T w}.$$

We can now try to find a w that satisfies both the data likelihood, and our prior conditions about w.

-条件概率密度公式（会用到）

$$P(A \mid B) = \frac{P(AB)}{P(B)}$$

MAP using our defined prior gives:

$$\begin{aligned} w_{\text{MAP}} &= \arg\max_w \ln p(y|w, X) + \ln p(w) \\ &= \arg\max_w -\frac{1}{2\sigma^2}(y - Xw)^T(y - Xw) - \frac{\lambda}{2} w^T w + \text{const.} \end{aligned}$$

Calling this objective $\mathcal{L}$, then as before we find w such that

$$\nabla_w \mathcal{L} = \frac{1}{\sigma^2} X^T y - \frac{1}{\sigma^2} X^T X w - \lambda w = 0$$

- The solution is $w_{\text{MAP}} = (\lambda \sigma^2 I + X^T X)^{-1} X^T y$.

区别于LS和ML中的w估计值：

$$w_{\text{LS}} = (X^T X)^{-1} X^T y.$$

-MAP方法相对于的好处是什么?

在贝叶斯统计学中，“最大后验概率估计”是后验概率分布的众数。利用最大后验概率估计可以获得对实验数据中无法直接观察到的量的点估计。它与最大似然估计中的经典方法有密切关系，但是它使用了一个增广的优化目标，进一步考虑了被估计量的先验概率分布。所以最大后验概率估计可以看作是规则化（regularization）的最大似然估计。

Model

Have vector $y \in \mathbb{R}^n$ and covariates matrix $X \in \mathbb{R}^{n \times d}$. The ith row of y and X correspond to the ith observation (y_i, x_i).

In a Bayesian setting, we model this data as:

$$\textbf{Likelihood}: \quad y \sim N(Xw, \sigma^2 I)$$
$$\textbf{Prior}: \quad w \sim N(0, \lambda^{-1} I)$$

The unknown model variable is $w \in \mathbb{R}^d$.

- The "likelihood model" says how well the observed data agrees with w
- The "model prior" is our prior belief (or constraints) on w.

This is called Bayesian linear regression because we have defined a prior on the unknown parameter and will try to learn its posterior.

Bayes model: our class only focus on its regression function

-Bayes Inference:

Point estimates

w_{MAP} and w_{ML} are referred to as *point estimates* of the model parameters.

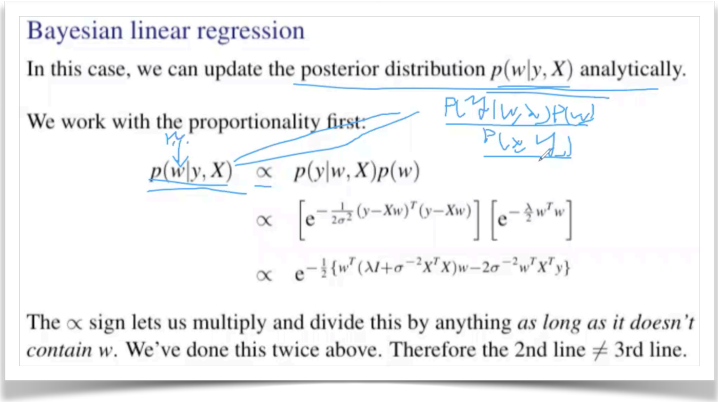
They find a specific value (point) of the vector w that maximizes an objective function — the posterior (MAP) or likelihood (ML).

- **ML**: Only considers the data model: $p(y|w, X)$.
- **MAP**: Takes into account model prior: $p(y, w|X) = p(y|w, X)p(w)$.

Bayesian inference

Bayesian inference goes one step further by characterizing uncertainty about the values in w using Bayes rule.

- 前面讲的ML和MAP方法都是点估计
  - 就是可能w是一个vector，但是我们只能估计出w的一个value；我们可能知道w的mean和variance，但我们不知道w的distribution。大概这个意思，可能有误解。



Bayesian linear regression

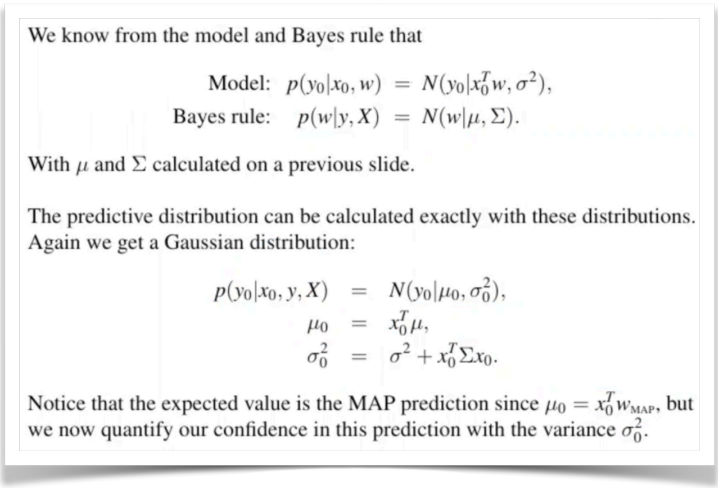
In this case, we can update the posterior distribution $p(w|y, X)$ analytically.

We work with the proportionality first:

$$\begin{aligned} p(w|y, X) &\propto p(y|w, X)p(w) \\ &\propto \left[e^{-\frac{1}{2\sigma^2}(y-Xw)^T(y-Xw)}\right]\left[e^{-\frac{\lambda}{2}w^Tw}\right] \\ &\propto e^{-\frac{1}{2}\{w^T(\lambda I+\sigma^{-2}X^TX)w-2\sigma^{-2}w^TX^Ty\}} \end{aligned}$$

The $\propto$ sign lets us multiply and divide this by anything *as long as it doesn't contain* w. We've done this twice above. Therefore the 2nd line $\neq$ 3rd line.

- 通过前面推到过的，通过Bayes Rules，转化后验概率成一个最后的形式。



We know from the model and Bayes rule that

$$\begin{aligned} \text{Model:} \quad p(y_0|x_0, w) &= N(y_0|x_0^Tw, \sigma^2), \\ \text{Bayes rule:} \quad p(w|y, X) &= N(w|\mu, \Sigma). \end{aligned}$$

With μ and Σ calculated on a previous slide.

The predictive distribution can be calculated exactly with these distributions. Again we get a Gaussian distribution:

$$\begin{aligned} p(y_0|x_0, y, X) &= N(y_0|\mu_0, \sigma_0^2), \\ \mu_0 &= x_0^T\mu, \\ \sigma_0^2 &= \sigma^2 + x_0^T\Sigma x_0. \end{aligned}$$

Notice that the expected value is the MAP prediction since $\mu_0 = x_0^Tw_{\text{MAP}}$, but we now quantify our confidence in this prediction with the variance σ_0^2.

- 又通过贝叶斯定理，和凑高斯分布等神操作，可以算出w的分布（就是上面的μ, Σ）；
- 然后就能算出y的分布。

Perceptron: 感知机，for classification

-定义Input, Output, Loss Function.

- perceptron使用随机梯度下降(stochastic gradient descent)的方法
  - 这个随机就体现在，在一次优化中，只对所有w中的一部分计算L然后优化，再重复多次
  - $w' \leftarrow w - \eta\nabla_w\mathcal{L},$

Logistic Regression

-用一个二分类问题做例子

Logistic link function

We can directly plug in the hyperplane representation for the log odds:

$$\ln \frac{p(y=+1|x)}{p(y=-1|x)} = x^T w + w_0$$

Setting $p(y=-1|x) = 1 - p(y=+1|x)$, solve for $p(y=+1|x)$ to find

$$p(y=+1|x) = \frac{\exp\{x^T w + w_0\}}{1+\exp\{x^T w + w_0\}} = \sigma(x^T w + w_0).$$

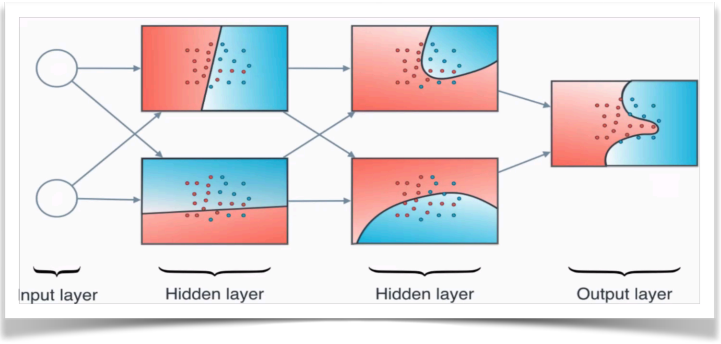
- This is called the *sigmoid function*.
- We have chosen $x^T w + w_0$ as the *link function* for the log odds.

- 通过Bayes rules 又可以变变变把原本的对数差形式转化为关于w的线性方程。
-

Boosting: applied for decision trees

Mentor session 3

Why deep learning?



ReLU, Rectified linear unit

-pros

- better performance, simple

-cons

- not enough to analyze non-linear model

Softmax and Sigmoid

-Softmax（归一化指数函数）主要是返回的和为1 的one-hot-vector，就可以用来实现多分类，因为可以范围最大的那一个，我们把它想象成概率；

- $$f(x_i) = \frac{e^{x_i}}{\sum_{i=0}^{k} e^{x_i}}$$

-Sigmoid（S型函数），很适合二分类；如果用来多分类的话，也可以实现，但结果可能和Softmax不一样，而且所有分类对应的值的和也不是1，就不好解释称概率的含义。

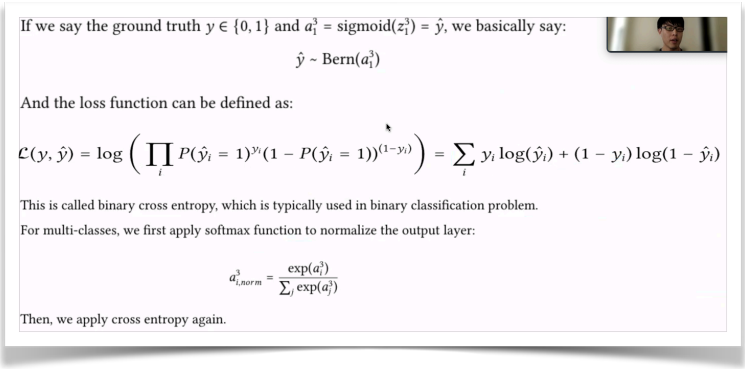
- $$S(x) = \frac{1}{1 + e^{-x}}$$

Activation function

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

- a在这里就是上面说的激活函数，共同计算出了这一层的输出

Binary classification, 用二分类做例子



- 如果ground truth y是一个二分类，也就是0或1，我们的输出$\hat{y}$是两个概率值；
- 于是我们把$\hat{y}$称为服从伯努利分布（特殊的二项分布）；
- 分类的问题使用cross entropy来计算Loss；
- 如果是多分类，就需要使用softmax来正则输出，形成one-hot vector，于是能够代表概率值。

Chain Rules for backward propagation

Optimizer

Universal Approximation Theorem

-优化器失败

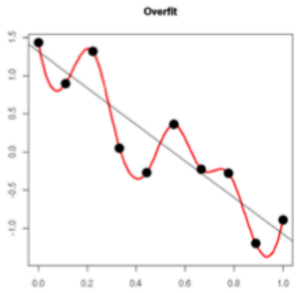
- 更好的初始化参数；
- 避免局部最优解；

-过拟合和欠拟合

- Regularization? why can it solve the overfitting problem??

到目前为止，我们只是解释了L2正则化项有让w“变小”的效果，但是还没解释为什么w“变小”可以防止overfitting？一个所谓“显而易见”的解释就是：更小的权值w，从某种意义上说，表示网络的复杂度更低，对数据的拟合刚刚好（这个法则也叫做奥卡姆剃刀），而在实际应用中，也验证了这一点，L2正则化的效果往往好于未经正则化的效果。当然，对于很多人（包括我）来说，这个解释似乎不那么显而易见，所以这里添加一个稍微数学一点的解释（引自知乎）：

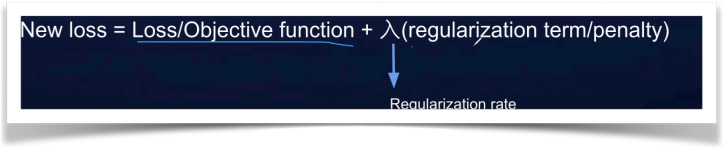
过拟合的时候，拟合函数的系数往往非常大，为什么？如下图所示，过拟合，就是拟合函数需要顾忌每一个点，最终形成的拟合函数波动很大。在某些很小的区间里，函数值的变化很剧烈。这就意味着函数在某些小区间里的导数值（绝对值）非常大，由于自变量值可大可小，所以只有系数足够大，才能保证导数值很大。



- **而正则化是通过约束参数的范数使其不要太大，所以可以在一定程度上减少过拟合情况。**

Regularization，正则化

-在loss function后面加上一个正则化项



- 这里Objective function 是目标函数的意思，不是做除法，这种写法令人误会；
- 所以后面括号里的正则项和penalty同理；

-L1 regularization

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|.$$

- produce sparse solutions（也就是很多接近0的w）
- 它的效果就是让w往0靠，使网络中的权重尽可能为0，也就相当于减小了网络复杂度，防止过拟合；

LASSO?

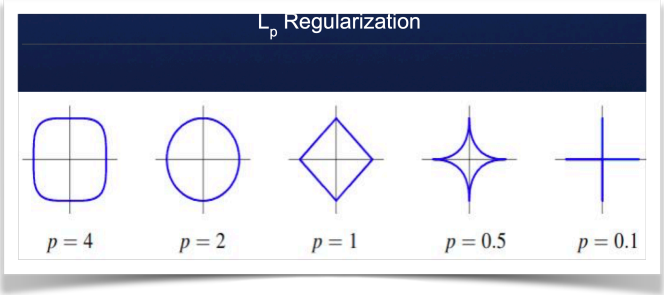
-L2 regularization， 权重衰减weight decay

L2正则化就是在代价函数后面再加上一个正则化项：

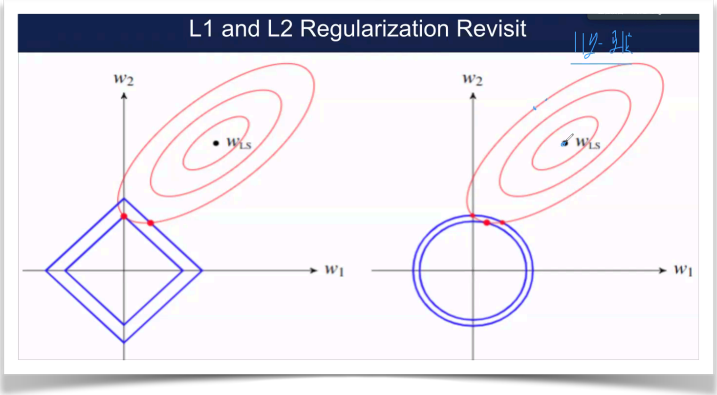
$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2,$$

C0代表原始的代价函数，后面那一项就是L2正则化项，它是这样来的：所有参数w的平方的和，除以训练集的样本大小\n。λ就是正则项系数，权衡正则项与C0项的比重。另外还有一个系数1/2，1/2经常会看到，主要是为了后面求导的结果方便，后面那一项求导会产生一个2，与1/2相乘刚好凑整。

- 通过反向传播求导可以知道，L2正则化项并不会对bias（w_0）有什么影响，只是对权重w们有影响。它的实际效果是减小w的值，所以称它是权重衰减weight decay（虽然事实上可能不会是减小，因为有无关的截距项）；



-



-

- 一个蓝色闭合曲线上的，拥有相同的penalty，我们选择一个最小的；一个红色闭合曲线上的，拥有相同的loss（应该是这样理解的吧！）；

Mentor session 4

Introduction to Deep Learning

Dropout

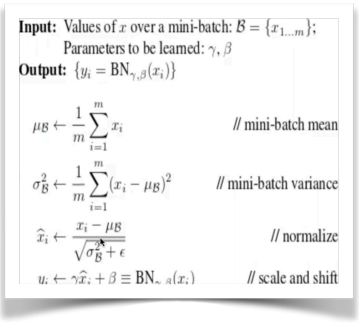
-每一次训练mini-batch 都会随机按设定的比率随机drop隐藏单元，所以每一次小批量训练的时候都会是一个随机的sub-network；最后进行infernece 的时候将会使用所有的参数。

-对于一个神经元来说，在training 的时候按比率出现，在test 的时候总出现；

-Dropout能避免overfitting，但是不回加快训练速度；

Batch Normalization, BN

-Why not use data normalization to every layer?



-因为你不能单因为一个feature 的数值比较大就觉得它比别的feature 重要；这个方法可以be applied to every layer;

-在训练过程中，pop_mean是动态更新的， 指数移动平均法，新进入的batch数据将会对其有最大的影响，计算公式为：

$$pop\_mean = pop\_mean * decay + batch\_mean * (1 - decay)$$

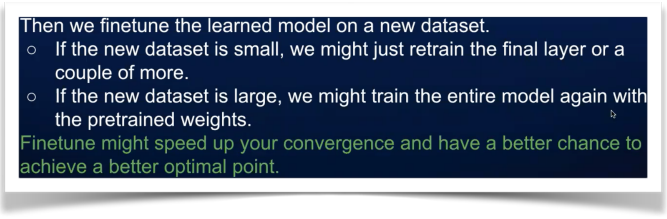
-在较小的网络中，BN层可能会降低模型的表现；
-BN is very useful in very deep NN as it will speed up the training.

-waste norm and layer norm, you can google it said the mentor.

Early Stopping

-stop learning before test loss ascend so we can **avoid overfitting**.

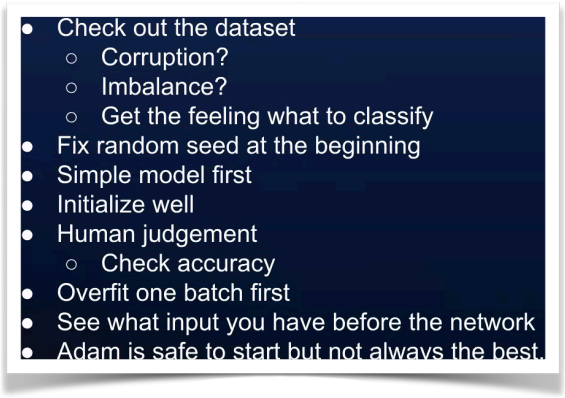
Finetune (for transfer learning)



Data Argumentation 数据增强（扩大）

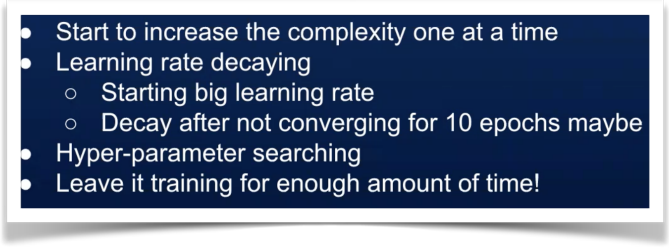
-Generate more data so that data argumentation can **prevent the model from overfitting and make the model robust**.

Mentor's personal notes



-Overfit one batch first:

- 先过拟合你的模型，看看过拟合的结果能不能达到performance的预期，达不到的话就是model本身就有问题

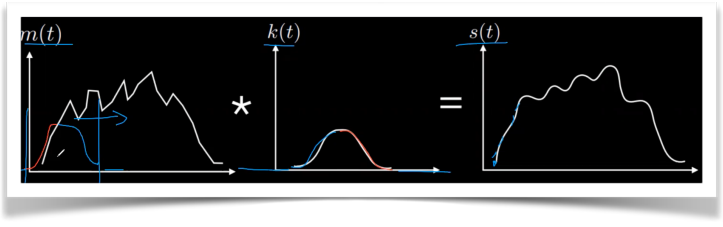


CNN

-Problems of NN:

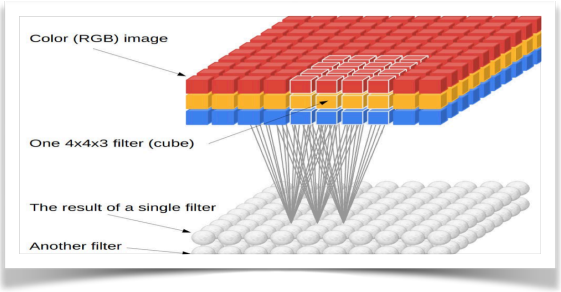
- Too many weights to optimize;
- Sensitive to image noise, dost not capture the structural information;

-Convolution in 1D



- 左图为原始的信号signal，中间的是卷积kernel，得到右图的output signal，很直观的感受就是通过这个卷积核，原始信号被smoothen了；由于我们有不同的卷积核，他们的目的都不同，所以能够找到原始信号不同的feature；

-Convolution in 2D

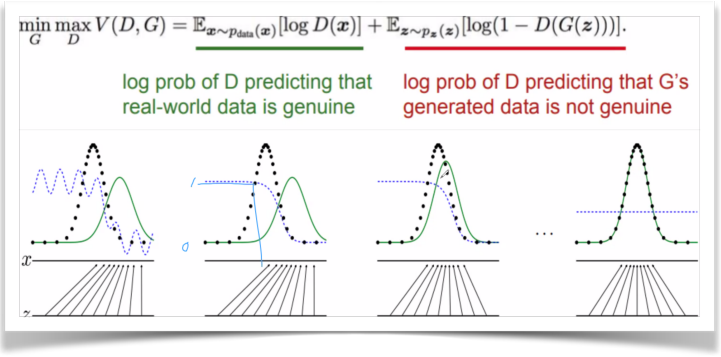


- 这里的filter大小为4x4x3，第三个维度的大小代表的是输入图像的通道的数目；
- 对于输出，一个filter只会有一层的输出；所以输出结果的通道数取决于，filter的个数；
  - 不同的filter / kernel， 会在全局寻找不同的特征；

Mentor Session 5

GAN

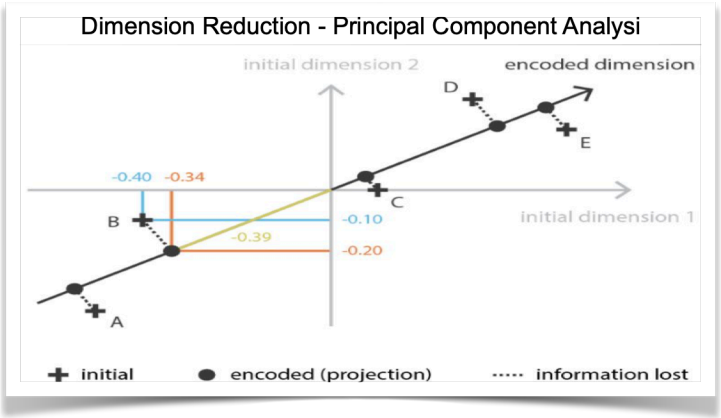
-Loss function



- 绿色线是G model，蓝色虚线是D model，一开始D model能分别出G model产生的fake data，但随着训练，G model越来越接近read data，D model将不能再分辨出。
- 这个过程中，又训练D的目的是什么呢?

Dimension Reduction

-PCA, Principal Component Analysis

- 

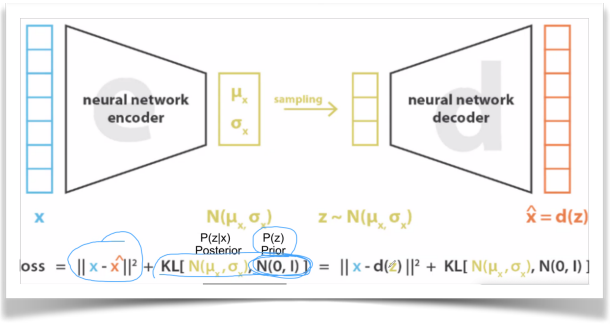
- $$Q = \arg\min_{Q} \sum_{i=1}^{n} \|x_i - QQ^T x_i\|^2, \quad \text{subject to } Q^T Q = I,$$
  - Q 是PCA中的变形矩阵；

Auto-Encoder

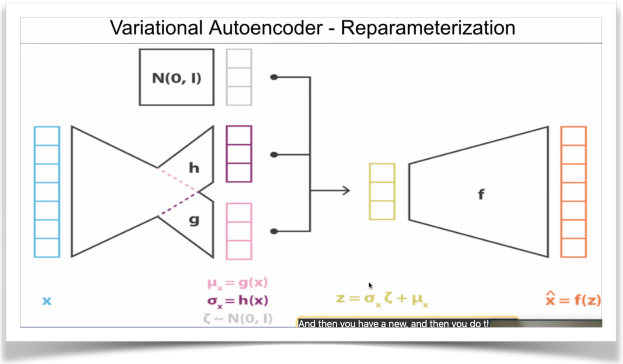
-Dimension Reduction

-Variational AE?

- learn the distribution of the data



- Loss这里的KL部分可以理解为我们在L2正则化中的Penalty term，为了让我们学习到的分布尽可能接近标准正态分布，估计能加快训练?



- 实际在模型中我们是学习到的是g(x) 和h(x)， 然后每次从相同的标准正态分布中取样；
- if so, how can we regularize our distribution to N(1,0)?

-Deepfake

- 我们可以share一个encoder，但是需要使用不同decoder for 不同的classes;
- ???

Markov Model

RNN, Recurrent NN

-motivation or problems

- can not parallelize the computation because calculation at t depends on the result from t-1;

- information lost in the long run even in LSTM;
- Gradient loss or explosion makes training even harder.

Attention 注意力机制**

-Attention is doing the same manner as database dose:

-

GRU, Gated Recurrent Unit

-reduce the parameters and speed up training

review it by yourself ! about LSTM…