

<u>Sentence Generation</u>	<u>Chat-bot</u>
Expert trajectory: 床 前 明 月 光	Expert trajectory: input: how are you Output: I am fine
(s_1, a_1) : ("<BOS>", "床")	(s_1, a_1) : ("input, <BOS>", "I")
(s_2, a_2) : ("床", "前")	(s_2, a_2) : ("input, I", "am")
(s_3, a_3) : ("床前", "明")	(s_3, a_3) : ("input, I am", "fine")
⋮	⋮

Maximum likelihood is behavior cloning. Now we have better approach like SeqGAN.

你可以把写句子这件事情，你在写句子的时候，你写下去的每一个 word，你都当成是一个 action，所有的 word 合起来就是一个 episode。

举例来说，sentence generation 里面，你会给机器看很多人类写的文字，那这个人类写的文字，你要让机器学会写诗，那你就给他看唐诗 300 首，这个人类写的文字，其实就是这个 expert 的 demonstration，每一个词汇，其实就是一个 action。

你让机器做 Sentence Generation 的时候，其实就是在 imitate expert 的 trajectory，或是如果 Chat-bot 也是一样，在 Chat-bot 里面你会收集到很多人互动对话的纪录，那一些就是 expert 的 demonstration。

如果我们今天单纯用 Maximum likelihood 这个技术，来 maximize 会得到 likelihood，这个其实就是 behavior cloning，对不对？用我们今天做 behavior cloning，就是看到一个 state，接下来预测，我们会得到，看到一个 state，然后有一个 Ground truth 告诉机器说，什么样的 action 是最好的，在做 likelihood 的时候也是一样，Given sentence 已经产生的部分，接下来 machine 要 predict 说，接下来要写哪一个 word 才是最好的。

所以，Maximum likelihood 对应到 Imitation Learning 里面，就是 behavior cloning。

那我们说光 Maximum likelihood 是不够的，我们想要用 Sequence GAN，其实 Sequence GAN 就是对应到 Inverse Reinforcement Learning，我们刚才已经有讲过说，其实 Inverse Reinforcement Learning，就是一种 GAN 的技术。

你把 Inverse Reinforcement Learning 的技术，放在 Sentence generation，放到 Chat-bot 里面，其实就是 Sequence GAN 跟他的种种的变形。

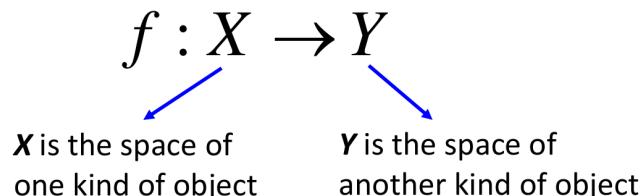
Structured Learning

Structured Learning

什么是 Structured Learning 呢？到目前为止，我们考虑的 input 都是一个 vector，output 也是一个 vector，不管是 SVM 还是 Deep Learning 的时候，我们的 input，output 都是 vector 而已。但是实际上我们要真正面对的问题往往比这个更困难，我们可能需要 input 或者 output 是一个 sequence，我们可能希望 output 是一个 list，是一个 tree，是一个 bounding box 等等。比如 recommendation 里面你希望 output 是一个 list，而不是一个 element。

当然，大原则上我们知道怎么做，我们就是要找一个 function，它的 input 就是我们要的 object，它的 output 就是另外一种 object，只是我们不知道要怎么做。比如说，我们目前学过的 deep learning 的 Neural Network 的架构，你可能不知道怎样 Network 的 input 才是一个 tree structure，output 是另外一个 tree structure。

- We need a more powerful function f
 - Input and output are both objects with structures
 - Object: sequence, list, tree, bounding box ...



In the previous lectures, the input and output are both vectors.

特点：

- 输入输出都是一种带有结构的对象
- 对象：sequence,list,tree,bounding box

Example Application

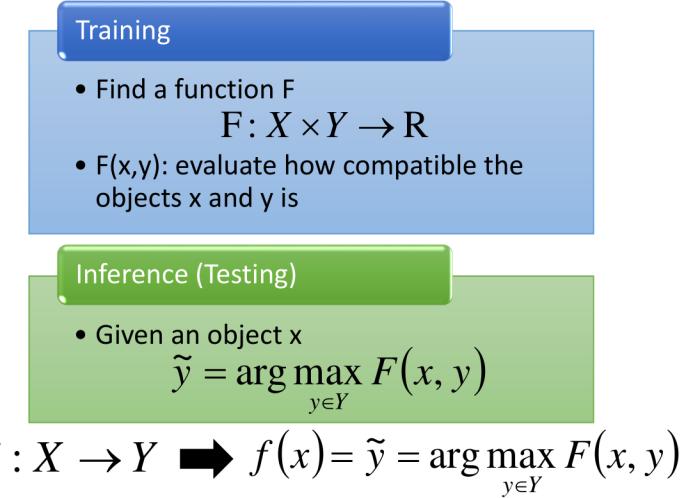
Structured Learning 的应用比比皆是

- Speech recognition(语音辨识)
input 是一个signal sequence, output是另一个text sequence
- Translation(翻译)
input 是一种语言的sequence,output是另外一种语言的sequence
- Syntactic Paring(文法解析)
input 是一个sentence, output 是一个文法解析树
- Object Detection(目标检测)
或者你要做Object detection, input 是一张image, output是一个bounding box。你会用这个bounding box把这个object给框出来。
- Summarization
或者你要做一个Summarization, input是一个大的document, output是一个summary。input 和output都是一个sequence。
- Retrieval
或者你要做一个Retrieval, input是搜寻的关键词, output是搜寻的结果, 是一个webpage的list。

Unified Framework

那么Structured到底要怎么做呢？虽然这个Structured听起来很困难，但是实际上它有一个Unified Framework，统一的框架。

在Training的时候，就是找到function，记为 F ，这个大写 F 的input是 X 跟 Y ，它的output是一个real number。这个大写的 F 它所要做的事情就是衡量出输入 x ，输出 y 都是structure的时候， x 和 y 有多匹配。越匹配， R 值越大。



那testing的时候，给定一个新的 x ，我们去穷举所有的可能的 y ，——带进大写的 F function，看哪一个 y 可以让 F 函数值最大，此时的 \tilde{y} 就是最后的结果，model的output。

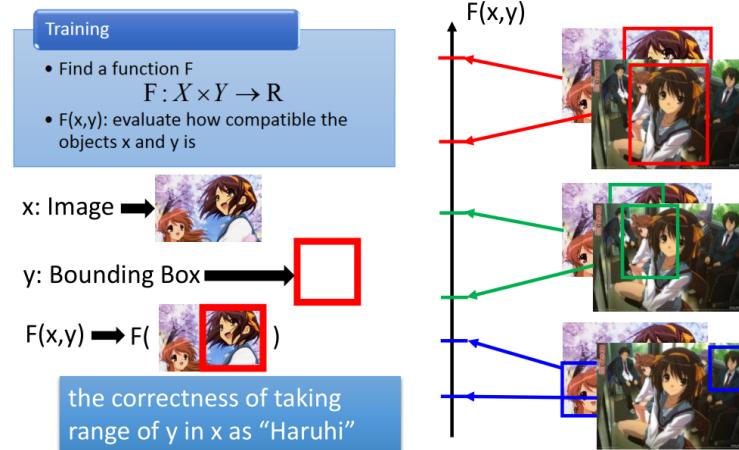
之前我们所要做的事情，是找一个小写的 $f : X \rightarrow Y$ ，可以想象成现在小写的 $f(x) = \tilde{y} = \arg \max_{y \in Y} F(x, y)$ ，这样讲可能比较抽象，我们来举个实际的例子。

Object Detection

用一个方框标识出一张图片中的要它找的object，在我们的task中input是一张image，output是一个Bounding Box。举例来说，我们的目标是要检测出Haruhi。input是一张image，output就是Haruhi所在的位置。可以用于侦测人脸，无人驾驶等等。

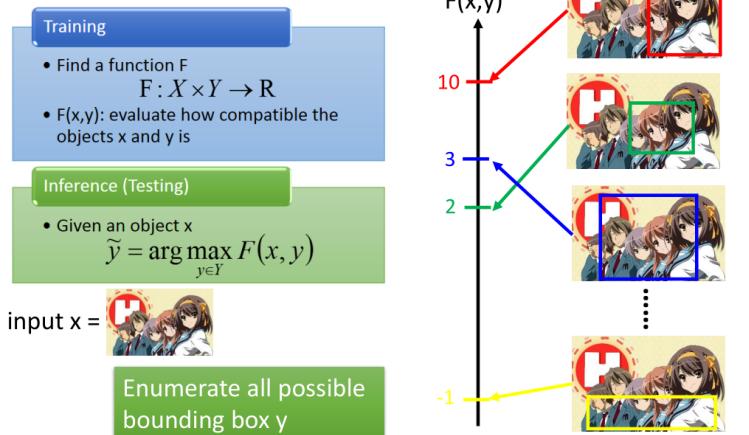
在做object detection的时候，也可以用Deep Learning。事实上，Deep Learning 和Structured Learning是有关系的，这个是我个人的想法，GAN就是 $F(X,Y)$ ，具体的后续再讲。

那么Object Detection是怎么做的呢? input就是一张image, output就是一个Bounding Box, $F(x,y)$ 就是这张image配上这个红色的bounding box, 它们有多匹配。如果是按照Object Detection的例子, 就是它有多正确, 真的吧Haruhi给框出来。所以你会期待, 给这一张图, 如果框得很对, 那么它的分数就会很高。如下图右侧所示。



接下来, testing的时候, 给一张image, 这个 x 是从来没有看过的东西。你穷举所有可能的bounding box, 画在各种不同的地方, 然后看说哪一个bounding box得到的分数最高。红色的最高, 所以红色的就是你的model output。

Unified Framework – Object Detection



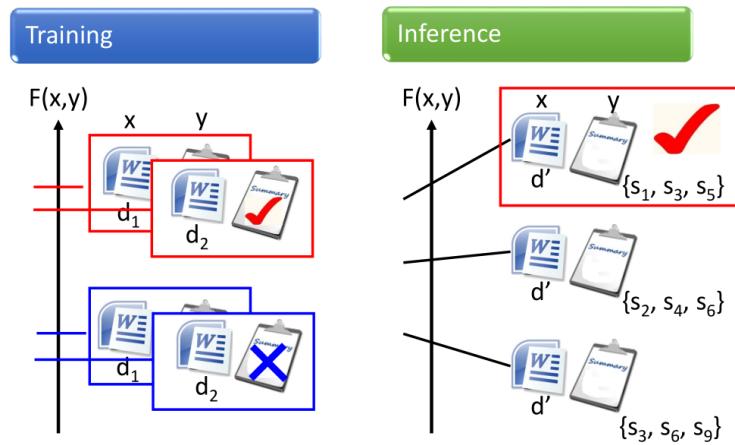
在别的task上其实也是差不多的, 比如

Summarization

input 一个长document, 里面有很多句子。output是一个summary, summary可以从document上取几个句子出来。

那么我们training的时候, 你的这个 $F(x,y)$, 当document和summary配成一对的时候, F 的值就很大, 如果document和不正确的summary配成一对的时候, F 的值就很小, 对每一个training data都这么做。

testing的时候, 就穷举所有可能的summary, 看哪个summary配上的值最大, 它就是model的output。



Retrieval

input 是一个查询集（查询关键字），output是一个webpages的list

Training的时候，我们要知道input一个query时，output是哪些list，才是perfect。以及那些output是不对的，分数就会很低。

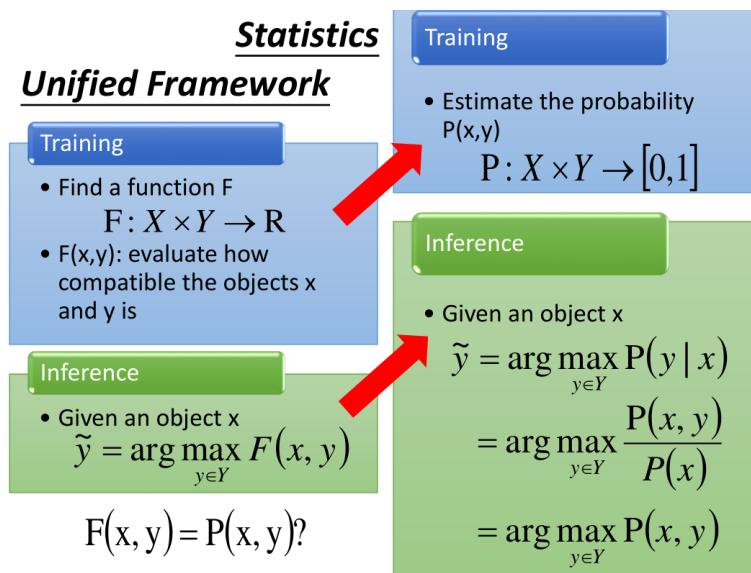
Testing的时候，根据input，穷举所有的可能，看看哪个list的分数最高，就是model的输出。

Statistics

这个Unified Framework或许你听得觉得很怪这样，第一次听到，搞什么东西呀。

那么我换一个说法，我们在Training的时候要estimate x和y的联合概率 $P(x,y)$ ，即x和y一起出现的机率，这样，input就是X和Y，output就是一个介于0到1之间的值。

那我在做testing的时候，给我一个object x，我去计算所有的 $p(y|x)$ ，经过条件概率的推导，哪一个 $p(x,y)$ 的机率最高， \tilde{y} 就是model的输出。



graphical model也是一种structured learning，就是把 $F(x, y)$ 换成机率

用机率表达的方式

- 缺点
 - 机率解释性有限，比如搜寻，我们说查询值和结果共同出现的机率就很怪
 - 机率值限定在[0,1]范围，X和Y都是很大的space，要变成机率可能很多时间都用来做normalization，不一定有必要
- 优点
 - 具有现象意义，机率比较容易描述现象

Energy-based model 也是structured learning

Three Problems

要做这个Framework要解三个问题

Problem 1: Evaluation

第一个问题是，在不同的问题中， $F(x,y)$ 到底应该是什么样的。

- **Evaluation:** What does $F(x,y)$ look like?

- How $F(x,y)$ compute the “compatibility” of objects x and y

Object Detection: $F(x = \text{[Image of a girl]}, y = \text{[Red box]})$

Summarization: $F(x = \text{[Icon of a document]}, y = \text{[Icon of a summary]})$
(a long document) (a short paragraph)

Retrieval: $F(x = \text{“Obama”} \text{ (keyword)}, y = \text{[Search Result]})$



Problem 2: Inference

再来就是那个荒唐的Inference，怎么解“arg max”这个问题。这个Y可是很大的，比如说你要做Object Detection，这个Y是所有可能的bounding box。这件事情做得到吗？

- **Inference:** How to solve the “arg max” problem

$$y = \arg \max_{y \in Y} F(x, y)$$

The space Y can be extremely large!

Object Detection: $Y = \text{All possible bounding box (maybe tractable)}$

Summarization: $Y = \text{All combination of sentence set in a document ...}$

Retrieval: $Y = \text{All possible webpage ranking}$

Problem 3: Training

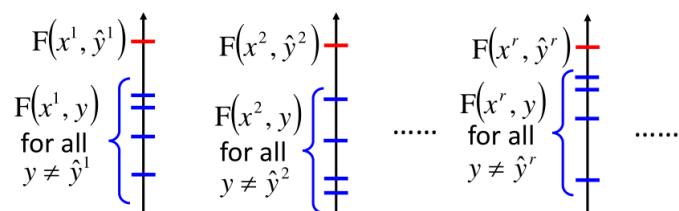
第三个问题是Training，给定training data，如何找到 $F(x, y)$ 。Training Principle是正确的 $F(x, \hat{y})$ 能大过其他的情况，这个Training 应该是可以完成的。

- **Training:** Given training data, how to find $F(x,y)$

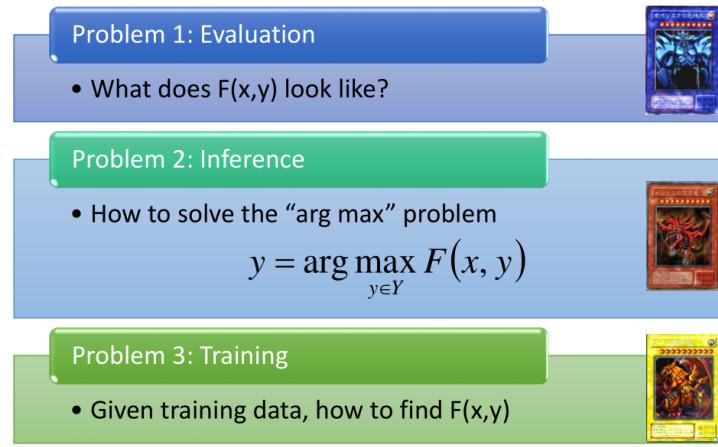
Principle

Training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^r, \hat{y}^r)\}, \dots\}$

We should find $F(x,y)$ such that



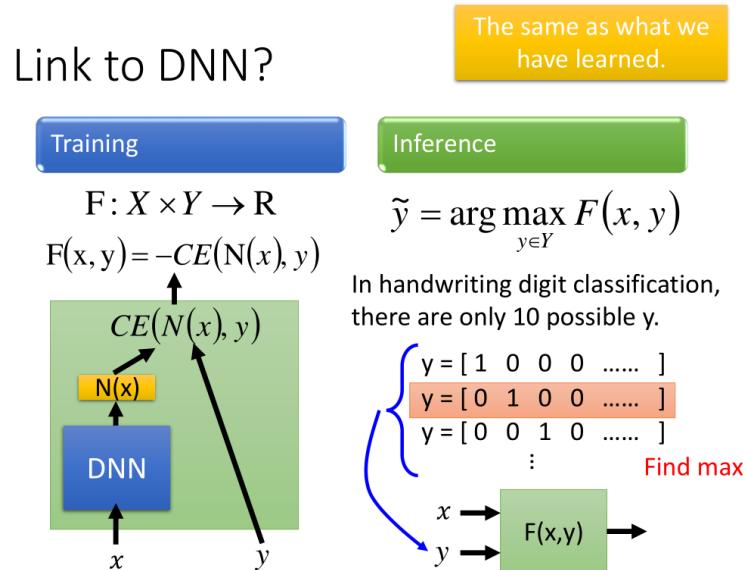
只要你解出这三个问题，你就可以做Structured Learning。



这三个问题可以跟HMM的三个问题联系到一起，也可以跟DNN联系到一起。

Link to DNN?

怎么说呢，比如说我们现在要做手写数字辨识，input一个image，把它分成10类，先把x扔进一个DNN，得到一个 $N(x)$ ，接下来我再input y , y 是一个vector，把这个 y 和 $N(x)$ 算cross entropy， $-CE(N(x), y)$ 就是 $F(x, y)$ 。



接下来，在testing的时候，就是说，我穷所有可能的辨识结果，也就是说10个 y ，每个都带进去这个Function里面，看哪个辨识结果能够让 $F(x, y)$ 最大，它就是我的辨识结果。这个跟我们之前讲的知识是一模一样的。

Structured Linear Model

Solution

假如Problem 1中的 $F(x,y)$ 有一种特殊的形式，那么Problem 3就不是个问题。所以我们要先来讲special form应该长什么样子。

Problem 1

Evaluation: What does $F(x,y)$ look like?

special form必须是Linear，也就是说一个 (x,y) 的pair，首先我用一组特征来描述 (x,y) 的pair，其中 ϕ_i 代表一种特征，也就是说 (x,y) 具有特征 ϕ_1 是 $\phi_1(x, y)$ 这个值，具有特征 ϕ_2 是 $\phi_2(x, y)$ 这个值，等等。然后 $F(x,y)$ 它长得什么样子呢？

$$F(x, y) = w_1\phi_1(x, y) + w_2\phi_2(x, y) + w_3\phi_3(x, y) + \dots$$

向量形式可以写为 $F(x, y) = \mathbf{w} \cdot \phi(x, y)$

- **Evaluation:** What does $F(x,y)$ look like?

Characteristics

$$\mathbf{X} \quad \mathbf{Y} \quad \begin{array}{c} \phi_1(x,y) \\ \phi_2(x,y) \\ \phi_3(x,y) \\ \vdots \end{array} \quad F(x,y) = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w \end{bmatrix} \cdot \begin{bmatrix} \phi_1(x,y) \\ \phi_2(x,y) \\ \phi_3(x,y) \\ \vdots \\ \phi(x,y) \end{bmatrix}$$

$$F(x,y) = w_1 \cdot \phi_1(x,y) + w_2 \cdot \phi_2(x,y) + w_3 \cdot \phi_3(x,y) \dots$$

Learning from data

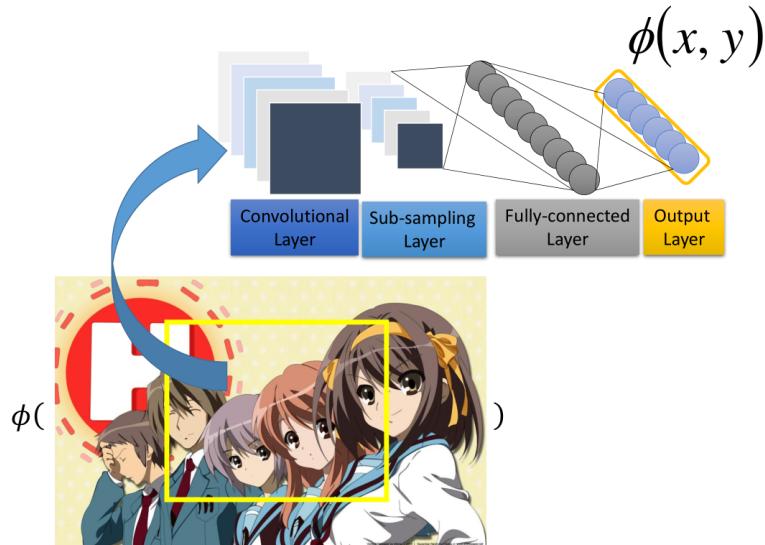
Object Detection

举个object detection的例子，框出Harihu， ϕ 函数可能为红色的pixel在框框里出现的百分比为一个维度，绿色的pixel在框框里出现的百分比为一个维度，蓝色的是一个维度，或者是红色在框框外的百分比是一个维度，等等，或者是框框的大小是一个维度。

现在image中比较state-of-the-art 可能是用visual word，visual word就是图片上的小方框片，每一个方片代表一种pattern，不同颜色代表不同的pattern，就像文章词汇一样。你就可以说在这个框框里面，编号为多少的visual word出现多少个就是一个维度的feature。

这些feature要由人找出来的吗？还是我们直接用一个model来抽呢， $F(x,y)$ 是一个linear function，它的能力有限，没有办法做太厉害的事情。如果你想让它最后performance好的话，那么就需要抽出很好的feature。用人工抽取的话，不见得能找出好的feature。

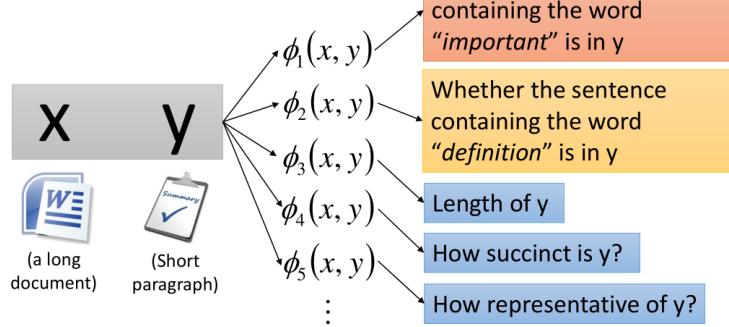
所以如果是在object detection 这个task上面，state-of-the-art 方法，比如你去train一个CNN，你可以把image丢进CNN，然后output一个vector，这个vector能够很好的代表feature信息。现在google在做object detection 的时候其实是用deep network 加上 structured learning 的方法做的，抽feature是用deep learning的方式来做，具体如下图



Summarization

你的 x 是一个document， y 是一个paragraph。你可以定一些feature，比如说 $\phi_1(x,y)$ 表示 y 里面包含“important”这个单词则为1，反之为0，包含的话 y 可能权重会比较大，可能是一个合理的summarization，或者是 $\phi_2(x,y)$ ， y 里面有没有包含“definition”这个单词，或者是 $\phi_3(x,y)$ ， y 的长度，或者你可以定义一个evaluation说 y 的精简程度等等，也可以想办法用deep learning找比较有意义的表示。具体如下图

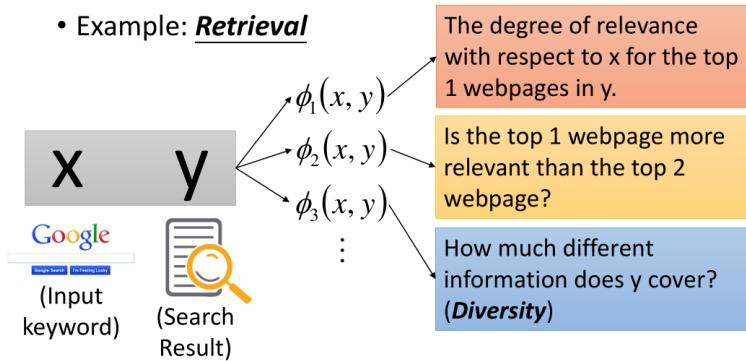
- Evaluation: What does $F(x,y)$ look like?
- Example: Summarization



Retrieval

那比如说是Retrieval，其实也是一样啦。 x 是keyword， y 是搜寻的结果。比如 $\phi_1(x, y)$ 表示 y 第一笔搜寻结果跟 x 的相关度，或者 $\phi_2(x, y)$ 表示 y 的第一笔搜寻结果有没有比第二笔高等等，或者 y 的Diversity的程度是多少，看看我们的搜寻结果是否包含足够的信息。具体如下图

- Evaluation: What does $F(x,y)$ look like?
- Example: Retrieval



Problem 2

如果第一个问题定义好了以后，那第二个问题怎么办呢。 $F(x, y) = w \cdot \phi(x, y)$ 但是我们一样需要去穷举所有的 y ， $y = \arg \max_{y \in Y} w \cdot \phi(x, y)$ 来看哪个 y 可以让 $F(x, y)$ 值最大。

这个怎么办呢？假设这个问题已经被解决了

Problem 3

假装第二个问题已经被解决的情况下，我们就进入第三个问题。

有一堆的Training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^r, \hat{y}^r), \dots\}$ ，我希望找到一个function $F(x, y)$ ，其实是希望找到一个 w ，怎么找到这个 w 使得以下条件被满足：

对所有的training data而言，希望正确的 $w \cdot \phi(x^r, \hat{y}^r)$ 应该大于其他的任何 $w \cdot \phi(x^r, y)$ 。

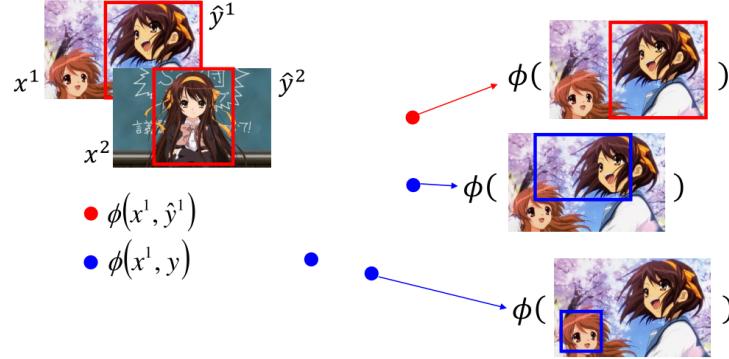
- Training: Given training data, how to learn $F(x,y)$
- $F(x,y) = w \cdot \phi(x,y)$, so what we have to learn is w

Training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^r, \hat{y}^r), \dots\}$

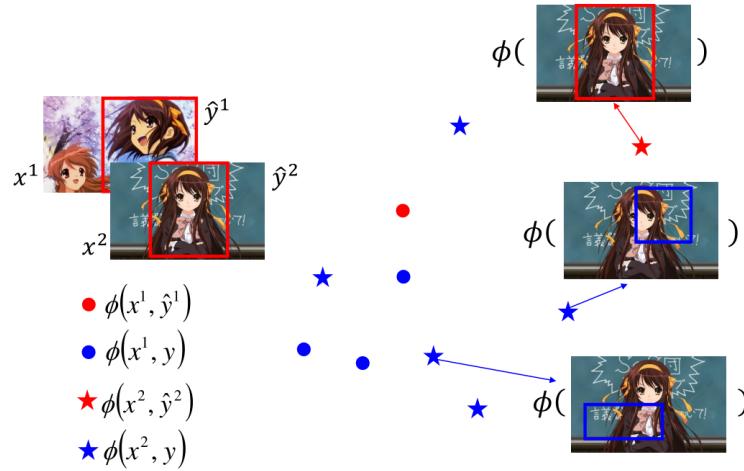
We should find w such that

$$\begin{aligned} & \forall r \text{ (All training examples)} \\ & \forall y \in Y - \{\hat{y}^r\} \text{ (All incorrect label for r-th example)} \\ & w \cdot \phi(x^r, \hat{y}^r) > w \cdot \phi(x^r, y) \end{aligned}$$

用比较具体的例子来说明，假设我现在要做的object detection，我们收集了一张image x^1 ，然后呢，知道 x^1 所对应的 \hat{y}^1 ，我们又收集了另外一张图片，对应的框框也标出。对于第一张图，我们假设 (x^1, \hat{y}^1) 所形成的feature是红色 $\phi(x^1, \hat{y}^1)$ 这个点，其他的 y 跟 x^1 所形成的是蓝色的点。红色的点只有一个，蓝色的点有好多好多。

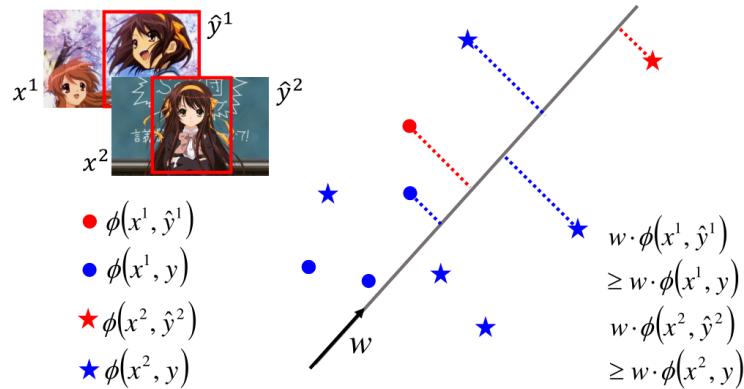


假设 (x^2, \hat{y}^2) 所形成的feature是红色的星星， x^2 与其他的 y 所形成的是蓝色的星星。可以想象，红色的星星只有一个，蓝色的星星有无数个。把它们画在图上，假设它们是如下图所示位置



我们所要达到的任务是，希望找到一个 w ，那这个 w 可以做到什么事呢？我们把这上面的每个点，红色的星星，红色的圈圈，成千上万的蓝色圈圈和蓝色星星通通拿去和 w 做inner cdot后，我得到的结果是红色星星所得到的大于所有蓝色星星，红色的圈圈大过于所有红色的圈圈所得到的值。

不同形状之间我们就不比较。圈圈自己跟圈圈比，星星自己跟星星比。做的事情就是这样子，也就是说我希望正确的答案结果大于错误的答案结果，即 $w \cdot \phi(x^1, \hat{y}^1) \geq w \cdot \phi(x^1, y^1), w \cdot \phi(x^2, \hat{y}^2) \geq w \cdot \phi(x^2, y^2)$ 。



你可能会觉得这个问题会不会很难，蓝色的点有成千上万，我们有办法找到这样的 w 吗？这个问题没有我们想象中的那么难，以下我们提供一个演算法。

Algorithm

输入: 训练数据 $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^r, \hat{y}^r), \dots\}$

输出: 权重向量 w

假设我刚才说的那个要让红色的大于蓝色的vector, 只要它存在, 用这个演算法可以找到答案。

这个演算法是长什么样子呢? 这个演算法的input就是我们的training data, output就是要找到一个vector w , 这个vector w 要满足我们之前所说的特性。

一开始, 我们先initialize $w = 0$, 然后开始跑一个循环, 这个循环里面, 每次我们都取出一笔training data (x^r, \hat{y}^r) , 然后我们去找一个 \tilde{y}^r , 它可以使 $w \cdot (x^r, y)$ 的值最大, 那么这个事情要怎么做呢?

这个问题其实就是Problem 2, 我们刚刚假设这个问题已经解决了的, 如果找出来的 \tilde{y}^r 不是正确答案, 即 $\tilde{y}^r \neq \hat{y}^r$, 代表这个 w 不是我要的, 就要把这个 w 改一下。

怎么改呢? 把 $\phi(x^r, \hat{y}^r)$ 计算出来, 把 $\phi(x^r, \tilde{y}^r)$ 也计算出来, 两者相减在加到 w 上, update w 。

有新的 w 后, 再去取一个新的example, 然后重新算一次max, 如果算出来不对再update, 步骤一直下去, 如果我们要找的 w 是存在的, 那么最终就会停止。

- **Input:** training data set $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^r, \hat{y}^r), \dots\}$
- **Output:** weight vector w
- **Algorithm:** Initialize $w = 0$
 - do
 - For each pair of training example (x^r, \hat{y}^r)
 - Find the label \tilde{y}^r maximizing $w \cdot \phi(x^r, y)$

$$\tilde{y}^r = \arg \max_{y \in Y} w \cdot \phi(x^r, y) \text{ (question 2)}$$
 - If $\tilde{y}^r \neq \hat{y}^r$, update w

$$w \rightarrow w + \phi(x^r, \hat{y}^r) - \phi(x^r, \tilde{y}^r)$$
 - until w is not updated \rightarrow We are done!

这个算法有没有觉得很熟悉呢? 这就是perceptron algorithm。perceptron 做的是二元分类, 其实也是structured learning的一个特例, 它们的证明几乎是一样的。

举个例子来说明一下, 刚才那个演算法是怎么运作的。

我们的目标是要找到一个 w , 它可以让红色星星大过蓝色星星, 红色圈圈大过蓝色圈圈, 假设这个 w 是存在的。首先我们假设 $w = 0$, 然后我们随便pick 一个example (x^1, \hat{y}^1) , 根据手上的data 和 w 去看哪一个 \tilde{y}^1 使得 $w \cdot \phi(x^1, y)$ 的值最大。

现在 $w = 0$, 不管是谁, 所算出来的值都为0, 所以结果值都是一样的。那么没关系, 我们随机选一个 y 当做 \tilde{y}^1 就可以。我们假设选了下图的点作为 \tilde{y}^1 , 选出来的 $\tilde{y}^1 \neq \hat{y}^1$, 对 w 进行调整, 把 $\phi(x^r, \hat{y}^r)$ 值减掉 $\phi(x^r, \tilde{y}^r)$ 的值再和 w 加起来, 更新 w

$$w \rightarrow w + \phi(x^1, \hat{y}^1) - \phi(x^1, \tilde{y}^1)$$

Algorithm - Example

Initialize $w = 0$

pick (x^1, \hat{y}^1)

$$\tilde{y}^1 = \arg \max_{y \in Y} w \cdot \phi(x^1, y)$$

If $\tilde{y}^1 \neq \hat{y}^1$, update w

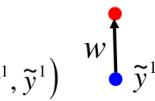
$$w \rightarrow w + \phi(x^1, \hat{y}^1) - \phi(x^1, \tilde{y}^1)$$

• $\phi(x^1, \hat{y}^1)$

• $\phi(x^1, y)$

★ $\phi(x^2, \hat{y}^2)$

★ $\phi(x^2, y)$



Because $w=0$ at this time, $\phi(x^1, y)$ always 0

Random pick one point as \tilde{y}^r

我们就可以获取到第一个 w , 第二步呢, 我们就在选一个example (x^2, \hat{y}^2) , 穷举所有可能的 y , 计算 $w \cdot \phi(x^2, y)$, 找出值最大时对应的 y , 假设选出下图的 \tilde{y}^2 , 发现不等于 \hat{y}^2 , 按照公式 $w \rightarrow w + \phi(x^2, \hat{y}^2) - \phi(x^2, \tilde{y}^2)$ 更新 w , 得到一个新的 w 。

Algorithm - Example

pick (x^2, \hat{y}^2)

$$\tilde{y}^2 = \arg \max_{y \in Y} w \cdot \phi(x^2, y)$$

If $\tilde{y}^2 \neq \hat{y}^2$, update w

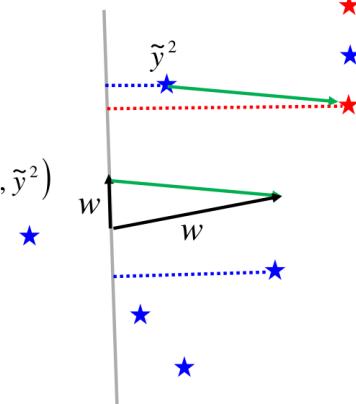
$$w \rightarrow w + \phi(x^2, \hat{y}^2) - \phi(x^2, \tilde{y}^2)$$

• $\phi(x^1, \hat{y}^1)$

• $\phi(x^1, y)$

★ $\phi(x^2, \hat{y}^2)$

★ $\phi(x^2, y)$



然后再取出 (x^1, \hat{y}^1) , 得到 $\tilde{y}^1 = \hat{y}^1$, 对于第一笔就不用更新。再测试第二笔data, 发现 $\tilde{y}^1 = \hat{y}^2$, w 也不用更新, 等等。看过所有data后, 发现 w 不再更新, 就停止整个training。所找出的 w 可以让 $\tilde{y}^r = \hat{y}^r$ 。

Algorithm - Example

pick (x^1, \hat{y}^1) again

$$\tilde{y}^1 = \arg \max_{y \in Y} w \cdot \phi(x^1, y)$$

$$\tilde{y}^1 = \hat{y}^1 \rightarrow \text{do not update } w$$

• $\phi(x^1, \hat{y}^1)$

• $\phi(x^1, y)$

★ $\phi(x^2, \hat{y}^2)$

★ $\phi(x^2, y)$



pick (x^2, \hat{y}^2) again

$$\tilde{y}^2 = \arg \max_{y \in Y} w \cdot \phi(x^2, y)$$

$$\tilde{y}^2 = \hat{y}^2 \rightarrow \text{do not update } w$$

$w \cdot \phi(x^1, \hat{y}^1) \geq w \cdot \phi(x^1, y)$

$w \cdot \phi(x^2, \hat{y}^2) \geq w \cdot \phi(x^2, y)$

So we are done

下一节会证明这个演算法的收敛性, 即演算法会结束。

Structured SVM

结构化学习要解决的问题，即需要找到一个强有力的函数 f

$$f : X \rightarrow Y$$

- 1. 输入和输出都是结构化的对象；
- 2. 对象可以为：sequence(序列), list(列表), tree(树结构), bounding box(包围框)，等等

其中，**X**是一种对象的空间表示，**Y**是另一种对象的空间表示。

这些问题有一个Unified Framework，只有两步

- 第一步：训练
 - 寻找一个函数 F , input是x和y, output是一个real number
- 第二步：推理 or 测试
 - 即给定任意一个x, 穷举所有的y, 将 (x, y) 带入F, 找出最适当的y作为系统的输出。

$$\tilde{y} = \arg \max_{y \in Y} F(x, y)$$

虽然这个架构看起来很简单，但是想要使用的话要回答三个问题

- Q1: 评估
 - **What** does $F(x, y)$ look like?
- Q2: 推理
 - **How** to solve the “arg max” problem, y 的可能性很多，穷举是一件很困难的事，需要找到某些方法解optimization的问题
- Q3: 训练
 - 给定训练数据，如何求解 $F(x, y)$?

Example Task: Object Detection

有比找框框更复杂的问题，比如画出物体轮廓，找出人的动作，甚至不只是image processing的问题，这些问题都可以套用接下来的解法。



Keep in mind that what you will learn today can be applied to other tasks.

- Q1: Evaluation
 - 假设 $F(x, y)$ 是线性的, $F(x, y) = w \cdot \phi(x, y)$, ϕ 是人为定义的规则, w 是在Q3中利用训练数据来学习到的参数。
 - 开放问题：如果 $F(x, y)$ 不是线性，该如何处理？ F 是线性的话会很weak，依赖于复杂的抽取特征的方式 ϕ ，我们希望机器做更复杂的事，减少人类的接入。如果是非线性的话等下的讨论就不成立了，因此目前的讨论多数是基于线性的 F 。
- Q2: Inference

$$\tilde{y} = \arg \max_{y \in Y} w \cdot \phi(x, y)$$

即给定一张图片x，穷举出所有可能的标记框y，对每一对 (x, y) ，用 $w \cdot \phi$ 计算出一对分数最大的 (x, y) ，我们就把对应的y作为输出。

算法的选择取决于task，也取决于 $\phi(x, y)$

- 对于Object Detection可以选择的解决方法有

- Branch & Bound algorithm(分支定界法)
- Selective Search(选择性搜索)
- Sequence Labeling
 - Viterbi Algorithm(维特比译码算法)
- Genetic Algorithm(基因演算)
- 开放问题: What happens if the inference is non exact? 对结果影响会有多大呢? 这件事目前还没有太多讨论。

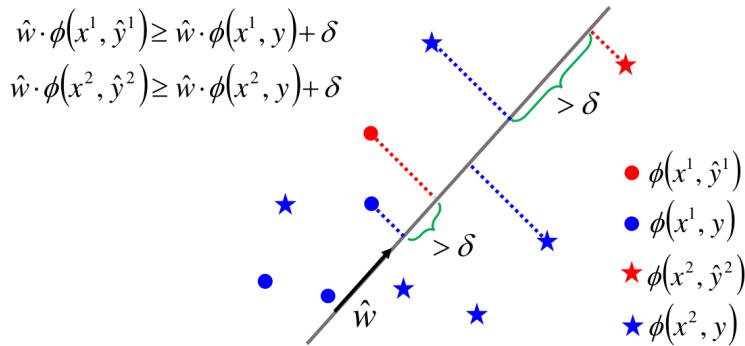
- Q3: Training

- Principle

对所有的training data $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2) \dots, (x^N, \hat{y}^N)\}$ 而言, 希望正确的 $F(x^r, \hat{y}^r)$ 应该大于其他的任何 $F(x^r, y)$ 。
假定我们已经解决了Q1和Q2, 只关注Q3如何处理: 找到最佳的 $F(x, y)$ 。

Assumption: Separable

- There exists a weight vector \hat{w}



Separable: 存在一个权值向量 \hat{w} , 使得:

$$\begin{aligned}\hat{w} \cdot \phi(x^1, \hat{y}^1) &\geq \hat{w} \cdot \phi(x^1, y) + \delta \\ \hat{w} \cdot \phi(x^2, \hat{y}^2) &\geq \hat{w} \cdot \phi(x^2, y) + \delta\end{aligned}$$

红色代表正确的特征点(feature point), 蓝色代表错误的特征点(feature point), 可分性可以理解为, 我们需要找到一个权值向量, 其与 $\phi(x, y)$ 做内积(inner product), 能够让正确的point比蓝色的point的值均大于一个 δ 。

如果可以找到的话, 就可以用以下的演算法找出 w

Structured Perceptron

- Input: training data set $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$
- Output: weight vector w
- Algorithm: Initialize $w = 0$
 - do
 - For each pair of training example (x^n, \hat{y}^n)
 - Find the label \tilde{y}^n maximizing $w \cdot \phi(x^n, y)$

$$\tilde{y}^n = \arg \max_{y \in Y} w \cdot \phi(x^n, y)$$
 (problem 2)
 - If $\tilde{y}^n \neq \hat{y}^n$, update w

$$w \rightarrow w + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$
 - until w is not updated \rightarrow We are done!

输入: 训练数据集

$$\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$$

输出: 可以让data point separate 的 weight vector w

算法: 首先我们假设 $w = 0$, 然后我们随便pick一个example (x^1, \hat{y}^1) , 根据手上的data 和 w 去看哪一个 \tilde{y}^1 使得 $w \cdot \phi(x^1, y)$ 的值最大。假设选出来的 $\tilde{y}^1 \neq \hat{y}^1$, 对 w 进行调整, 把 $\phi(x^r, \hat{y}^r)$ 值减掉 $\phi(x^r, \tilde{y}^r)$ 的值再和 w 加起来, 更新 w . 不断进行iteration, 当对于所有data来说, 找到的 \tilde{y}^n 与 \hat{y}^n 都相等, w 不再更新, 就停止整个training. 所找出的 w 可以让 $\tilde{y}^r = \hat{y}^r$.

问题是这个演算法要花多久的时间才可以收敛, 是否可以轻易的找到一个vector把蓝色的点和红色的点分开?

结论: 在可分情形下, 我们最多只需更新 $(R/\delta)^2$ 次就可以找到 \hat{w} . 其中, δ 为margin(使得误分的点和正确的点能够线性分离), R 为 $\phi(x, y)$ 与 $\phi(x, y')$ 的最大距离, 与y的space无关, 因此蓝色的点非常多也不会影响我们update的次数。

Proof of Termination

一旦有错误产生, w 将会被更新

$$\begin{aligned} w^0 &= 0 \rightarrow w^1 \rightarrow w^2 \rightarrow \dots \rightarrow w^k \rightarrow w^{k+1} \rightarrow \dots \\ w^k &= w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n) \end{aligned}$$

注意: 此处我们仅考虑可分情形

假定存在一个权值向量 \hat{w} 使得对于 $\forall n$ (所有的样本) 、 $\forall y \in Y - \{\hat{y}^n\}$ (对于一个样本的所有不正确的标记)

$$\hat{w} \cdot \phi(x^n, \hat{y}^n) \geq \hat{w} \cdot \phi(x^n, y) + \delta$$

不失一般性, 假设 $\|\hat{w}\| = 1$

证明: 随着 k 的增加 \hat{w} 与 w^k 之间的角度 ρ_k 将会变小, $\cos \rho_k$ 会越来越大

$$\begin{aligned} \cos \rho_k &= \frac{\hat{w}}{\|\hat{w}\|} \cdot \frac{w^k}{\|w^k\|} \\ \hat{w} \cdot w^k &= \hat{w} \cdot (w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)) \\ &= \hat{w} \cdot w^{k-1} + \hat{w} \cdot \phi(x^n, \hat{y}^n) - \hat{w} \cdot \phi(x^n, \tilde{y}^n) \end{aligned}$$

在可分情形下, 有

$$[\hat{w} \cdot \phi(x^n, \hat{y}^n) - \hat{w} \cdot \phi(x^n, \tilde{y}^n)] \geq \delta$$

所以得到

$$\hat{w} \cdot w^k \geq \hat{w} \cdot w^{k-1} + \delta$$

可得:

$$\begin{aligned} \hat{w} \cdot w^1 &\geq \hat{w} \cdot w^0 + \delta \quad \text{and} \quad w^0 = 0 \Rightarrow \hat{w} \cdot w^1 \geq \delta \\ \hat{w} \cdot w^2 &\geq \hat{w} \cdot w^1 + \delta \quad \text{and} \quad \hat{w} \cdot w^1 \geq \delta \Rightarrow \hat{w} \cdot w^2 \geq 2\delta \\ &\dots \\ \hat{w} \cdot w^k &\geq k\delta \end{aligned}$$

分子项不断增加

考虑分母 $\|w^k\|$, w^k 的长度

$$w^k = w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$

则:

$$\begin{aligned} \|w^k\|^2 &= \|w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\|^2 \\ &= \|w^{k-1}\|^2 + \|\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\|^2 + 2w^{k-1} \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)) \end{aligned}$$

其中,

$$\begin{aligned} \|\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\|^2 &> 0 \\ 2w^{k-1} \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)) &< 0 \end{aligned}$$

由于 w 是错误的, 和此时找出的 \tilde{y}^n 内积要大于与正确 \hat{y}^n 的内积, 因此第二个式子是小于零。

我们假设任意两个特征向量之间的距离 $\|\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\|^2$ 小于 R , 则有

$$\|w^k\|^2 \leq \|w^{k-1}\|^2 + R^2$$

于是

$$\begin{aligned}\|w^1\|^2 &\leq \|w^0\|^2 + R^2 = R^2 \\ \|w^2\|^2 &\leq \|w^1\|^2 + R^2 \leq 2R^2 \\ \dots\dots \\ \|w^k\|^2 &\leq kR^2\end{aligned}$$

综上可以得到

$$\hat{w} \cdot w^k \geq k\delta \quad \|w^k\|^2 \leq kR^2$$

则

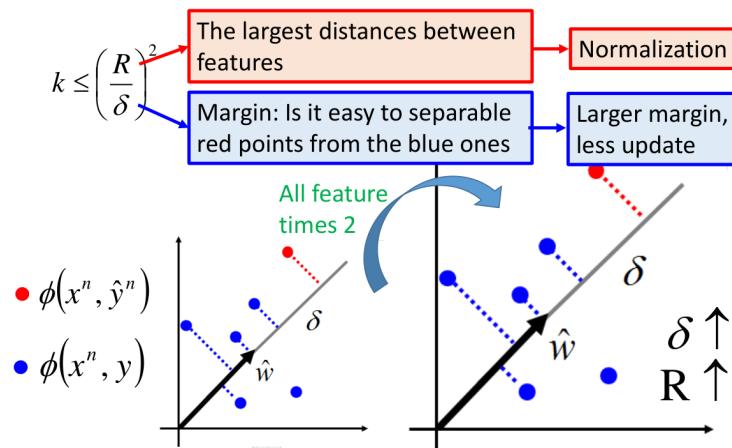
$$\cos \rho_k = \frac{\hat{w}}{\|\hat{w}\|} \cdot \frac{w^k}{\|w^k\|} \geq \frac{k\delta}{\sqrt{kR^2}} = \sqrt{k} \frac{\delta}{R} \leq 1$$

因此随着k的增加, $\cos \rho_k$ 的lower bound也在增加, 并且 $\cos \rho_k \leq 1$

即得到

$$k \leq \left(\frac{R}{\delta}\right)^2.$$

How to make training fast?



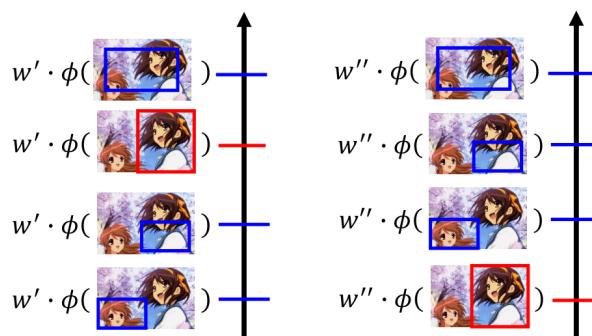
单纯把feature×2, 随着 δ 的增大, R 也会增大, 因此training不会变快

Non-separable Case

虽然可能没有任何一个vector可以让正确和错误答案完全分开, 但是还是可以鉴别出vector的好坏。比如下图左就比右要好。

Non-separable Case Undoubtedly, w' is better than w'' .

- When the data is non-separable, some weights are still better than the others.



Defining Cost Function

定义一个成本函数C来评估w的效果有多差，然后选择w，从而最小化成本函数C。

第n笔data的Cost为，在此w下，与 x^n 最匹配的y的分数减去真实的 \hat{y} 的分数

$$C^n = \max_y [w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

$$C = \sum_{n=1}^N C^n$$

What is the minimum value?

$$C^n \geq 0$$

Other alternatives?

Problem 2中已经计算出了第一名的值是多少，因此用第一名的值减去 \hat{y} 最方便，其他的方案，比如用前三名的值，需要算出前三名的结果才可以

(Stochastic) Gradient Descent

Find w minimizing the cost C

$$C = \sum_{n=1}^N C^n$$

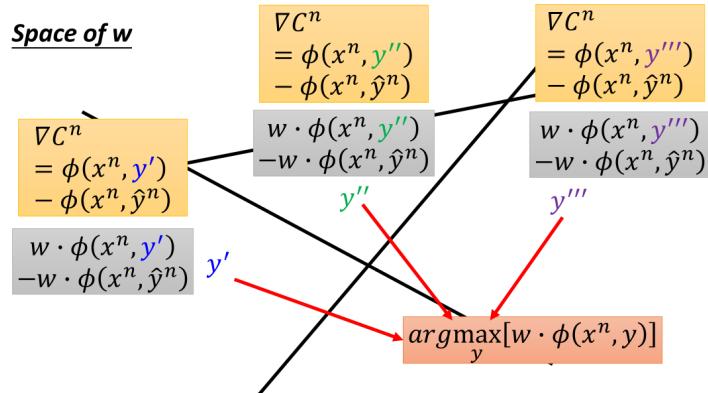
$$C^n = \max_y [w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

我们只需要算出 C^n 的梯度，就可以利用梯度下降法，但是式子中有 \max ，如何求梯度？

$$C^n = \max_y [w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

How to compute ∇C^n ?

When w is different,
the y can be different.



当w不同时，得到的 $y = \arg \max_y [w \cdot \phi(x^n, y)]$ 也会改变；假设w的space被 $y = \arg \max_y [w \cdot \phi(x^n, y)]$ 切割成好几块，得到的 $y = \arg \max_y [w \cdot \phi(x^n, y)]$ 分别等于 y', y'', y'''' ，在边界的地方没有办法微分，但是在每一个region里面都是可以微分的。得到的梯度如图中黄色方框中。

利用(Stochastic) Gradient Descent求解

For $t = 1$ to T : ← Update the parameters T times

Randomly pick a training data $\{x^n, \hat{y}^n\}$ ← stochastic

$\hat{y}^n = \arg \max_y [w \cdot \phi(x^n, y)]$ ← Locate the region

$\nabla C^n = \phi(x^n, \hat{y}^n) - \phi(x^n, \hat{y}^n)$ ← simple

$$w \rightarrow w - \eta \nabla C^n$$

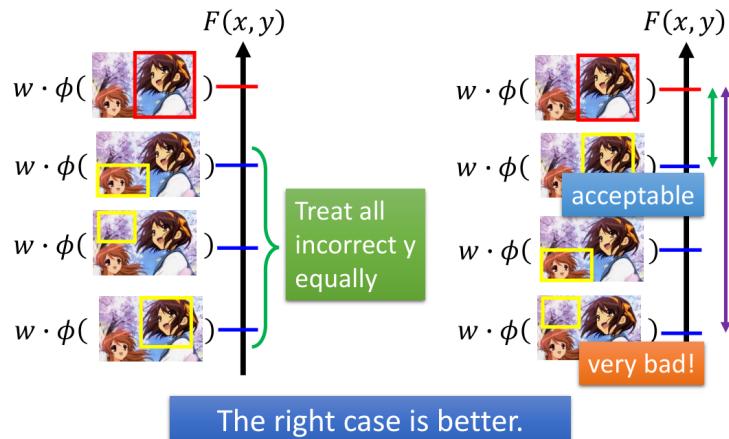
$$= w - \eta [\phi(x^n, \hat{y}^n) - \phi(x^n, \hat{y}^n)]$$

If we set $\eta = 1$, then we are doing structured perceptron.

当学习率设为1时，就转换为structured perceptron。

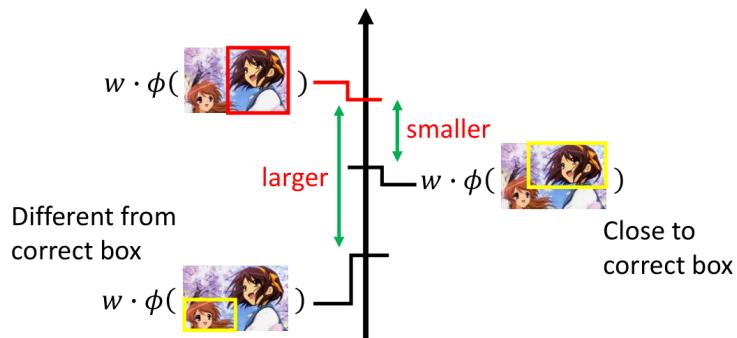
Considering Errors

在刚才，所有错误是视为一样的，然而不同的错误之间是存在差异的，错误可以分为不同的等级，我们在训练时需要考虑进去。比如框在樱花树上分数会特别低，框在凉宫春日脸上，分数会比较高，接近正确的分数也是可以的。如果有一个 w 只知道把正确的摆在第一位；相反另一个 w ，可以按照方框好坏来排序，那learn到的结果是比较安全的，因为分数比较高的和第一名差距没有很大。



Defining Error Function

错误的结果和正确的结果越像，那么分数的差距比较小；相反，差距就比较大。问题是如何衡量这种差异呢？

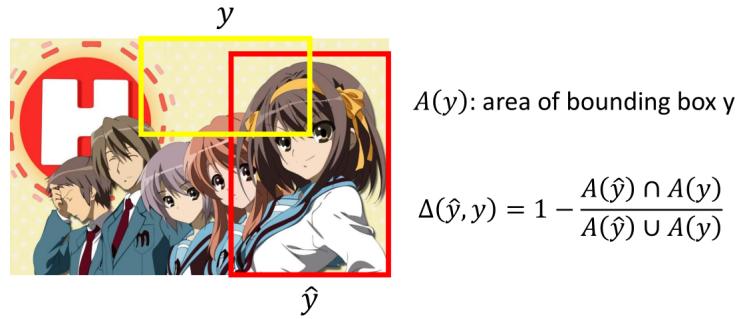


How to measure the difference

\hat{y} (正确的标记)与某一个 y 之间的差距定义为 $\Delta(\hat{y}, y) (> 0)$ ，如果和真实结果相同 $\Delta = 0$ ，具体形式根据任务不同而不同。

在下面的讨论中我们定义为

- $\Delta(\hat{y}, y)$: difference between \hat{y} and y (> 0)



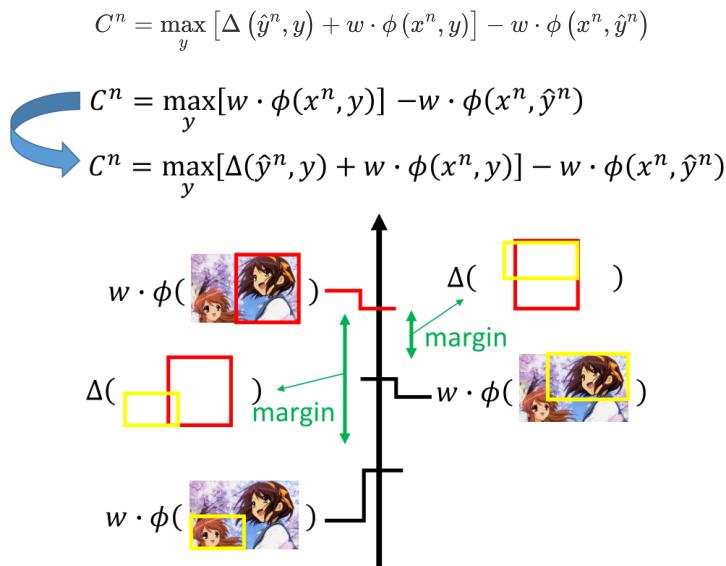
Another Cost Function

修改Cost Function, 本来的Cost是取分数最高的 y 的分数减去 \hat{y} 得到的分数;

我们会把 y 的分数加上 Δ , 这样可以使得当存在与 x^n 最匹配的 y 分数大, margin也大的项时, Cost会很大, 当分数大, Δ 小, 我们才认为他是真正的比较好的。

当 Δ 很大时, 我们希望他的分数很小; 当 Δ 很小时, 即使它的分数高也没有关系。margin越大, 也就说明和真实之间的差距越大, 损失也就越大, 当然你可以定其他的差距式子, 定的好不好可能会影响损失函数的结果。

什么时候Cost最小? 当真实值比最大的 $y+margin$ 的值还要大时, Cost最小。



Gradient Descent

In each iteration, pick a training data $\{x^n, \hat{y}^n\}$

$$\hat{y}^n = \underset{y}{\operatorname{argmax}} [w \cdot \phi(x^n, y)] \quad \underset{y}{\operatorname{argmax}} [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)]$$

Oh no! Problem 2.1

$$\nabla C^n(w) = \phi(x^n, \hat{y}^n) - \phi(x^n, \hat{y}^n)$$

$$w \rightarrow w - \eta [\phi(x^n, \hat{y}^n) - \phi(x^n, \hat{y}^n)]$$

Another Viewpoint

我们也可以从另外一个角度来分析, 最小化新的目标函数, 其实就是最小化训练集里的损失上界, 我们想最小化我们的最大 y 和真实 y 之间的差距本来是这样的, 假设我们的output是 \tilde{y} , 希望minimize C' 。

但是这个很难, 因为 Δ 可能是任何的函数, 比如阶梯状函数, 就不好微分了, 梯度下降法就不好做了, 比如语音识别, 就算 w 有改变, 但是 Δ 不一定就有改变, 可能要到某个点上才可能会出现变化。所以我们就最小化它的上界, 或许没办法让他变小, 至少不会变大。

Another Viewpoint

$$\tilde{y}^n = \arg \max_y w \cdot \phi(x^n, y)$$

- Minimizing the new cost function is minimizing the upper bound of the errors on training set

$$C' = \sum_{n=1}^N \Delta(\hat{y}^n, \tilde{y}^n) \leq C = \sum_{n=1}^N C^n \text{ upper bound}$$

We want to find w minimizing C' (errors)

It is hard!

Because y can be any kind of objects, $\Delta(\cdot, \cdot)$ can be any function

C serves as the surrogate of C'

Proof that $\Delta(\hat{y}^n, \tilde{y}^n) \leq C^n$

那接下来就是证明上面的式子为什么最小化新的代价函数，就是在最小化训练集上误差的上界：

$$C' = \sum_{n=1}^N \Delta(\hat{y}^n, \tilde{y}^n) \leq C = \sum_{n=1}^N C^n$$

只需要证明：

$$\Delta(\hat{y}^n, \tilde{y}^n) \leq C^n$$

$$C^n = \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

Proof that $\Delta(\hat{y}^n, \tilde{y}^n) \leq C^n$

$$\begin{aligned} \Delta(\hat{y}^n, \tilde{y}^n) &\leq \Delta(\hat{y}^n, \tilde{y}^n) + [w \cdot \phi(x^n, \tilde{y}^n) - w \cdot \phi(x^n, \hat{y}^n)] \\ &\quad \tilde{y}^n = \arg \max_y w \cdot \phi(x^n, y) \geq 0 \\ &= [\Delta(\hat{y}^n, \tilde{y}^n) + w \cdot \phi(x^n, \tilde{y}^n)] - w \cdot \phi(x^n, \hat{y}^n) \\ &\leq \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n) \\ &= C^n \end{aligned}$$

More Cost Functions

也可以满足下式

$$\Delta(\hat{y}^n, \tilde{y}^n) \leq C^n$$

- Margin Rescaling(间隔调整)

$$C^n = \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

- Slack Variable Rescaling(松弛变量调整)

$$C^n = \max_y \Delta(\hat{y}^n, y) [1 + w \cdot \phi(x^n, y) - w \cdot \phi(x^n, \hat{y}^n)]$$

Regularization

训练数据和测试数据可以有不同的分布；

如果 w 与 0 比较接近，那么我们就可以最小化误差匹配的影响；

即在原来的基础上，加上一个正则项 $\frac{1}{2} \|w\|^2$, λ 为权衡参数；

$$C = \sum_{n=1}^N C^n \Rightarrow C = \lambda \sum_{n=1}^N C^n + \frac{1}{2} \|w\|^2$$

Training data and testing data can have different distribution.

w close to zero can minimize the influence of mismatch.

Keep the incorrect answer from a margin depending on errors

$$\begin{aligned} C &= \sum_{n=1}^N C^n \quad \longrightarrow \quad C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N C^n \\ C^n &= \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] \\ &\quad - w \cdot \phi(x^n, \hat{y}^n) \end{aligned}$$

Regularization:
Find the w close to zero

每次迭代，选择一个训练数据 $\{x^n, \hat{y}^n\}$

$$C = \sum_{n=1}^N C^n \quad \longrightarrow \quad C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N C^n$$

In each iteration, pick a training data $\{x^n, \hat{y}^n\}$

$$\bar{y}^n = \operatorname{argmax}_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)]$$

$$\nabla C^n = \phi(x^n, \bar{y}^n) - \phi(x^n, \hat{y}^n) + w$$

$$w \rightarrow w - \eta [\phi(x^n, \bar{y}^n) - \phi(x^n, \hat{y}^n)] - \eta w$$

$$= (1 - \eta)w - \eta [\phi(x^n, \bar{y}^n) - \phi(x^n, \hat{y}^n)]$$

Weight decay as in DNN

得到的结果类似于DNN中的weight decay

Structured SVM

$$\begin{aligned} &\text{Find } w \text{ minimizing } C \\ &C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N C^n \\ &C^n = \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n) \\ &C^n + w \cdot \phi(x^n, \hat{y}^n) = \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] \\ &\text{Are they equivalent?} \quad \text{We want to minimize } C \\ &\text{For } \forall y: \\ &C^n + w \cdot \phi(x^n, \hat{y}^n) \geq \Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y) \\ &w \cdot \phi(x^n, \hat{y}^n) - w \cdot \phi(x^n, y) \geq \Delta(\hat{y}^n, y) - C^n \end{aligned}$$

注意：第二个蓝色箭头并不完全等价，当最小化 C^n 时等价。

一般我们将 C^n 用 ε^n 代替之，表示松弛变量，此时条件变成了 Find $w, \varepsilon^1, \dots, \varepsilon^N$ minimizing C

Find w minimizing C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N C^n$$

$$C^n = \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

III

Find $w, \varepsilon^1, \dots, \varepsilon^N$ minimizing C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

For $\forall n$:

For $\forall y$:

$$w \cdot \phi(x^n, \hat{y}^n) - w \cdot \phi(x^n, y) \geq \Delta(\hat{y}^n, y) - \varepsilon^n$$

单独讨论 $y = \hat{y}^n$ 时的情况，得到新的表达式

Find $w, \varepsilon^1, \dots, \varepsilon^N$ minimizing C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

For $\forall n$:

For $\forall y$:

$$w \cdot \phi(x^n, \hat{y}^n) - w \cdot \phi(x^n, y) \geq \Delta(\hat{y}^n, y) - \varepsilon^n$$

For $\forall y \neq \hat{y}^n$:

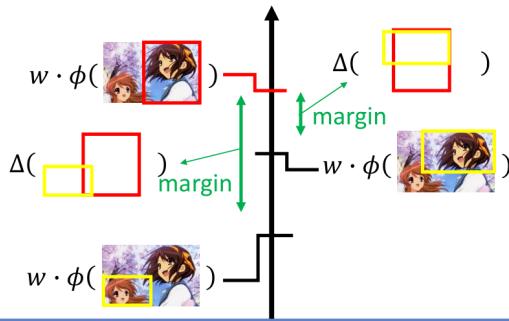
$$w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{y}^n, y) - \varepsilon^n, \quad \varepsilon^n \geq 0$$

If $y = \hat{y}^n$: $w \cdot \phi(x^n, \hat{y}^n) - w \cdot \phi(x^n, \hat{y}^n) \geq \Delta(\hat{y}^n, \hat{y}^n) - \varepsilon^n$

$$=0 \quad =0 \quad \rightarrow \varepsilon^n \geq 0$$

Intuition

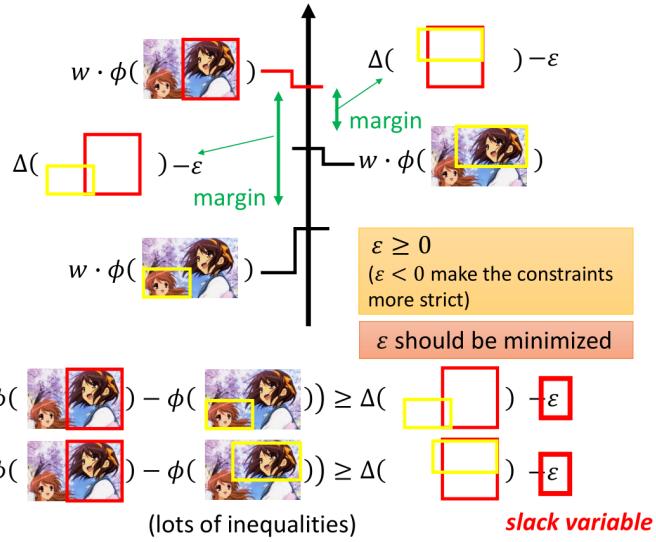
我们希望分数差大于 margin



It is possible that no w can achieve this.

$$\left. \begin{aligned} w \cdot (\phi(\text{red}) - \phi(\text{blue})) &\geq \Delta(\text{red}, \text{blue}) \\ w \cdot (\phi(\text{red}) - \phi(\text{blue})) &\geq \Delta(\text{red}, \text{blue}) \end{aligned} \right\} \quad \text{(lots of inequalities)} \quad \text{margin} \quad \forall y \neq \hat{y}$$

我们可能找不到一个 w 满足以上所有的不等式都成立。



因此将margin减去一个 ε (为了放宽限制, 但限制不应过宽, 否则会失去意义, ε 越小越好, 且要大于等于0)

假设, 我们现在有两个训练数据: (x^1, \hat{y}^1) 和 (x^2, \hat{y}^2)

对于 x^1 而言, 我们希望正确的分数减去错误的分数大于它们之间的 Δ 减去 ε^1 , 同时满足 $\varepsilon^1 \geq 0$

对于 x^2 而言, 同理, 我们希望正确的分数减去错误的分数, 要求大于它们之间的 Δ 减去 ε^2 , 同时满足: $\varepsilon^2 \geq 0$

在满足以上这些不等式的前提之下, 我们希望 $\lambda \sum_{n=1}^N \varepsilon^n$ 是最小的, 同时加上对应的正则项也满足最小化。

Training data: \hat{y}^1 \hat{y}^2
 Minimize $\frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$
 For x^1
 $w \cdot (\phi(x_1) - \phi(y^1)) \geq \Delta(\hat{y}^1, y^1) - \varepsilon^1$
 $w \cdot (\phi(x_1) - \phi(y^2)) \geq \Delta(\hat{y}^1, y^2) - \varepsilon^1$
(lots of inequalities) $\varepsilon^1 \geq 0$
 For x^2
 $w \cdot (\phi(x_2) - \phi(y^2)) \geq \Delta(\hat{y}^2, y^2) - \varepsilon^2$
 $w \cdot (\phi(x_2) - \phi(y^1)) \geq \Delta(\hat{y}^2, y^1) - \varepsilon^2$
(lots of inequalities) $\varepsilon^2 \geq 0$

我们的目标是, 求得 $w, \varepsilon^1, \dots, \varepsilon^N$, 最小化C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

同时, 要满足:

对所有的训练样本的所有不是正确答案的标记, $w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{y}^n, y) - \varepsilon^n$, $\varepsilon^n \geq 0$

Find $w, \varepsilon^1, \dots, \varepsilon^N$ minimizing C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

For $\forall n$:

For $\forall y \neq \hat{y}^n$:

$$w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{y}^n, y) - \varepsilon^n, \quad \varepsilon^n \geq 0$$

Solve it by the solver in SVM package

Quadratic Programming (QP) Problem

Too many constraints

可以利用SVM包中的solver来解决以上的问题；是一个二次规划(Quadratic Programming QP)的问题；但是约束条件过多，需要通过切割平面算法(Cutting Plane Algorithm)解决受限的问题。

Cutting Plane Algorithm for Structured SVM

在 w 和 ε^i 组成的参数空间中，颜色表示 C 的值，在没有限制的情况下， w 和 ε 越小越好，在有限制的情况下，只有内嵌的多边形区域内是符合约束条件的，因此需要在该区域内寻找最小值，即

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

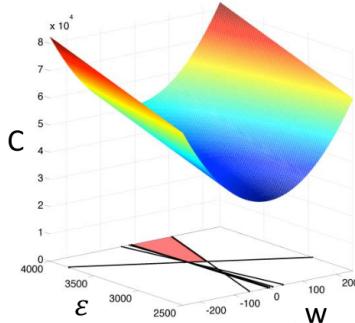
Find $w, \varepsilon^1, \dots, \varepsilon^N$ minimizing C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

For $\forall n$:

For $\forall y \neq \hat{y}^n$:

$$w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{y}^n, y) - \varepsilon^n, \quad \varepsilon^n \geq 0$$

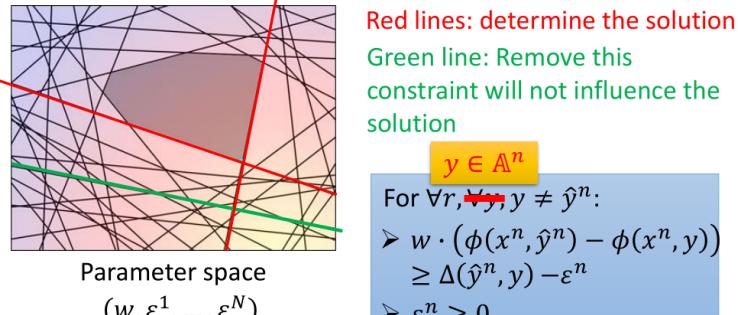


Source of image:

http://abnerguzman.com/publications/gkb_aistats13.pdf

Cutting Plane Algorithm

Although there are lots of constraints, most of them do not influence the solution.



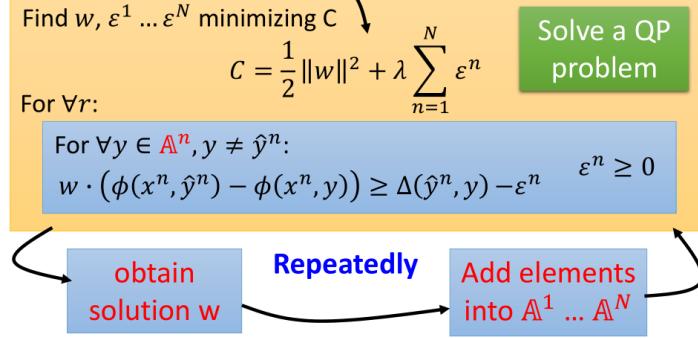
A^n : a very small set of $y \rightarrow$ working set

虽然有很多约束条件，但它们中的大多数的约束都是冗余，并不影响问题的解决；

原本是穷举 $y \neq \hat{y}^n$ ，而现在我们需要移除那些不起作用的线条，保留有用的线条，这些有影响的线条集可以理解为Working Set，用 A^n 表示。

Elements in working set A^n is selected iteratively

- Elements in **working set A^n** is selected iteratively
- Initialize $A^1 \dots A^N$

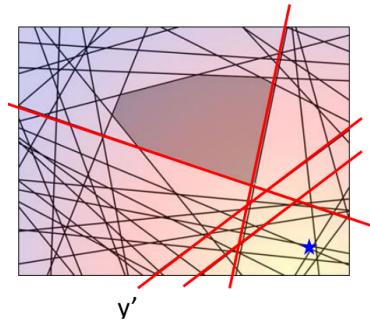


Strategies of adding elements into working set A^n

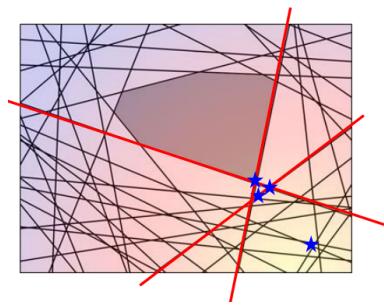


Initialize $\mathbb{A}^n = \text{null}$
No constraint at all

Solving QP
The solution w is
the blue point.



There are lots of constraints
is violated
Find the most violated one
Suppose it is the constraint
from y'
Extent the working set
 $\mathbb{A}^n = \mathbb{A}^n \cup \{y'\}$



假设 \mathbb{A}^n 初始值为空集合null，即没有任何约束限制，求解QP的结果就是对应的蓝点，但是不能满足条件的线条有很多很多，我们现在只找出没有满足的最“严重的”那一个即可。那么我们就把 $\mathbb{A}^n = \mathbb{A}^n \cup \{y'\}$

根据新获得的Working Set中唯一的成员 y' ，找寻新的最小值，进而得到新的 w ，尽管得到新的 w 和最小值，但依旧存在不满足条件的约束，需要继续把最难搞定的限制添加到有效集中，再求解一次。得到新的 w ，直到所有难搞的线条均添加到Working Set之中，最终Working Set中有三个线条，根据这些线条确定求解区间内的point，最终得到问题的解。

Find the most violated one

Given w' and ε' from working sets at hand, which constraint is the most violated one?

Constraint: $w' \cdot (\phi(x, \hat{y}) - \phi(x, y)) \geq \Delta(\hat{y}, y) - \varepsilon'$

Violate a Constraint:

$$w' \cdot (\phi(x, \hat{y}) - \phi(x, y)) < \Delta(\hat{y}, y) - \varepsilon'$$

Degree of Violation

$$\Delta(\hat{y}, y) - \varepsilon' - w' \cdot (\phi(x, \hat{y}) - \phi(x, y))$$

$$\rightarrow \Delta(\hat{y}, y) + w' \cdot \phi(x, y)$$

The most violated one:

$$\arg \max_y [\Delta(\hat{y}, y) + w' \cdot \phi(x, y)]$$

Cutting Plane Algorithm

- 给定训练数据集

$$\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$$

Working Set初始设定为

$$\mathbb{A}^1 \leftarrow \text{null}, \mathbb{A}^2 \leftarrow \text{null}, \dots, \mathbb{A}^N \leftarrow \text{null}$$

- 重复以下过程
 - 在初始的Working Set中求解一个QP问题的解，只需求解出w即可。
 - 针对求解出的w，要求对每一个训练数据 (x^n, \hat{y}^n) ，寻找最violated的限制，同时更新Working Set
- 直到Working Set中的元素不再发生变化，迭代终止，即得到要求解的w。

Given training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$

Working Set $A^1 \leftarrow null, A^2 \leftarrow null, \dots, A^N \leftarrow null$

Repeat

$w \leftarrow$ Solve a **QP** with Working Set A^1, A^2, \dots, A^N

$$\textbf{QP:} \quad \text{Find } w, \varepsilon^1 \dots \varepsilon^N \text{ minimizing} \quad \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

For $\forall n:$

For $\forall y \in A^n:$

$$w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{y}^n, y) - \varepsilon^n, \varepsilon^n \geq 0$$

Given training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$

Working Set $A^1 \leftarrow null, A^2 \leftarrow null, \dots, A^N \leftarrow null$

Repeat

$w \leftarrow$ Solve a **QP** with Working Set A^1, A^2, \dots, A^N

For each training data $(x^n, \hat{y}^n):$

$$\bar{y}^n = \arg \max_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)]$$

find the most violated constraints

Update working set $A^n \leftarrow A^n \cup \{\bar{y}^n\}$

Until A^1, A^2, \dots, A^N doesn't change any more

Return w

Multi-class and binary SVM

Multi-class SVM

Multi-class SVM

$$F(x, y) = w \cdot \phi(x, y)$$

• Problem 1: Evaluation

- If there are K classes, then we have K weight vectors $\{w^1, w^2, \dots, w^K\}$

$$y \in \{1, 2, \dots, k, \dots, K\}$$

$$F(x, y) = w^y \cdot \vec{x}$$

\vec{x} : vector

representation of x

$$w = \begin{pmatrix} w^1 \\ w^2 \\ \vdots \\ w^k \\ \vdots \\ w^K \end{pmatrix} \quad \phi(x, y) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vec{x} \\ \vdots \\ 0 \end{pmatrix}$$

- Problem 2: Inference

$$F(x, y) = w^y \cdot \vec{x}$$

$$\hat{y} = \arg \max_{y \in \{1, 2, \dots, k, \dots, K\}} F(x, y)$$

$$= \arg \max_{y \in \{1, 2, \dots, k, \dots, K\}} w^y \cdot \vec{x}$$

The number of classes are usually small,
so we can just enumerate them.

Multi-class SVM • Problem 3: Training	$y \in \{\text{dog, cat, bus, car}\}$ $\Delta(\hat{y}^n = \text{dog}, y = \text{cat}) = 1$ $\Delta(\hat{y}^n = \text{dog}, y = \text{bus}) = 100$ (defined as your wish)
--	---

Find $w, \varepsilon^1, \dots, \varepsilon^N$ minimizing C

$$C = \frac{1}{2} \|w\|^2 + \lambda \sum_{n=1}^N \varepsilon^n$$

For $\forall n$:

For $\forall y \neq \hat{y}^n$:	There are only N(K-1) constraints.
$(w^{\hat{y}^n} - w^y) \cdot \vec{x} \geq \underline{\Delta(\hat{y}^n, y)} - \varepsilon^n, \varepsilon^n \geq 0$	

$w \cdot \phi(x^n, \hat{y}^n) = w^{\hat{y}^n} \cdot \vec{x}$
 $w \cdot \phi(x^n, y) = w^y \cdot \vec{x}$

Some types of misclassifications may be worse than others.

Binary SVM

- Set $K = 2 \quad y \in \{1, 2\}$

For $\forall y \neq \hat{y}^n$:

$$(w^{\hat{y}^n} - w^y) \cdot \vec{x} \geq \underline{\Delta(\hat{y}^n, y)} - \varepsilon^n, \varepsilon^n \geq 0$$

$$= 1$$

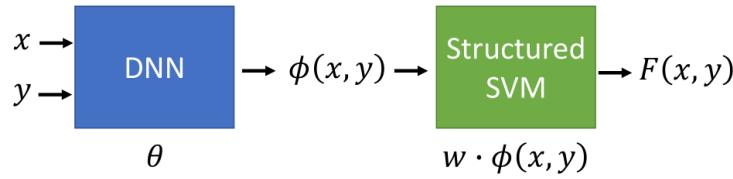
If $y=1$: $(w^1 - w^2) \cdot \vec{x} \geq 1 - \varepsilon^n \quad \Rightarrow \quad w \cdot \vec{x} \geq 1 - \varepsilon^n$

If $y=2$: $(w^2 - w^1) \cdot \vec{x} \geq 1 - \varepsilon^n \quad \Rightarrow \quad -w \cdot \vec{x} \geq 1 - \varepsilon^n$

Beyond Structured SVM

结构化SVM是线性结构的，如果想要结构化SVM的表现更好，我们需要定义一个较好的特征，但是人为设定特征往往十分困难，一个较好的方法是利用DNN生成特征，先用一个DNN，最后训练的结果往往十分有效。

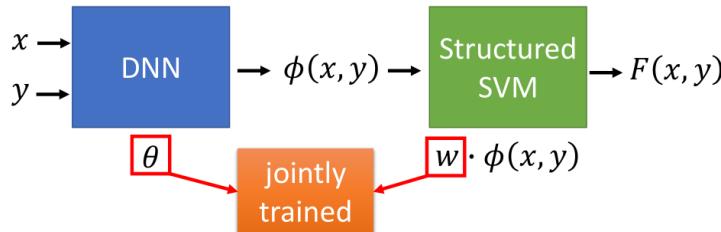
- Involving DNN when generating $\phi(x, y)$



Ref: Hao Tang, Chao-hong Meng, Lin-shan Lee, "An initial attempt for phoneme recognition using Structured Support Vector Machine (SVM)," ICASSP, 2010
 Shi-Xiong Zhang, Gales, M.J.F., "Structured SVMs for Automatic Speech Recognition," in Audio, Speech, and Language Processing, IEEE Transactions on, vol.21, no.3, pp.544-555, March 2013

将DNN与结构化SVM一起训练，同时更新DNN与结构化SVM中的参数。

- Jointly training structured SVM and DNN

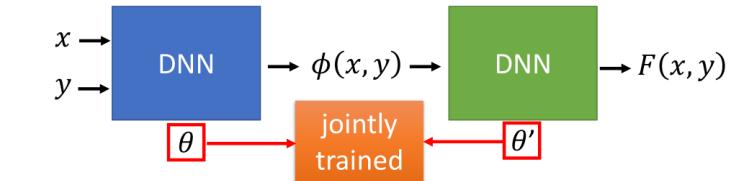


Ref: Shi-Xiong Zhang, Chaojun Liu, Kaisheng Yao, and Yifan Gong, "DEEP NEURAL SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION", Interspeech 2015

用一个DNN代替结构化SVM，即将x和y作为输入， $F(x, y)$ (为一个标量)作为输出。

- Replacing Structured SVM with DNN

A DNN with x and y as input and $F(x, y)$ (a scalar) as output



$$C = \frac{1}{2} \|\theta\|^2 + \frac{1}{2} \|\theta'\|^2 + \lambda \sum_{n=1}^N C^n$$

$$C^n = \max_y [\Delta(\hat{y}^n, y) + F(x^n, y)] - F(x^n, \hat{y}^n)$$

Ref: Yi-Hsiu Liao, Hung-yi Lee, Lin-shan Lee, "Towards Structured Deep Neural Network for Automatic Speech Recognition," ICASSP 2015
http://speech.ee.ntu.edu.tw/~tlkagk/paper/DNN_ASRU15.pdf

Sequence Labeling Problem

Sequence Labeling

$$f : X \rightarrow Y$$

序列标注的问题可以理解为：机器学习所要寻找的目标函数的输入是一个序列，输出也为一个序列，并且假设输入输出的序列长度相同，即输入可以写成序列向量的形式，输出也为序列向量。该任务可以利用循环神经网络来解决，但本章节我们可以基于结构化学习的其它方法进行解决(两步骤，三问题)。

Example Task

词性标记(POS tagging)

- 标记一个句子中每一个词的词性(名词、动词等等);
- 输入一个句子(比如，John saw the saw)，系统将会标记John为专有名词，saw为动词，the为限定词，saw为名词；

- 其在自然语言处理(NLP)中，是非常典型且重要的任务，也是许多文字理解的基石，用于后续句法分析和词义消歧。

如果不考虑序列，问题就无法解决(POS tagging仅仅依靠查表的方式是不够的，比如Hash Table，你需要知道一整个序列的信息，才能有可能把每个词汇的词性找出)

- John saw the saw.
 - 第一个"saw"更有可能是动词V，而不是名词N；
 - 然而，第二个"saw"是名词N，因为名词N更可能跟在限定词后面。

Hidden Markov Model (HMM)

How to generate a sentence?

Step 1

- 生成POS序列
- 基于语法(根据脑中内建的语法)

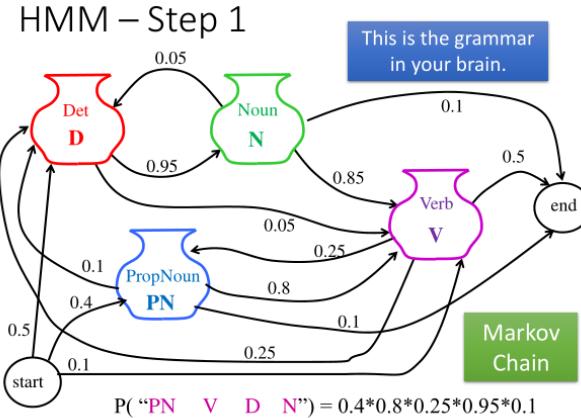
假设你大脑中一个马尔科夫链，开始说一句话时，放在句首的词性有50%的可能性为冠词，40%的可能性为专有名词，10%的可能性为动词，然后进行随机采样，再从专有名词开始，有80%的可能性后面为动词，动词后面有25%的可能性为冠词，冠词后面有95%的可能性为名词，名词后面有10%的可能性句子就结束了。

Step 2

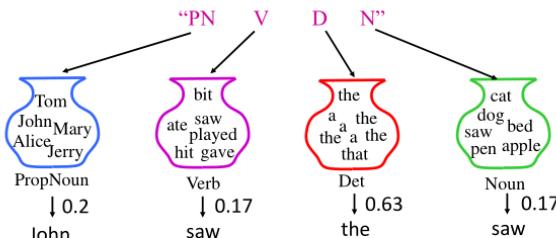
- 根据词序生成一个句子
- 基于词典

根据词性找到词典中对应的词汇，从不同的词性集合中采样出不同词汇所出现的机率。HMM可以描述为利用POS标记序列得到对应句子的机率，即

$$P(x, y) = P(y)P(x|y)$$



HMM – Step 2



$$\begin{aligned} P(\text{"John saw the saw"} | \text{"PN V D N"}) \\ = 0.2 * 0.17 * 0.63 * 0.17 \end{aligned}$$

x : John saw the saw.

Y : PN V D N

对应于：

$$x = x_1, x_2 \cdots x_L$$

$$y = y_1, y_2 \cdots y_L$$

其中，

$$P(x, y) = P(y)P(x|y)$$

- Step1(Transition probability)

$$P(y) = P(y_1|start) \times \prod_{l=1}^{L-1} P(y_{l+1}|y_l) \times P(end|y_L)$$

- Step2(Emission probability)

$$P(x|y) = \prod_{l=1}^L P(x_l|y_l)$$

Estimating the probabilities

- 我们如何知道 $P(V|PN)$, $P(saw|V)$?
 - 从训练数据中得到

$$P(x, y) = P(y_1|start) \prod_{l=1}^{L-1} P(y_{l+1}|y_l) P(end|y_L) \prod_{l=1}^L P(x_l|y_l)$$

其中，计算 $y_l = s$, 下一个标记为 s' 的机率，就等价于现在训练集里面 s 出现的次数除去 s 后面跟 s' 的次数；

$$\frac{P(y_{l+1} = s'|y_l = s)}{(s \text{ and } s' \text{ are tags })} = \frac{\text{count}(s \rightarrow s')}{\text{count}(s)}$$

计算某一个标记为 s 所产生的词为 t 的机率，就等价于 s 在整个词汇中出现的次数除去某个词标记为 t 的次数。

$$\frac{P(x_l = t|y_l = s)}{(s \text{ is tag, and } t \text{ is word })} = \frac{\text{count}(s \rightarrow t)}{\text{count}(s)}$$

How to do POS Tagging?

We can compute $P(x, y)$

给定 x (Observed), 发现 y (Hidden), 即如何计算 $P(x, y)$ 的问题

given x , find y

$$\begin{aligned} y &= \arg \max_{y \in Y} P(y|x) \\ &= \arg \max_{y \in Y} \frac{P(x, y)}{P(x)} \\ &= \arg \max_{y \in Y} P(x, y) \end{aligned}$$

Viterbi Algorithm

$$\tilde{y} = \arg \max_{y \in Y} P(x, y)$$

- 穷举所有可能的 y
 - 假设有 $|S|$ 个标记，序列 y 的长度为 L ;
 - 有可能的 y 即 $|S|^L$ (空间极为庞大)。
- 利用维特比算法解决此类问题
 - 复杂度为：

$$O(L|S|^2)$$

HMM - Summary

Evaluation

$$F(x, y) = P(x, y) = P(y)P(x|y)$$

该评估函数可以理解为x与y的联合概率。

Inference

$$\tilde{y} = \arg \max_{y \in \mathbb{Y}} P(x, y)$$

给定一个x，求出最大的y，使得我们定义函数的值达到最大(即维特比算法)。

Training

从训练数据集中得到 $P(y)$ 与 $P(x|y)$

该过程就是计算机率的问题或是统计语料库中词频的问题。

HMM - Drawbacks

- 在推理过程

$$\tilde{y} = \arg \max_{y \in \mathbb{Y}} P(x, y)$$

把求解最大的y作为我们的输出值。

- 为了得到正确的结果，我们需要让

$$(x, \hat{y}) : P(x, \hat{y}) > P(x, y)$$

但是HMM可能无法处理这件事情，它不能保证错误的y带进去得到的 $P(x,y)$ 一定是小的。

- Inference:

$$\tilde{y} = \arg \max_{y \in \mathbb{Y}} P(x, y)$$

- To obtain correct results ...

$$(x, \hat{y}) : P(x, \hat{y}) > \underline{P(x, y)} \quad \text{Can HMM guarantee that?}$$

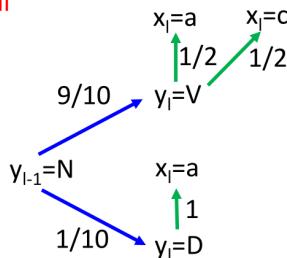
not necessarily small

Transition probability:

$$P(V|N)=9/10 \quad P(D|N)=1/10 \dots$$

Emission probability:

$$P(a|V)=1/2 \quad P(a|D)=1 \dots$$



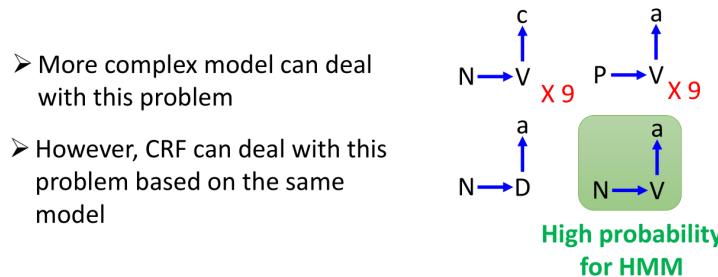
假设我们知道在 $l-1$ 时刻词性标记为N，即 $y_{l-1} = N$ ，在l时刻我们看到的单词为a，现在需要求出 $y_l = ?$

根据计算可以得到V的机率是0.45，D的机率是0.1。但是如果测试数据中有9个 $N \rightarrow V \rightarrow c$, 9个 $P \rightarrow V \rightarrow a$, 1个 $N \rightarrow D \rightarrow a$, 里面有和训练数据一样的数据，因此D更合理。

- The (x, y) never seen in the training data can have large probability $P(x, y)$.

• Benefit:

- When there is only little training data



通常情况下，隐马尔可夫模型是判断**未知数据**出现的**最大可能性**，即 (x, y) 在训练数据中从未出现过，但也可能有较大的概率 $P(x, y)$ ；

当训练数据很少的时候，使用隐马尔可夫模型，其性能表现是可行的，但当训练集很大时，性能表现较差；

隐马尔可夫模型会产生**未卜先知**的情况，是因为转移概率和发散概率，在训练时是分开建模的，两者是相互独立的，我们也可以用一个更复杂的模型来模拟两个序列之间的可能性，但要避免过拟合。

条件随机场的模型和隐马尔可夫模型是一样的，同时可以克服隐马尔可夫模型的缺点。

Conditional Random Field (CRF)

$$P(x, y) \propto \exp(w \cdot \phi(x, y))$$

条件随机场模型描述的也是 $P(x, y)$ 的问题，但与HMM表示形式很不一样(本质上是在训练阶段不同)，其机率正比于 $\exp(w \cdot \phi(x, y))$ 。

- $\phi(x, y)$ 为一个特征向量；
- w 是一个权重向量，可以从训练数据中学习得到；
- $\exp(w \cdot \phi(x, y))$ 总是正的，可能大于1。

$$P(x, y) = \frac{\exp(w \cdot \phi(x, y))}{R}$$

$$P(y|x) = \frac{P(x, y)}{\sum_{y'} P(x, y')} = \frac{\exp(w \cdot \phi(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x, y'))} = \frac{\exp(w \cdot \phi(x, y))}{Z(x)}$$

其中 $\sum_{y' \in \mathcal{Y}} \exp(w \cdot \phi(x, y'))$ 仅与 x 有关，与 y 无关

$P(x, y)$ for CRF

- HMM

$$P(x, y) = P(y_1 | start) \prod_{l=1}^{L-1} P(y_{l+1} | y_l) P(end | y_L) \prod_{l=1}^L P(x_l | y_l)$$

取对数

$$\begin{aligned} & \log P(x, y) \\ &= \log P(y_1 | start) + \sum_{l=1}^{L-1} \log P(y_{l+1} | y_l) + \log P(end | y_L) \\ &+ \sum_{l=1}^L \log P(x_l | y_l) \end{aligned}$$

其中，

$$\sum_{l=1}^L \log P(x_l | y_l) = \sum_{s,t} \log P(t | s) \times N_{s,t}(x, y)$$

- $\sum_{s,t}$ 穷举所有可能的标记s和所有可能的单词t；
- $\log P(t | s)$ 表示给定标记s的得到单词t的概率取对数
- $N_{s,t}(x, y)$ 表示为单词t被标记成s的事情，在 (x, y) 对中总共出现的次数。

Example

x: The dog ate the homework.
 ↓ ↓ ↓ ↓ ↓
 y: D N V D N

$N_{D,\text{the}}(x, y) = 2$
 $N_{N,\text{dog}}(x, y) = 1$
 $N_{V,\text{ate}}(x, y) = 1$
 $N_{N,\text{homework}}(x, y) = 1$
 $N_{S,t}(x, y) = 0$
 (for any other s and t)

$$\begin{aligned}
 & \sum_{l=1}^L \log P(x_l | y_l) \\
 &= \log P(\text{the}|D) + \log P(\text{dog}|N) + \log P(\text{ate}|V) \\
 &+ \log P(\text{the}|D) + \log P(\text{homework}|N) \\
 &= \log P(\text{the}|D) \times 2 + \log P(\text{dog}|N) \times 1 + \log P(\text{ate}|V) \times 1 \\
 &+ \log P(\text{homework}|N) \times 1 \\
 &= \sum_{s,t} \log P(t|s) \times N_{s,t}(x, y)
 \end{aligned}$$

每个单词都已经标记成对应的词性，我们分别计算出 D, N, V 在(x, y)对中出现的次数

然后计算所有的机率相乘的结果 $\sum_{l=1}^L \log P(x_l | y_l)$, 如上图, 整理之后的结果为 $\sum_{s,t} \log P(t|s) \times N_{s,t}(x, y)$

$$\begin{aligned}
 \log P(x, y) &= \log P(y_1 | start) + \sum_{l=1}^{L-1} \log P(y_{l+1} | y_l) + \log P(end | y_L) \\
 &+ \sum_{l=1}^L \log P(x_l | y_l) \\
 \log P(y_1 | start) &= \sum_s \log P(s | start) \times N_{start,s}(x, y) \\
 \sum_{l=1}^{L-1} \log P(y_{l+1} | y_l) &= \sum_{s,s'} \log P(s' | s) \times N_{s,s'}(x, y) \\
 \log P(end | y_L) &= \sum_s \log P(end | s) \times N_{s,end}(x, y)
 \end{aligned}$$

分析 $\log P(x, y)$ 的其他项

其中, 黄色表示对所有词性s放在句首的机率取对数, 再乘上在(x, y)对中, s放在句首所出现的次数;

绿色表示计算s后面跟s'在(x, y)里面所出现的次数, 再乘上s后面跟s'的机率取对数;

紫色同理, 最后一项表示两项相乘的形式。

则有 $\log P(x, y)$

$P(x, y)$ for CRF

$$\begin{aligned}
 \log P(x, y) &= \sum_{s,t} \log P(t|s) \times N_{s,t}(x, y) \\
 &+ \sum_s \log P(s | start) \times N_{start,s}(x, y) \\
 &+ \sum_{s,s'} \log P(s' | s) \times N_{s,s'}(x, y) \\
 &+ \sum_s \log P(end | s) \times N_{s,end}(x, y) \\
 &= \begin{bmatrix} \vdots \\ \log P(t|s) \\ \vdots \\ \log P(s | start) \\ \vdots \\ \log P(s' | s) \\ \vdots \\ \log P(end | s) \end{bmatrix} \cdot \begin{bmatrix} \vdots \\ N_{s,t}(x, y) \\ \vdots \\ N_{start,s}(x, y) \\ \vdots \\ N_{s,s'}(x, y) \\ \vdots \\ N_{s,end}(x, y) \end{bmatrix} \\
 &= w \cdot \phi(x, y) \\
 P(x, y) &= \exp(w \cdot \phi(x, y))
 \end{aligned}$$

等价于两个向量做内积, 进而可以用 $\log P(x, y) = w \cdot \phi(x, y)$ 表示, 第二个向量每一个element是依赖于(x, y)的, 因此可以写成 $\phi(x, y)$

由此可知, $P(x, y) = \exp(w \cdot \phi(x, y))$, 其中每一个w, 都对应着HMM模型中的某一个机率取对数。

因此对于每一个w, 取exponential就可以变为机率。但是我们在训练时对w没有任何限制, 得到w大于0时, 机率会大于一。

因此需要把 $P(x, y)$ 表达式变化为 $P(x, y) \propto \exp(w \cdot \phi(x, y))$

$$P(x, y) \propto \exp(w \cdot \phi(x, y))$$

$$\phi(x, y) = \begin{bmatrix} N_{s,t}(x, y) \\ \vdots \\ N_{start,s}(x, y) \\ \vdots \\ N_{s,s'}(x, y) \\ \vdots \\ N_{s,end}(x, y) \end{bmatrix}$$

$$w = \begin{bmatrix} w_{s,t} \\ \vdots \\ w_{start,s} \\ \vdots \\ w_{s,s'} \\ \vdots \\ w_{s,end} \end{bmatrix}$$

However, we do not give w any constraints during training

- $\rightarrow \log P(x_i = t | y_i = s)$
- $P(x_i = t | y_i = s) = e^{w_{s,t}}$ means
- $\rightarrow \log P(s | start)$
- $P(s | start) = e^{w_{start,s}}$ means
- $\rightarrow \log P(y_i = s' | y_{i-1} = s)$
- $P(y_i = s' | y_{i-1} = s) = e^{w_{s,s'}}$ means
- $\rightarrow \log P(end | s)$
-

Feature Vector

$\phi(x, y)$ 的形式是什么样的? $\phi(x, y)$ 分为两部分

Part 1: relations between tags and words

Feature Vector

- What does $\phi(x, y)$ look like?

x: The dog ate the homework.
 ↓ ↓ ↓ ↓ ↓
 y: D N V D N

- $\phi(x, y)$ has two parts

- Part 1: relations between tags and words

- Part 2: relations between tags
 If there are $|S|$ possible tags,
 $|L|$ possible words

Part 1 has $|S| \times |L|$ dimensions

Part 1	Value
D, the	2
D, dog	0
D, ate	0
D, homework	0
.....
N, the	0
N, dog	1
N, ate	0
N, homework	1
.....
V, the	0
V, dog	0
V, ate	1
V, homework	0
.....

如果有 $|S|$ 个可能的标记, $|L|$ 个可能的单词, Part 1的维度为 $|S| \times |L|$, value表示在(标记, 单词)对中出现的次数, 所以这是一个维度很大的稀疏vector;

Part 2: 标签之间的关系

Feature Vector

- What does $\phi(x, y)$ look like?

x: The dog ate the homework.
 ↓ ↓ ↓ ↓ ↓
 y: D N V D N

- $\phi(x, y)$ has two parts

- Part 1: relations between tags and words

- Part 2: relations between tags

$N_{s,s'}(x, y)$: Number of tags s and s' consecutively in (x, y)

Part 2	Value
$N_{D,D}(x, y) \rightarrow D, D$	0
$N_{D,N}(x, y) \rightarrow D, N$	2
D, V	0
.....
N, D	0
N, N	0
N, V	1
.....
V, D	1
V, N	0
V, V	0
.....
Start, D	1
Start, N	0
.....
End, D	0
End, N	1

定义 $N_{S,S'}(x, y)$: 为标记 s 和 s' 在 (x, y) 对中连续出现的次数，如果有 $|S|$ 个可能的标记，这部分向量维度为 $|S| \times |S| + 2|S|$ (s 之间、start, end)。

CRF 中可以自己定义 $\phi(x, y)$

CRF – Training Criterion

给定训练数据：

$$\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$$

找到一个权重向量 w^* 去最大化目标函数 $O(w)$ ；

其中， w^* 与目标函数定义如下：

$$w^* = \arg \max_w O(w)$$

$$O(w) = \sum_{n=1}^N \log P(\hat{y}^n | x^n)$$

表示为我们要寻找一个 w ，使得最大化给定的 x_n 所产生 \hat{y}^n 正确标记的机率，再取对数进行累加，此处可以联想到交叉熵也是最大化正确维度的机率再取对数，只不过此时是针对整个序列而言的。

对 $\log P(y|x)$ 做相应的转换

$$P(y|x) = \frac{P(x,y)}{\sum_{y'} P(x,y')}$$

$$\log P(\hat{y}^n | x^n) = \log P(x^n, \hat{y}^n) - \log \sum_{y'} P(x^n, y')$$

CRF – Training Criterion

- Given training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$
- Find the weight vector w^* maximizing objective function $O(w)$:

$$w^* = \arg \max_w O(w) \quad O(w) = \sum_{n=1}^N \log P(\hat{y}^n | x^n)$$

$$\log P(\hat{y}^n | x^n) = \log \underline{P(x^n, \hat{y}^n)} - \log \overline{\sum_{y'} P(x^n, y')}$$

Maximize what we observe

Minimize what we don't observe

根据CRF的定义可知，可以分解为两项再分别取对数，即最大化观测到的机率，最小化没有观测到的机率。

Gradient Ascent

梯度下降：找到一组参数 θ ，最小化成本函数 $C(\theta)$ ，即梯度的反方向

$$\theta \rightarrow \theta - \eta \nabla C(\theta)$$

梯度上升：找到一组参数 θ ，最大化成本函数 $O(\theta)$ ，即梯度的同方向

$$\theta \rightarrow \theta + \eta \nabla O(\theta)$$

Gradient descent

Find a set of parameters θ minimizing cost function $C(\theta)$

$$\theta \rightarrow \theta - \eta \nabla C(\theta)$$

Opposite direction of the gradient

Gradient Ascent

Find a set of parameters θ maximizing objective function $O(\theta)$

$$\theta \rightarrow \theta + \eta \nabla O(\theta)$$

The same direction of the gradient

CRF - Training

$$O(w) = \sum_{n=1}^N \log P(\hat{y}^n | x^n) = \sum_{n=1}^N O^n(w)$$

Compute $\nabla O^n(w) = \begin{bmatrix} \vdots \\ \partial O^n(w)/\partial w_{s,t} \\ \vdots \\ \partial O^n(w)/\partial w_{s,s'} \\ \vdots \end{bmatrix}$

Let me show $\frac{\partial O^n(w)}{\partial w_{s,t}}$
 $\frac{\partial O^n(w)}{\partial w_{s,s'}}$ very similar

求偏导

$$O^n(w) = \log \frac{\exp(w \cdot \phi(x^n, \hat{y}^n))}{Z(x^n)} \quad Z(x^n) = \sum_{y'} \exp(w \cdot \phi(x^n, y'))$$

$$= w \cdot \phi(x^n, \hat{y}^n) - \log Z(x^n)$$

$$\frac{\partial O^n(w)}{\partial w_{s,t}} = \underline{N_{s,t}(x^n, \hat{y}^n)}$$

↓

The number of word t labeled as s in (x^n, \hat{y}^n)

$$w \cdot \phi(x^n, \hat{y}^n)$$

The value of the dimension in $\phi(x^n, \hat{y}^n)$ corresponding to $w_{s,t}$.

$$= \sum_{s,t} w_{s,t} \cdot N_{s,t}(x^n, \hat{y}^n)$$

$$+ \sum_{s,s'} w_{s,s'} \cdot N_{s,s'}(x^n, \hat{y}^n)$$

$$O^n(w) = \log \frac{\exp(w \cdot \phi(x^n, \hat{y}^n))}{Z(x^n)} \quad Z(x^n) = \sum_{y'} \exp(w \cdot \phi(x^n, y'))$$

$$= w \cdot \phi(x^n, \hat{y}^n) - \log Z(x^n)$$

$$\frac{\partial O^n(w)}{\partial w_{s,t}} = \underline{N_{s,t}(x^n, \hat{y}^n)} - \frac{1}{Z(x^n)} \underline{\frac{\partial Z(x^n)}{\partial w_{s,t}}}$$

$$= \sum_{y'} \frac{\exp(w \cdot \phi(x^n, y'))}{Z(x^n)} N_{s,t}(x^n, y') \cancel{= \sum_{y'} P(y'|x^n) N_{s,t}(x^n, y')}$$

$$\frac{\partial Z(x^n)}{\partial w_{s,t}} = \sum_{y'} \exp(w \cdot \phi(x^n, y')) N_{s,t}(x^n, y')$$

$$P(y'|x^n) = \frac{\exp(w \cdot \phi(x^n, y'))}{Z(x^n)}$$

CRF - Training

$$w_{s,t} \rightarrow w_{s,t} + \eta \frac{\partial O(w)}{\partial w_{s,t}}$$

After some math

Can be computed by Viterbi algorithm as well

$$\frac{\partial O^n(w)}{\partial w_{s,t}} = \underline{N_{s,t}(x^n, \hat{y}^n)} - \sum_{y'} \underline{P(y'|x^n) N_{s,t}(x^n, y')}$$

If word t is labeled by tag s in training examples (x^n, \hat{y}^n) , then increase $w_{s,t}$

If word t is labeled by tag s in (x^n, y') which not in training examples, then decrease $w_{s,t}$

偏导求解得到两项：

$$\frac{\partial O^n(w)}{\partial w_{s,t}} = \underline{N_{s,t}(x^n, \hat{y}^n)} - \sum_{y'} \underline{P(y'|x^n) N_{s,t}(x^n, y')}$$

- 第一项为单词t被标记为s, 在 (x^n, \hat{y}^n) 中出现的次数;
- 第二项为累加所有可能的y, 每一项为 单词t被标记成s在 x_n 与任意y的pair里面出现的次数乘上给定 x_n 下产生这个y的机率。
- 实际意义解释
 - 第一项说明：如果(s, t)在训练数据集正确出现的次数越多，对应的w的值就会越大，即如果单词t在训练数据对集 (x^n, \hat{y}^n) 中被标记成s，则会增加 $w_{s,t}$;
 - 第二项说明：如果(s, t)在训练数据集任意的y与x配对之后出现的次数依然越多，那么我们应该将其权值进行减小(可以通过Viterbi 算法计算)，即如果任意一个单词t在任意一个训练数据对集 (x^n, y') 中被标记成s的话，我们要减小 $w_{s,t}$ 。

对所有的权值向量来说，更新过程是：正确的 \hat{y}^n 所形成的的向量减去任意一个y形成的的向量乘上y的机率。

$$\nabla O(w) = \phi(x^n, \hat{y}^n) - \sum_{y'} P(y'|x^n) \phi(x^n, y')$$

Stochastic Gradient Ascent

Random pick a data (x^n, \hat{y}^n)

$$w \rightarrow w + \eta \left(\phi(x^n, \hat{y}^n) - \sum_{y'} P(y'|x^n) \phi(x^n, y') \right)$$

CRF – Inference

$$y = \arg \max_{y \in Y} P(y|x) = \arg \max_{y \in Y} P(x,y)$$

$$= \arg \max_{y \in Y} w \cdot \phi(x, y) \quad \text{Done by Viterbi as well}$$

$$P(x,y) \propto \exp(w \cdot \phi(x, y))$$

等同于找一个 y , 使得 $w \cdot \phi(x, y)$ 机率最大, 因为由 $P(x,y) \propto \exp(w \cdot \phi(x, y))$ 可知。

CRF v.s. HMM

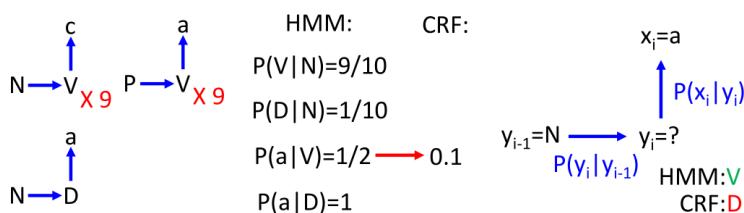
CRF增加 $P(x, \hat{y})$, 减少 $P(x, y')$ (HMM做不到这一点)

- 如果要得到正确的答案, 我们希望

$$(x, \hat{y}) : P(x, \hat{y}) > P(x, y)$$

条件随机场更有可能得到正确的结果。

CRF可能会想办法调整参数, 把V产生a的机率变小, 让正确的机率变大, 错误的变小。



Synthetic Data

输入输出分别为:

$$x_i \in \{a-z\}, y_i \in \{A-E\}$$

从混合顺序隐马尔科夫模型生成数据

- 转移概率

$$\alpha P(y_i|y_{i-1}) + (1-\alpha)P(y_i|y_{i-1}, y_{i-2})$$

α 取1时, 变为一般的隐马尔科夫模型, 其值可以任意地进行调整。

- 发散概率

$$\alpha P(x_i|y_i) + (1-\alpha)P(x_i|y_i, x_{i-1})$$

如果 α 取1时, 变为一般的隐马尔科夫模型。

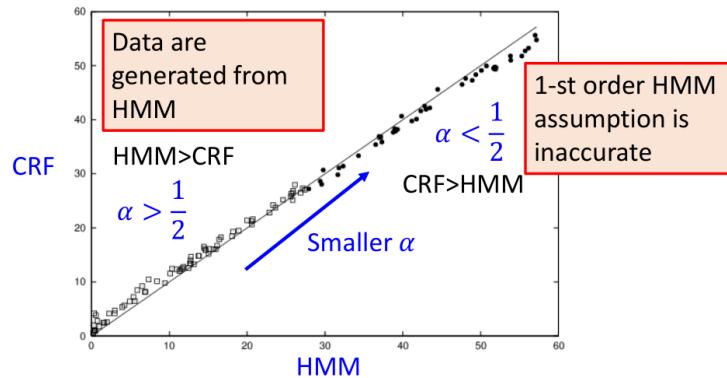
Comparing HMM and CRF

- $x_i \in \{a - z\}, y_i \in \{A - E\}$
- Generating data from a mixed-order HMM
 - Transition probability:
 - $\alpha P(y_i|y_{i-1}) + (1 - \alpha)P(y_i|y_{i-1}, y_{i-2})$
 - Emission probability:
 - $\alpha P(x_i|y_i) + (1 - \alpha)P(x_i|y_i, x_{i-1})$
- Comparing HMM and CRF
 - All the approaches only consider 1-st order information
 - Only considering the relation of y_{i-1} and y_i
 - In general, all the approaches have worse performance with smaller α

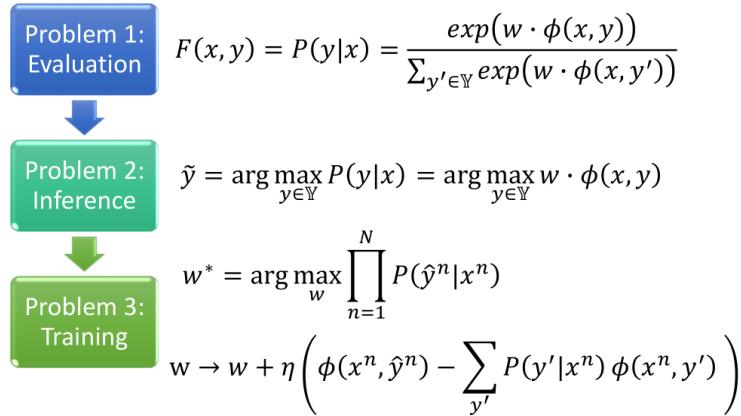
Ref: John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", ICML, 2001

α 从左下方到右上方不断减小，每一个圈圈表示不同的 α 所得到的结果，对每一个点都做一个隐马尔科夫模型与条件随机场的实验，横轴代表隐马尔科夫模型犯错的百分比，纵轴表示条件随机场犯错的百分比。

当模型与假设不合的时候，CRF比HMM得到了更好的结果



CRF - Summary



$w^* = \arg \max$ 的式子，可以写成对数相加形式。

Structured Perceptron

x, y 假设都为序列，可以用条件随机场模型来定义 $\phi(x, y)$ ；

Problem 2 利用维特比算法求解即可；

训练时，对所有的训练数据 n ，以及对所有的 $y \neq \hat{y}^n$ ，我们希望：

$$w \cdot \phi(x^n, \hat{y}^n) > w \cdot \phi(x^n, y)$$

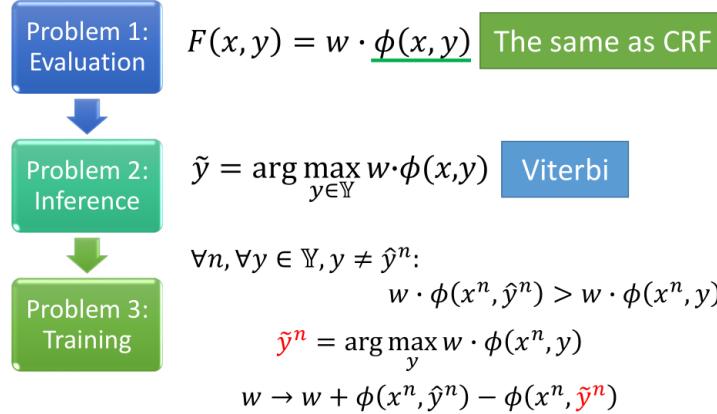
在每个iteration里面，我们会根据目前的w，找到一个 \tilde{y}^n ，使得：

$$\tilde{y}^n = \arg \max_y w \cdot \phi(x^n, y)$$

然后，更新w

$$w \rightarrow w + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$

即正确的 \hat{y}^n 减去其他的 \tilde{y}^n 所形成的向量。



Structured Perceptron v.s. CRF

Structured Perceptron

$$\begin{aligned}\tilde{y}^n &= \arg \max_y w \cdot \phi(x^n, y) \\ w &\rightarrow w + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\end{aligned}$$

只减去机率最大的y的特征向量

CRF

$$w \rightarrow w + \eta \left(\frac{\phi(x^n, \hat{y}^n)}{\sum_{y'} P(y'|x^n) \phi(x^n, y')} \right)$$

减去了所有的 y' 所形成的特征向量与对应的机率做weighted sum

- Structured Perceptron

$$\begin{aligned}\tilde{y}^n &= \arg \max_y w \cdot \phi(x^n, y) \\ w &\rightarrow w + \underline{\phi(x^n, \hat{y}^n)} - \underline{\phi(x^n, \tilde{y}^n)}\end{aligned}$$

Hard

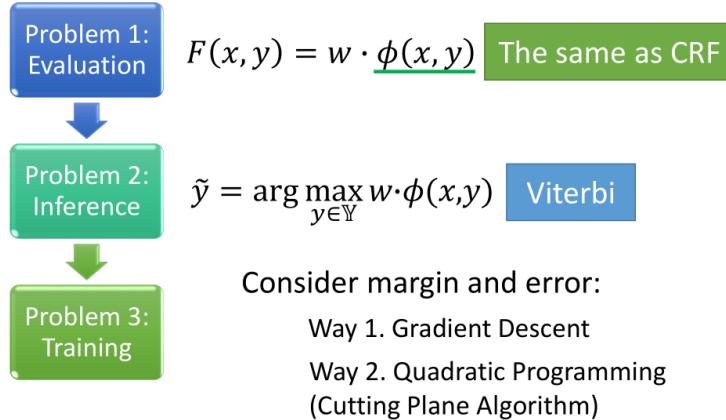
- CRF

$$w \rightarrow w + \eta \left(\underline{\phi(x^n, \hat{y}^n)} - \underline{\sum_{y'} P(y'|x^n) \phi(x^n, y')} \right)$$

Soft

Structured SVM

目标函数需要考虑到间隔和误差，训练时可以采用梯度下降法，也可以作为QP问题，因为限制条件过多，所以采用切割平面算法解。



Error Function

Error function

$$\Delta(\hat{y}^n, y)$$

- Δ 用来计算 y 与 \hat{y}^n 之间的差异性；
- 结构化支持向量机的成本函数就是 Δ 的上界，最小化Cost Function就是在最小化上界；
- 理论上讲， Δ 可以为任何适当的函数；但是，我们必须要考虑到，我们需要穷举所有的 y ，看哪一个使得 Δ 加上 $w \cdot \phi$ 最大化（这是Problem 2.1，相比Problem 2是一个不一样的问题）

在下图示例情况下，把 Δ 定义成错误率，Problem 2.1可以通过维特比算法求解。

- Error function: $\Delta(\hat{y}^n, y)$
- $\Delta(\hat{y}^n, y)$: Difference between y and \hat{y}^n
- Cost function of structured SVM is the upper bound of $\Delta(\hat{y}^n, y)$
- Theoretically, $\Delta(y, \hat{y}^n)$ can be any function you like
- However, you need to solve **Problem 2.1**
- $\bar{y}^n = \operatorname{argmax}_y [\Delta(\hat{y}^n, y) + w \cdot \phi(x^n, y)]$

Example \hat{y} : A T T C G G G G A T
 $\Delta(\hat{y}, y) = 3/10$ y : A T T A G G A G A A

In this case, problem 2.1 can be solved by Viterbi Algorithm

Performance of Different Approaches

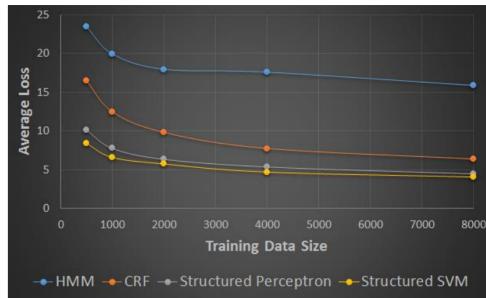
POS

- HMM performance表现最差，但是当训练数据更少的情况下，可能隐马尔科夫模型表现会相比其他方法好一些；CRF赢过HMM，
- 条件随机场与结构化感知机谁比较强文献上是没有定论的，条件随机场是soft的，而结构化感知机是hard的。但是结构化感知机只需要求解Problem 2就可以了，而条件随机场需要summation over 所有的 y' ，这件事不一定知道怎么解，如果不知道怎么解的话，推荐用结构化感知机。
- 结构化支持向量机整体表现最好；

命名实体识别（把tag换成公司名/地名/人名等）

- 结构化支持向量机模型表现最好；
- 隐马尔科夫模型表现最差。

Ref: Nguyen, Nam, and Yunsong Guo.
POS Tagging "Comparisons of sequence labeling
algorithms and extensions." *ICML*, 2007.



Name Entity Recognition

Method	HMM	CRF	Perceptron	SVM
Error	9.36	5.17	5.94	5.08

Ref: Tsochantaridis, Ioannis, et al. "Large margin methods for structured and interdependent output variables." *Journal of Machine Learning Research*. 2005.

RNN v.s. Structured Learning

RNN, LSTM

- 单方向的循环神经网络或长短时记忆网络并没有考虑到全部的序列，换言之，只考虑时间 t_1 至当前时间 t_k 的情形，对 t_{k+1} 的情形没有考虑。
- 有足够的data，或许可以learn到标签之间的依赖关系
- Cost和Error并不见得总是相关的
- 可以叠加很多层（利用Deep的特性）

HMM, CRF, Structured Perceptron/SVM

- 在输出结果之前，做的都是利用维特比算法穷举所有的序列，观测最大的序列，在计算过程中考虑到的是整个序列；（开放问题：如果利用双向的循环神经网络与其相比，结果如何？）胜的可能比较牵强。
- 我们可以直接把限制加到维特比算法中，可以只穷举符合限制的sequence，可以把标签之间的依赖关系明确的描述在model中；
- 结构化支持向量机的Cost Function就是Error的上界，当Cost Function不断减小的时候，Error很有可能会随之降低。
- 其实也可以是deep，但是它们要想拿来做deep learning 是比较困难的。在我们讲的内容里面它们都是linear，因为他们定义的 evaluation函数是线性的。如果不是线性的话也会很麻烦，因为只有是线性的我们才能套用这些方法来做inference和training。

RNN, LSTM

- Unidirectional RNN does not consider the whole sequence
- Cost and error not always related
- Deep

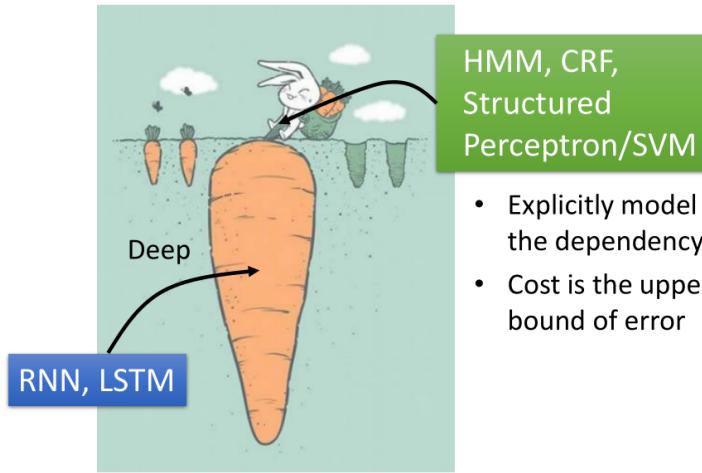


HMM, CRF, Structured Perceptron/SVM

- Using Viterbi, so consider the whole sequence
- How about Bidirectional RNN?
- Can explicitly consider the label dependency
- Cost is the upper bound of error

最后总结来看，RNN/LSTM在deep这件事的表现其实会比较好，同时在SOTA上，RNN是不可或缺的，如果只是线性的模型，function space就这么大，就算可以直接最小化一个错误的上界，但是这样没什么，因为所有的结果都是坏的，所以相比之下，deep learning占到很大的优势。

Integrated Together



- 底层（埋在土里的萝卜）
 - 循环神经网络与长短时记忆网络
- 叶子
 - 隐马尔可夫模型，条件随机场与结构化感知机/支持向量机，有很多优点。

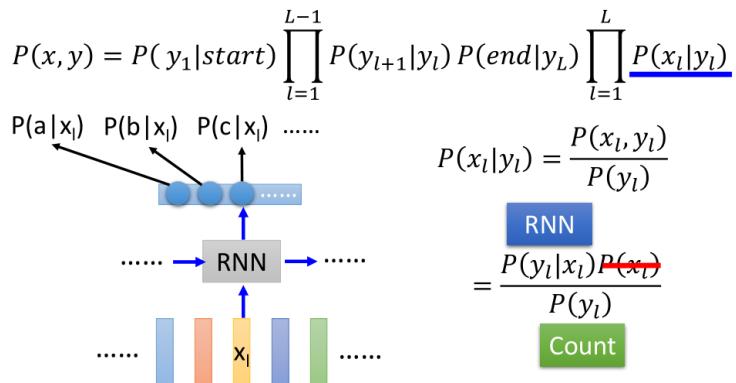
语音识别

- 卷积神经网络/循环神经网络或长短时记忆网络/深度神经网络 + **隐马尔可夫模型**

$$P(x, y) = P(y_1 | \text{start}) \prod_{l=1}^{L-1} P(y_{l+1} | y_l) P(\text{end} | y_L) \prod_{l=1}^L P(x_l | y_l)$$

- 根据隐马尔科夫模型中，发散概率可以由神经网络得到，用循环神经网络的输出结果经过变换代替发散概率；不需要考虑 $P(x_l)$ 的概率；
- 双向循环神经网络/长短时记忆网络 + 条件随机场/结构化支持向量机。

- **Speech Recognition: CNN/LSTM/DNN + HMM**



其实加上HMM在语音辨识里很有帮助，就算是用RNN，但是在辨识的时候，常常会遇到问题，假设我们是一个frame，用RNN来问这个frame属于哪个form，往往会产生奇怪的结果，比如说一个frame往往是蔓延好多个frame，比如理论是看到第一个frame是A，第二个frame是A，第三个是A，第四个是A，然后BBB，但是如果用RNN做的时候，RNN每个产生的label都是独立的，所以可能会若无其事的改成B，然后又是A，RNN很容易出现这个现象。HMM则可以把这种情况修复。因为RNN在训练的时候是每个frame分来考虑的，因此不同地方犯的错误对结果的影响相同，结果就会不好，如果想要不同，加上结构化学习的概念才可以做到。所以加上结构化学习的概念会很有帮助。

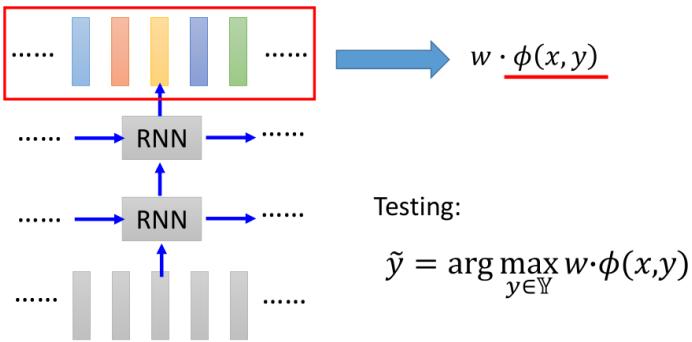
Semantic Tagging

- 从RNN的输出结果中，抽出新的特征再计算； $w \cdot \phi(x, y)$ 作为评估函数
 - 训练阶段
利用梯度下降法让w和循环神经网络中的所有参数一起训练；
 - 测试阶段

$$\tilde{y} = \arg \max_{y \in \mathbb{Y}} w \cdot \phi(x, y)$$

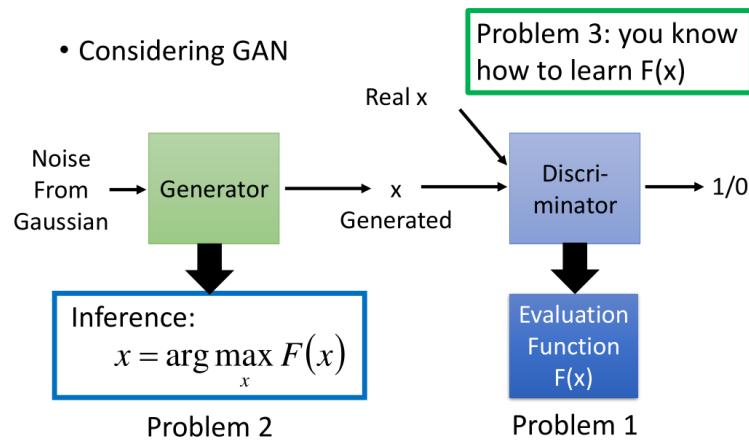
找一个 y , 使得 $w \cdot \phi(x, y)$ 的结果最大化, 但此时的 x 不是input x , 而是来自于循环神经网络的输出结果。

- Semantic Tagging: Bi-directional LSTM + CRF/Structured SVM



Is Structure learning practical?

structured learning需要解三个问题, 其中problem 2往往很困难, 因为要穷举所有的 y 让其最大, 解一个optimization的问题, 大部分状况都没有好的解决办法。所有有人说structured learning应用并不广泛, 但是未来未必是这样的。



其实GAN就是一种structured learning。可以把discriminator看做是evaluation function (也就是problem 1) 我们要解一个inference的问题 (problem 2) , 我们要穷举我们未知的东西, 看哪个可以让我们的evaluation function最大。这步往往比较困难, 因为 x 的可能性太多了。但这个东西可以就是generator, 我们可以想成generator就是给出一个noise, 输出一个object, 它输出的这个object, 就是让discriminator分辨不出的object, 如果discriminator就是evaluation function的话, 那output的值就是让evaluation function的值很大的那个对应值。所以这个generator就是在解problem 2, 其实generator的输出就是argmax的输出, 可以把generator当做在解inference这个问题。Problem 3的solution就是train GAN的方法。

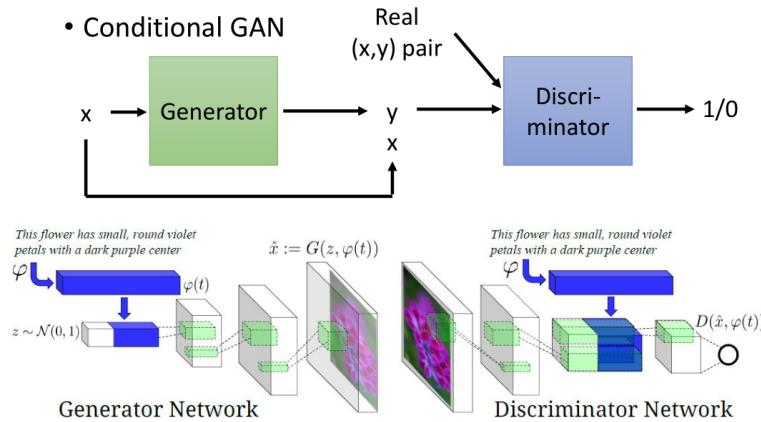
在structured SVM的training里面, 我们每次找出最competitive的那些example, 然后我们希望正确的example的evaluation function的分数大过competitive的example, 然后update我们的model, 然后再重新选competitive的example, 然后再让正确的, 大过competitive, 就这样iterative去做。

GAN的training是我们有正确的example, 它应该要让evaluation function的值比Discriminator的值大, 然后我们每次用这个Generator, Generate出最competitive的那个 x , 也就是可以让Discriminator的值最大的那个 x , 然后再去train Discriminator。Discriminator要分辨正确的跟Generated的。也就是Discriminator要给real的example比较大的值, 给那些most competitive的 x 比较小的值, 然后这个process就不断的iterative的进行下去, 你会update你的Discriminator, 然后update你的Generator。

其实这个跟Structured SVM的training是有异曲同工之妙的。

我们在讲structured SVM的时候都是有一个input/output, 有一个 x 有一个 y ; GAN只有 x , 听起来好像不太像, 那我们就另外讲一个像的。

GAN也可以是conditional的GAN, example都是 x, y 的pair, 现在的任务是, given x 找出最有可能的 y 。



比如语音辨识， x 是声音讯号， y 是辨识出来的文字，如果是用conditional的概念，generator输入一个 x ，就会output一个 y ，discriminator是去检查 y 的pair是不是对的，如果给他一个真正的 x, y 的pair，会得到一个比较高的分数，给一个generator输出的一个 y 配上输入的 x 所产生的一个假的pair，就会给他一个比较低的分数。

训练的过程就和原来的GAN就是一样的，这个已经成功运用在文字产生图片这个task上面。这个task的input就是一句话，output就是一张图，generator做的事就是输入一句话，然后产生一张图片，而discriminator要做的事就是给他一张图片和一句话，要他判断这个 x, y 的pair是不是真的，如果把discriminator换成evaluation function，把generator换成解inference的问题，其实conditional GAN和structured learning就是可以类比，或者说GAN就是训练structured learning model的一种方法。

很多人都有类似的想法，比如GAN可以跟energy based model做connection，可以视为train energy based model的一种方法。所谓energy based model，它就是structured learning的另外一种称呼。

Generator视做是在做inference这件事情，是在解arg max这个问题，听起来感觉很荒谬。但是也有人觉得一个neural network，它有可能就是在解arg max这个问题，这里也给出一些Reference。

所以也许deep and structured就是未来一个研究的重点的方向。

Sounds crazy?
 People do think in this way ...
Deep and Structured
will be the future.

- Connect Energy-based model with GAN:
 - A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models
 - Deep Directed Generative Models with Energy-Based Probability Estimation
 - ENERGY-BASED GENERATIVE ADVERSARIAL NETWORKS
- Deep learning model for inference
 - Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures
 - Conditional Random Fields as Recurrent Neural Networks

Concluding Remarks