Decision Tree

- Entropy
  - Uncertainty를 reducing하는 attribute로 chech하는 것이 좋다.
  - Uncertainty를 measure하자.
  - Attribute를 어떤 특정 distribution으로 생성된 random variable로 보고 생각해보자.
    - Higer entropy means more uncertainty
    - $H(X) = -\sum_X P(X=x) log_b P(X=x)$
    - 연속함수 일때는 sigma가 적분으로
  - Conditional Entropy
    - 특정 feature에 대한 정보가 있을 때 엔트로피 설정. (given X)
    - $H(Y|X) = \sum_X P(X=x) log_b P(Y|X=x)$
    - $= \sum_X P(X=x) \left\{ -\sum_Y P(Y=y|X=x) log_b P(Y=y|X=x) \right\}$
    - 앞의 $\sum_X P(X=x)$는 사전정보의 역할을 함을 알 수 있다.
- Information Gain
  - feature 말고, 클래스에 대해서 엔트로피를 잴 수 있을 것이다.
  - 클래스 Y에 대한 엔트로피를 측정하고, given feature에 대한 Y에 대한 조건부 엔트로피를 알아보면.

    - Let's calculate the entropy values
      - $H(Y) = -\sum_{Y\in\{+,-\}} P(Y=y) log_2 P(Y=y)$
      - $H(Y|A1) = \sum_{X\in\{a,b,?\}} \sum_{Y\in\{+,-\}} P(A1=x, Y=y) log_2 \frac{P(A1=x)}{P(A1=x, Y=y)}$
      - $H(Y|A9) = \sum_{X\in\{t,f\}} \sum_{Y\in\{+,-\}} P(A9=x, Y=y) log_2 \frac{P(A9=x)}{P(A9=x, Y=y)}$
  - $IG(Y, A_i) = H(Y) - H(Y|A_i)$
    - H(Y)는 원래 이정도의 | 레벨이었는데 condition을 줬을 때, 이렇게 바뀌었다.
    - Information Gain이 높은 것을 루트로 트리를 만든다.

Naive Bayes

설명

나이브 베이즈 분류기는 given data에서 class의 사후확률이 높은 방향으로 베이즈 정리를 활용하여 분류한다.
이때 사전확률은 Map나 MLE를 활용해 구할 수 있고
Likelihood를 구하기 위해서 모든 feature와 class에대한 결합확률을 계산해야하지만, $p(C_k|\mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$
조건부 독립성을 가정하여 (2^d-1)k 개의 파라미터를 계산해야하는 것을 (2-1)dk개로 줄여 분류한다.

- 설명 알데마 수업 중간고사 정리부분

Ch4. 나이브 베이스 알고리즘 (Naive Bayes Algorithm)
Posterior Prob. = Likelihood * Prior Prob. / Marginal Likelihood

$$P(A|B) = \frac{P(B \cap A)}{P(B)} = \frac{P(A)P(B|A)}{P(B)} = \frac{P(A)P(B|A)}{P(A)P(B|A) + P(A^c)P(B|A^c)}$$

Pr(A): 사전 확률(prior probability), 새로운 자료가 없는 상태에서 어떤 사건이 일어날 확률에 대한 가정
Pr(B|A): 가능도(likelihood), 사건이 일어났다는 가정 하에서 새로이 가지게 된 자료가 관측될 확률
Pr(A|B): 사후 확률(posterior probability), 사전확률과 가능도를 이용해서 새롭게 계산한, (새로운 자료로 업데이트 하여 판단한) 어떤 사건이 일어날 확률
P(B): 베이즈 정리의 분모에 해당하는 부분은 가능도를 구할 때 조건으로 걸린 사건(위의 예의 경우, (실제 병의 유무와는 상관 없이) '양성 판정이 나올 확률')의 확률. 기능적으로는 사후 확률이 확률의 정의(0 이상 1 이하임)을 충족시키도록 사전확률과 가능도의 곱을 보정해주는 역할.

위와 같은 예에서는 쉽게 계산할 수 있고 엄밀하게 사후확률을 구하려면 반드시 필요한 부분이지만, 실제로 생각보다 계산이 까다로울 경우 등식을 비례 관계로 바꾸고 생략할 수도 있다.

- 의미

교재: 사전정보(Prior Info.)를 활용하여 다음의사결정의 확률이 달라짐. 사람 뇌 신경계(Neuron)은 불확실성의 세계에서 주어진 정보를 토대로 최적에 가까운 의사결정을 내리게 진화했다는 토마스 베이즈의 철학적 관점 반영
Bayes Theorem은 의사결정에서 주어진 데이터 이외에 우리가 이전에 알고 있던 사전지식을 적절히 활용하여 의사결정하는 방법론(Bayesian Approach)로 발전하여 스팸분류기, 문자, 음성인식 등 학습형 알고리즘에 활용

나무위키(추가): 베이스 통계학(사후확률을 추론하는 베이스정리를 이용해 통계학의 문제 접근 흐름)
치명적인 장애물은 사후분포를 사람의 손으로 계산하기 쉽지 않은 케이스들이 많다는 것. 예를들어 사전분포와 가능도가 특정한 짝을 이루고 있다면, 여기서 추출되는 사후분포는 사전분포와 동일한 형태를 갖는데 이러한 사전분포를 공액사전분포, 켤레사전분포(conjugate prior)라고 한다.
ex) 사전분포와 가능도가 정규분포를 따를 때 사후분포는 동일한 정규분포
사전분포와 가능도가 각각 베타분포와 이항분포(또는 이것의 특수한 사례로서의 베르누이 분포)를 따른다면,
사후분포는 사전분포와 동일한 베타분포가 된다.
이러한 경우에는 정해진 업데이트 공식을 계산하면 쉽게 사후분포를 유도할 수 있어 베이즈 통계학 아이디어가 발견된 이후 줄곧 사용되었다(?)
하지만 현실에서 데이터는 사전분포와 가능도가 잘 매칭되지 않는다. 데이터가 많아짐에 따라 사후확률의 계산을 몬테카를로 방법, 변분법을 이용한 Variational inference같은 기법으로 있어 베이스주의의 데이터 분석이 발전

- 나이브의 의미: 실제로 속성들간에 독립을 가정하는 단순한 접근 방식이라는 의미.
독립성 가정은 정확한 확률은 구할 수 없어도 계산이 쉬워짐
실제로 스팸 구분을 위해서는 메일 발신인, 시간, 본문 내용과같은 더 많은 속성 구분에 영향을 줌

* 라플라스 에스티메이터를 쓰는 이유: 어떤 한 사건에 대한 몇 개의 속성의 사전확률이 데이터가 부족함 등의 이유로 사전확률중 하나의 속성이 0일 경우 사후확률도 0으로 계산되는 것을 방지하기 위함.
(블로그)
학습데이터에는 없고 검사데이터에서는 있는 범주(category)에서 확률이 0이 되어 정상적인 예측이 불가능한 “zero frequency" 인 것. 이 이슈를 피하기 위해 smoothing technique를 필요로 하며 laplace estimator가 대표적인 기법.

- Using numeric features with Naive Bayes: 양적변수를 이산형으로 바꿔줘야한다 케이스는 적절하게

$$P(\text{Spam} \mid W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)$$

여기서, $P(W_1 \cap !W_2 \cap !W_3 \cap W_4 | spam) \xrightarrow{naive} P(W_1|spam)P(!W_2|spam)P(!W_3|spam)P(W_4|spam)$

$P(W_1 \cap !W_2 \cap !W_3 \cap W_4 | ham) \xrightarrow{naive} P(W_1|ham)P(!W_2|ham)P(!W_3|ham)P(W_4|ham)$

	Viagra (W_1)		Money (W_2)		Groceries (W_3)		Unsubscribe (W_4)		
Likelihood	Yes	No	Yes	No	Yes	No	Yes	No	Total
spam	4 / 20	16 / 20	10 / 20	10 / 20	0 / 20	20 / 20	12 / 20	8 / 20	20
ham	1 / 80	79 / 80	14 / 80	66 / 80	8 / 80	71 / 80	23 / 80	57 / 80	80
Total	5 / 100	95 / 100	24 / 100	76 / 100	8 / 100	91 / 100	35 / 100	65 / 100	100

$$= \frac{\frac{4}{20}\cdot\frac{10}{20}\cdot\frac{20}{20}\cdot\frac{12}{20}\cdot\frac{20}{100}}{\frac{4}{20}\cdot\frac{10}{20}\cdot\frac{20}{20}\cdot\frac{12}{20}\cdot\frac{20}{100}+\frac{1}{80}\cdot\frac{66}{80}\cdot\frac{71}{80}\cdot\frac{23}{80}\cdot\frac{80}{100}} = \frac{0.012}{0.014} = 0.857$$

사전과제 사용알고리즘:
MultinomialNB: 다항 이벤트 모델에서는, 샘플(특성 벡터)들은 다항분포 (p1,...pn)에 의해 생성된 어떤 이벤트의 빈도수를 나타낸다 . (pi는 해당 이벤트가 발생할 확률)
BernoulliNB: 이항분포에 맞춘 것 같음
ComplementNB: 클래스가 imbalance할 때 하는 multinomial NB
cf) 가우시안나이브베이즈: 연속적 값의 데이터를 처리할 때 각 크래스의 연속적 값들이 가우스 분포를 따른다고 가정해 분류 뒤에 나옴.

Linear Regression 선대느낌으로.

보통 제약조건이 더 많은 것이 우리가 주어진 데이터와 그 것을 위한 line fitting하는 것이 기본인 ML에서 Ax = b를 만족하는 x벡터들 즉 파라미터들은 없으니, || Ax - b ||를 최소화하는 제일 적절한 Column Space의 솔루션을 찾자. 컬럼으로 보는 것은 어떻게 보면 featuere들로 보는 것. 아래로

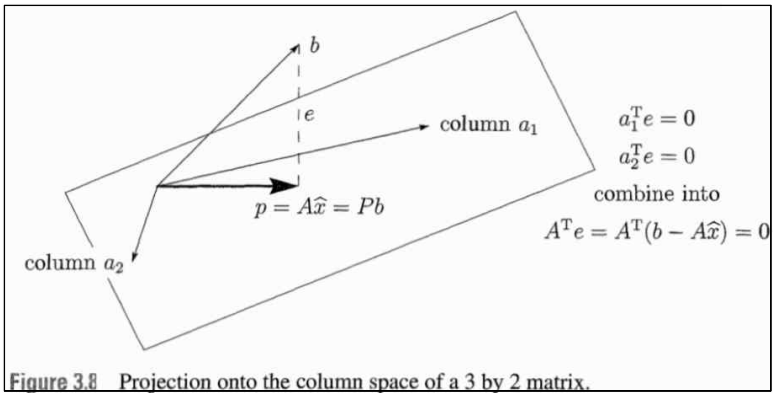


Figure 3.8 Projection onto the column space of a 3 by 2 matrix.

방법 1. 선대 맛 수직임을 이용

Finding $\hat{x}$ and the projection is so fundamental the we do it in 2 ways:

1) The error vector must be orthogonal to each column vector

$$\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \left[b - A\hat{x} \right] = 0$$

2) Since Column space is perpendicular to left null space, The error vector ($e = b - A\hat{x}$ must be in the nullspace of A^T (left null space).

$A^T(b - A\hat{x}) = 0$ or $A^T A\hat{x} = A^T b$

- Normal form ($A^T A$의 역행렬이 항상 존재하지 않으니) : $A^T A\hat{x} = A^T b$
- Best estimate : $\hat{x} = (A^T A)^{-1} A^T b$ and $(A^T A)^{-1}$ can be pseudo inverse
- Projectioin Matrix: $\mathbf{P} = A(A^T A)^{-1} A^T$, Point:
  $p = A\hat{x} = \mathbf{P}b = A(A^T A)^{-1} A^T b$

방법 2. 편미분

$y_1 = ax_1 + b,\ y_2 = ax_2 + b,\ \ldots y_n = ax_n + b$ 에서 error를 최소화 하는 직선 a,b

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

error의 형태는 Ax-b의 형태이므로

$$\begin{bmatrix} ax_1 + b - y_1 \\ \vdots \\ ax_n + b - y_n \end{bmatrix} \rightarrow \|Ax - b\|^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

또는 미분을 활용해 $\|Ax - b\|^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2 = f(a, b)$ 이니까.

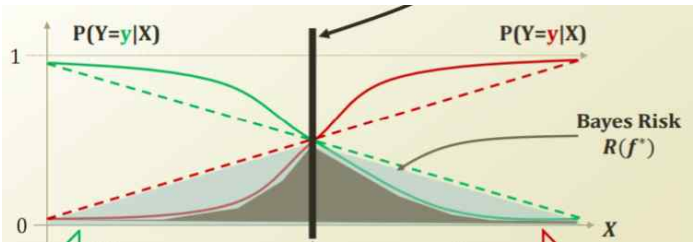
각각의 편미분을 활용해, $\partial f/\partial a = 0, \partial f/\partial b = 0$ a,b를 구해도 결과는 위와 같다.

from MIC

- To make the hypothesis better, we need to find the better $\boldsymbol{\theta}$
  - $h: \hat{f}(x;\theta) = \sum_{i=0}^{n} \theta_i x_i \rightarrow \hat{f} = X\theta$
  - $X = \begin{pmatrix} 1 & \cdots & x_n^D \\ \vdots & \ddots & \vdots \\ 1 & \cdots & x_n^D \end{pmatrix}, \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{pmatrix}$
- The reality would be the noisy, so....
  - $f(x;\theta) = \sum_{i=0}^{n} \theta_i x_i + e = y \rightarrow f = X\theta + e = Y$
- The difference is the error from the noise, so let's make it minimum
  - $\hat{\theta} = argmin_\theta (f - \hat{f})^2 = argmin_\theta (Y - X\theta)^2$
    $= argmin_\theta (Y - X\theta)^T (Y - X\theta) = argmin_\theta (Y - X\theta)^T (Y - X\theta)$
    $= argmin_\theta (\theta^T X^T X\theta - 2\theta^T X^T Y + Y^T Y) = argmin_\theta (\theta^T X^T X\theta - 2\theta^T X^T Y)$
- Now, we need to optimize $\boldsymbol{\theta}$

- $\hat{\theta}$
  $= argmin_\theta (\theta^T X^T X\theta - 2\theta^T X^T Y)$
- Same technique as in Thumbtack
  - $\nabla_\theta (\theta^T X^T X\theta - 2\theta^T X^T Y) = 0$
  - $2X^T X\theta - 2X^T Y = 0$
  - $\theta = (X^T X)^{-1} X^T Y$

Logistic Regression
나이브 베이즈처럼 naive assumption을 안하는 것.
로지스틱과 리니어 리그레션으로의 분류의 가장 큰 차이는 로지스틱은 Bayes Risk를 줄인다는 것. 성능 차이로 직결
디시전 바운더리 근처에서 샤프한 posterior 분포를 관측하기 위해



어떻게 fitting하냐 logit함수를 이용해 feature값을 확률로 변환한 후에,

$$P(y=1|x)=\mu(x)=\frac{1}{1+e^{-\dot{\theta}^T x}}=\frac{e^{X\theta}}{1+e^{X\theta}} \qquad X\theta=\log\left(\frac{P(Y|X)}{1-P(Y|X)}\right)$$

이를 파라미터 세타에 대해 MLE를 하려고하면,
Linear Regression과 다르게 closed form solution이 아니라 open form이여서 approximate해야 한다!
그리고 여기서 gradient descent개념을 이용하여 방향(방향은 어떻게 경사가 내려가는 반대방향 해당 데이터의 미분값)을 정해 세타에 대해 gradient ascent를 활용해 MLE로 세타를 추정해 PAC function을 구성한다.

- 선형회귀에서도 X가 너무 클때는 역행렬을 구하기 너무 힘들어 Gradient Method를 이용해 Approximation한다.

로지스틱 regression에서 loss function으로 최소제곱법을 실제로 사용하지 않는다.
L(yhat, y) = 1/2 (yhat - y)^2를 가르칠 수 있지만 로지스틱회귀에서 잘 안쓴다. Loss를 이렇게 설정하면 함수가 볼록하지 않아 최적화 문제에서 경사하강법을 사용할 때 로컬옵티멈에 빠질 수 있어서, 볼록한 것을 하기 위해 크로스엔트로피를 사용한다. (앤드류 응)

- 나이브 베이즈와 로지스틱리그레션의 관계: 제너러티브, 디스크리미너티브 페어의 관계!

가우시안 나이브 베이즈: feature 들이 categorical에서 continuous로 -> 그러면 그 분포가 가우시안으로 가정하고 진행하자. 그러면 가우시안 파라미터를 학습하는 것으로 바꿔 생각하자

$$P(Y=y|X)=\frac{P(X|Y=y)P(Y=y)}{P(X)}$$

여기서 P(Y=y|X)를 바로구하려고 한 것이 로지스틱 likelihood와 prior까지 고려한게 GNB
MLE와 MAP와 유사하다. 또 이것이 Discriminative , Generative
GNB를 Logistic Regress로 유도해보자.
가정) 각 feature의 분산이 두 클래스에 대해 같다고 가정하면

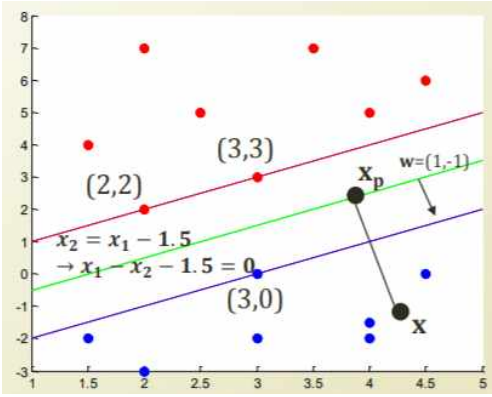
GNB는 각 feature의 분산이 두 클래스에 대해 같다고 가정하면 == Logistic Regress로 동일하다.

근데 여기까지 오려고 가정을 몇 개 했냐, 나이브베이즈에서 가정, Same Var 가정, 가우시안분포(P(X|Y)), 베르누이 분포 가정(P(Y)). 따라서 GNB의 파라미터가 Logistic Regress보다 파라미터 수는 더 많다.

무엇이 더 좋은가? 로지스틱이 더 좋아 보이지만 (정확도도 같은 수준으로 노력에 비해 더 좋지만) prior를 추가할점 및 나머지 좋은 점은 PRML에서 더 볼 수 있음 Generative , Discriminative 모델들 비교 부분
결정이론:
https://github.com/NamSahng/Summary/blob/master/Pattern_Recognition%26Machine_Learning/1.5%20Decision%20Theory.ipynb

SVM

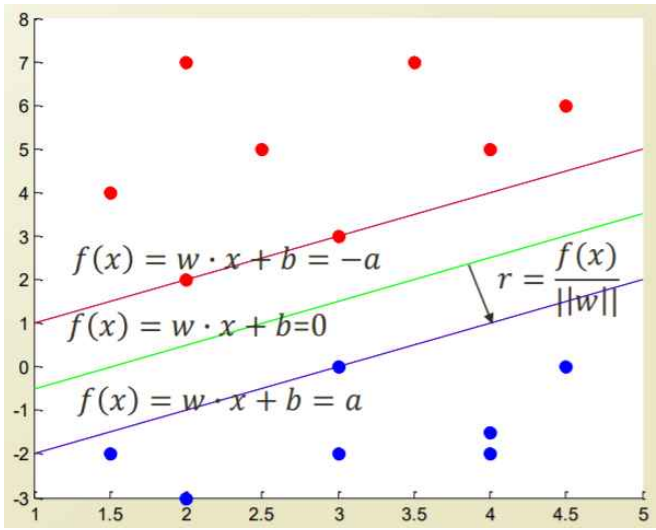
위에 보았듯이 디시전 바운드리가 얼마나 중요한지 알 수 있다.
확률적 관점의 모델은 아니다.
Decision Boundary를 Supporting 하는 점을 찾아 Margin이 가장 크도록 (Margin: SVM직선과 점의 거리)



projection을 이용해 양음으로 나뉘도록.

- Optimization problem?
  - $max_{w,b} 2r = \frac{2a}{||w||}$
    $s.t. (wx_j + b)y_j \geq a, \forall j$

s.t.조건. j = learning instance
a는 constant로 그냥 정하는 것이므로



maximization 문제를 역수를 취해

- $min_{w,b} ||w||$
  $s.t. (wx_j + b)y_j \geq 1, \forall j$

||w||이 sqrt(w1^2+w2^2)이므로 quadratic 문제가 된다.
quadratic optimizer를 이용해 한다.
그리고 loss는 힌지 로쓰라고 한다. squared hinge loss

constraint에 맞는 constraint를 찾을 수 있을 수도 있지만 못할 수도 있다.
빨간 점 하나가 아래 있으면 위의 s.t.조건이 만족을 못해 해가 없음. 그래서 hard margin의 단점 해결하기 위해

a. soft margin(penalization : 반칙 정도를 주어서)

Opt1) 0-1 Loss , C는 임의 상수 // 인기 없, quadratic 풀기 어렵

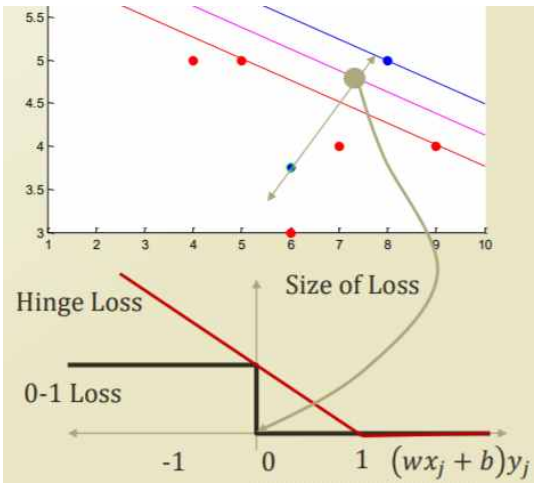
- $min_{w,b} ||w|| + C \times \#_{error}$
  $s.t. (wx_j + b)y_j \geq 1, \forall j$

먼거와 가까운게 동일한게 말이안됨

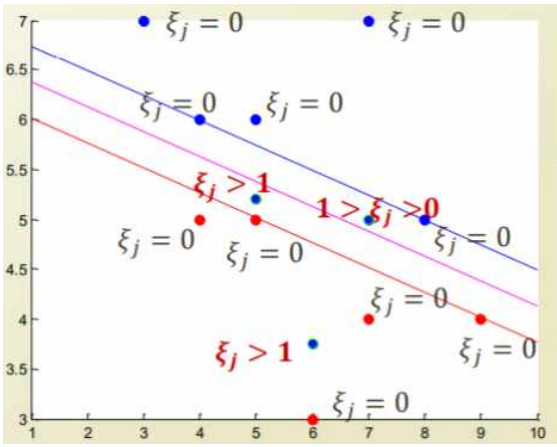
Opt2) Hinge Loss : 그래서 이걸 마이쓤

- Introduce a slack variable
  - $\xi_j > 1$ when mis-classified
- $min_{w,b} ||w|| + C \sum_j \xi_j$
  $s.t. (wx_j + b)y_j \geq 1 - \xi_j, \forall j$
  $\xi_j \geq 0, \forall j$

slack var.은 mis되는 정도
C는 그 강도 파라미러



장점은 quadratic prob이 안변함, but 하이퍼파라미터가 늘음



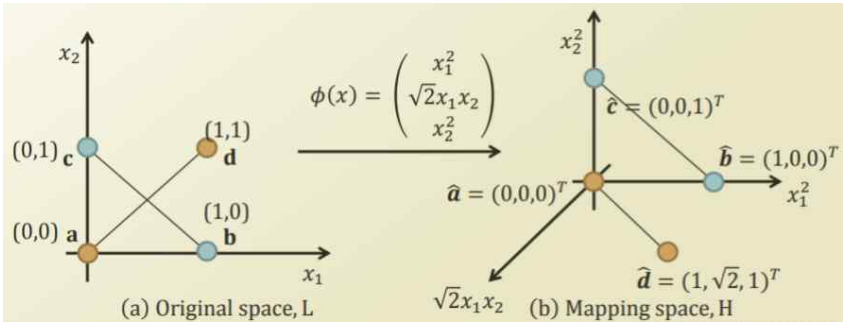
(로그로쓰와 힌지 로쓰 비교해 보면 로그로쓰는 맞는 것에서도 로쓰를 조금줌)
힌지를 쓸 때 C를 어케 결정하는지, 이거에 따라 성능이 달라짐

b. kernel trick: Decision boundary를 Complex하게 하자. 이는 데이터가 complex하니까 하는 거다.
차원을 뻥튀기 하는 거다, 이걸 좀더 스마트하게, 무한대까지 늘리는 방법을 알아보자.

$$\varphi(< x_1, x_2 >) = < x_1, x_2, {x_1}^2, {x_2}^2, x_1x_2, {x_1}^3, {x_2}^3, {x_1}^2x_2, x_1{x_2}^2 >$$

커널 이면에는 옵티메이젼의 성질을 이해해야하는데 프라이멀 문제를 듀얼로 바꿔야하는 것을 알아야한다.
라그랑지 멀티플라이어를 사용해 듀얼로 바꾼다. 그리고 strong duality를 만족해주어야 해가 같다는 보장이 되며 이것이 KKT 컨디션을 만족해야한다. ㅇㅋ;;
어쨌든 이런 과정이 결국 W에서 라그랑지 멀티플라이어 알파를 최적화 문제로 바뀐다. (그리고 이것도 쿼드라틱)

또 어쨌든 이렇게 알파로 보내는 것의 장점으로는 기존의 XOR문제를 다른 차원으로 보내면 선형으로 풀 수 있으니 선형으로 문제를 풀 수 있다. 하지만 이는 차원이 늘어나(Feature)가 좋다고만은 할 수 없다.

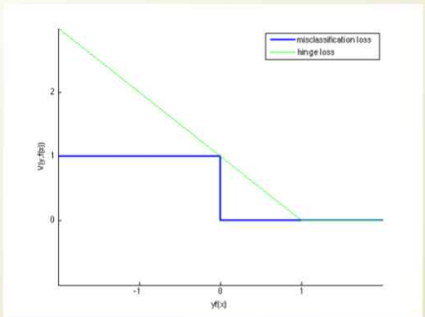


따라서 커널은 원래 다른 차원(더 많은 차원)으로 보내서 내적을 하는 것인데 이는 내적을 하고 다른 차원(더 많은 차원)으로 보내는 것이 같게 설계해서 (수직으로 설계해서인가?) 무한한 차원으로 보낼 수 있고, 계산도 적게 할 수 있는 것 같다. 이는 높은 정보의 차원을 쉽게 다루게 해준다.

w는 프사이로 남아있어서 다른 차원으로 보내는 데 인풋들을 넣어서 구하거나 그릴 수는 없게 되었다. 분류는 가능.

- Regularization and SVM
사실 SVM에는 C라는 term이 Regularization역할을 한다.

$$f = \arg\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(x_i)) + \lambda ||f||_{\mathcal{H}}^2 \right\}$$



$$V(y_i, f(x_i)) = (1 - yf(x))_+$$

$$(s)_+ = max(s, 0)$$

$$f = \arg\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (1 - yf(x))_+ + \lambda ||f||_{\mathcal{H}}^2 \right\}$$

$$f = \arg\min_{f \in \mathcal{H}} \left\{ C \sum_{i=1}^{n} (1 - yf(x))_+ + \frac{1}{2} ||f||_{\mathcal{H}}^2 \right\}$$

$$C = \frac{1}{2\lambda n}$$

- Support vector is a special case of regularization with the hinge loss

확률론

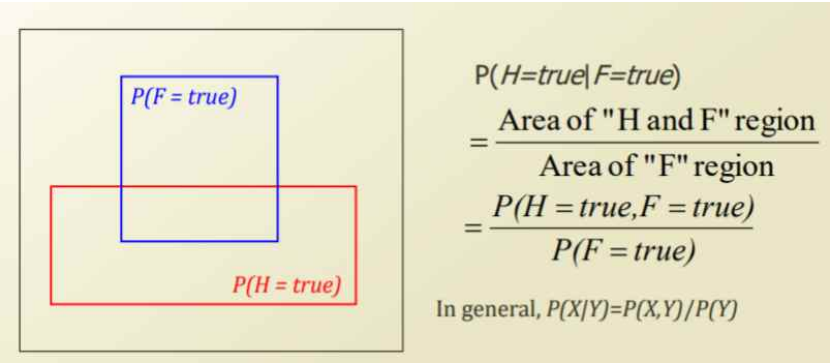
확률이란:

Frequentistic View: It is the relative frequency with which an outcome would be obtained if the process were repeated a large number of times under similar conditions.

Bayesian View: Probability is your degree of belief in an outcome.

확률 자체로 사용보다 조건부 확률, 무엇이 Given 일 때의 확률을 더 많이 사용한다.

결합확률 또한 중요한데 이는 많은 정보를 주기 때문



* Law of Total Prob.

- Law of Total Probability
  - a.k.a "summing out" or marginalization
  - $P(a) = \sum_b P(a, b) = \sum_b P(a \mid b)\, P(b)$
  - When B is any random variable

Joint를 알면 Marginalization을 통해 개별에 대해서 알 수 있다.

- Also, consider this case
  - given a joint distribution (e.g., P(a,b,c,d))
  - We can obtain any conditional probability of interest
  - $P(c \mid b) = \sum_a \sum_d P(a, c, d \mid b) = 1/P(b) \sum_a \sum_d P(a, c, d, b)$
  - Where 1 / P(b) is just a normalization constant

즉 joint를 알면 개별 individual prob이나 conditional prob도 알 수 있지만

근데 지수적으로 알아야하는 값들이 늘어남 나이브 베이즈 때와 비슷한 얘기 했다.

* Chain Rule & Factorization

- We can always write
  - $P(a, b, c, \ldots z) = P(a \mid b, c, \ldots. z)\, P(b, c, \ldots z)$
  - by definition of joint probability
- Repeatedly applying this idea, we can write
  - $P(a, b, c, \ldots z) = P(a \mid b, c, \ldots. z)\, P(b \mid c, .. z)\, P(c \mid .. z) .. P(z)$

우변 뒤쪽으로 나누면 정의

이렇게 펙토라이즈 가능.

- Variables *A* and *B* are independent if any of the following hold:
  - P(A | B) = P(A)
    - P(A,B) = P(A)P(B)
    - P(B|A) = P(B)

원래 동전도 앞 뒤 상황 상관 없이 독립이라 쉽게 확률을 계산 할 수 있었다.

이것이 Marginal independence의 예시 이며, 아래와 같으면 아니다. 조건부 독립은 나이브 베이즈!

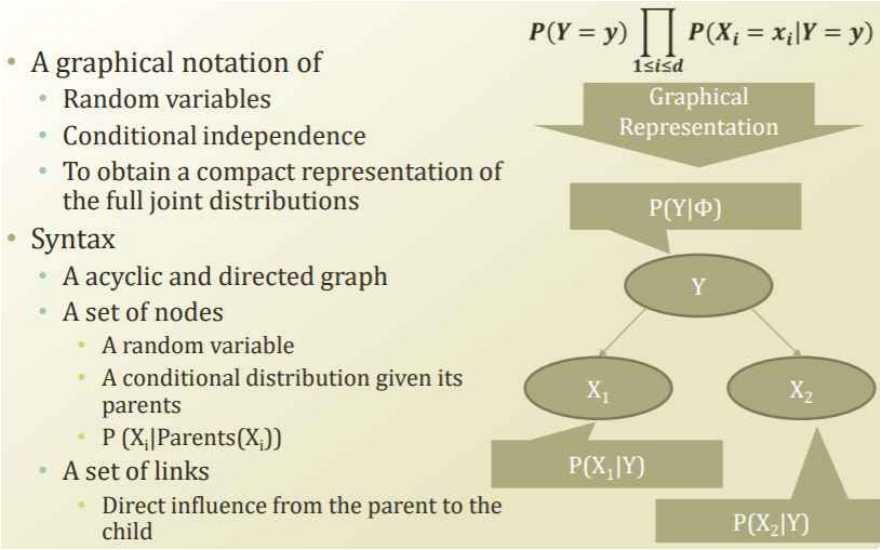
- P(OfficerA=Go|OfficerB=Go) > P(OfficerA=Go)
- **This is not marginally independent!**

- Conditional independence
  - P(OfficerA=Go|OfficerB=Go,Commander=Go) =P(OfficerA=Go|Commander=Go)
  - **This is conditionally independent!**

Bayesian Network

나이브 베이즈도 베이지안 넷웍이였고, 최신모델들도 BN이 많다?

BN은 Random Variables, 조건부 독립 등의 graphical Notation이며 전체 결합확률을 컴펙트하게 보여준다!



참고 ##

베이지안 네트워크는 Directed Ayclic Graph 이다. 나이브 베이즈, k means, gmm, hmm이 베이지안 네트워크의 예시들이다. 로지스틱 리그레션은 Undirected Graphical 모델이며 이는 CRF의 제한된 버전이다. 로지스틱회귀가 하나의 NN의 레이어의 퍼셉트론이다. 그리고 조경현 교수님은 NN도 DAG의 관점으로 보는 것이 유리하다고 말한다.

https://stats.stackexchange.com/questions/94511/difference-between-bayes-network-neural-network-decision-tree-and-petri-nets

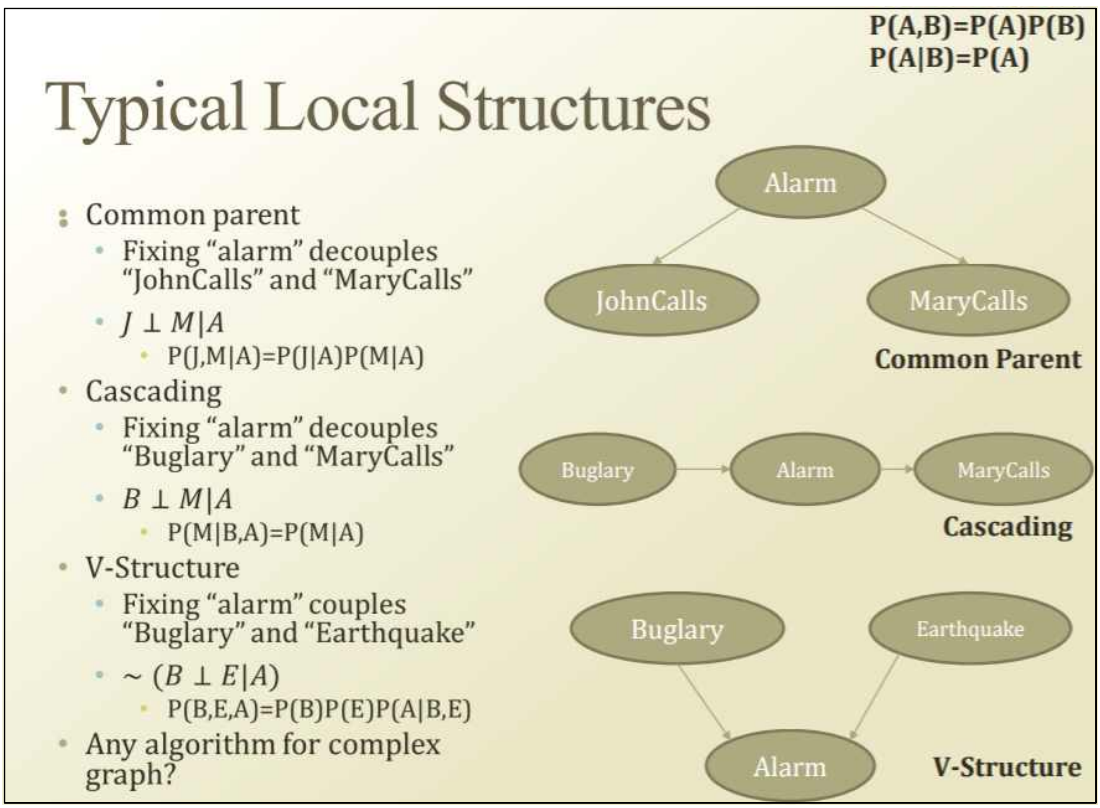
https://github.com/NamSahng/Summary/blob/master/NLP%26DL_CKH/Basics-HypothesisSet.ipynb

##

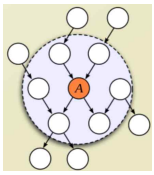
조건부 독립이 이미 가정되어있다! Encoded

Common parent와 Cascading의 관계의 경우에는 A를 알면 독립이 되는데

V-Structure에서는 특정 관계가 생긴다!



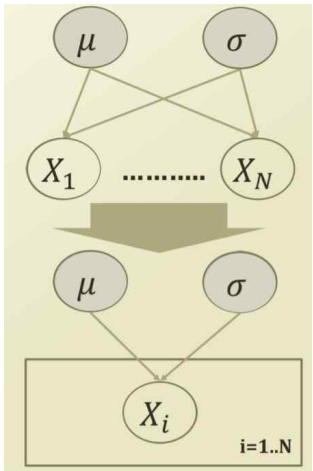
- 베이즈볼: 독립을 판정할 때 사용. 위와 같이 V 구조만 안 굴러감.
- 마르코프 담요 (Markov Blanket)

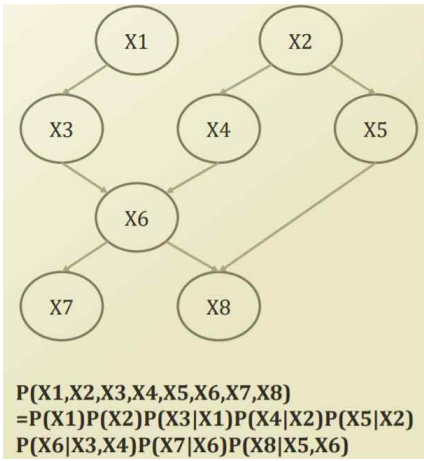


P(A|blanket, B)=P(A|blanket)
Blanket={parents, children, children's other parents}
Random variable A에 대하여 다른 12개의 랜덤 버라이어블에 대해 6개만 알면(given)이면 conditional independent가 만족된다. (D-Separation도 비슷한 내용이다.)
- Ransdom variable(확률변수): 표본 공간에 일정한 화귤릉ㄹ 가지고 발생하는 사건에 수치를 일대일 대응한 함수
'주사위에서는 주사위를 한번 던질 때 나오는 숫자'

- Factorization of Bayesian Network
조건부 독립이 인정? 확인 이되면 , 결합확률은 빡세니 팩토라이즈하면(그냥 정리에 의해 할 수 있는 거니까 이건)
P(a, b, c, … z) = P(a | b, c, …. z) P(b | c,.. z) P(c| .. z)..P(z) 여기서 중간 것들을 간단히 할 수 있다.
ex) P(a | b, c, d) = P(a | b) -> 알아야하는 파라미터 개줄음 (파라미터 8개에서 2개 다 바이너리이면)

그래서 BN에서 보면 이렇게 줄은걸 볼 수 있다.



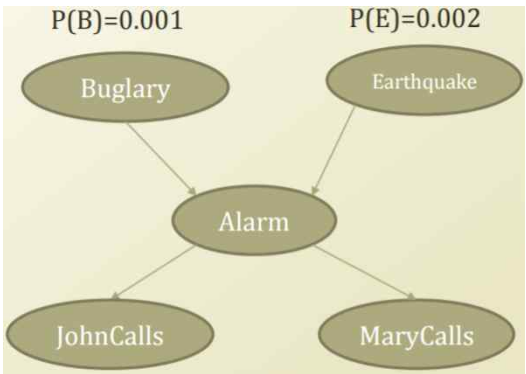


- Plate Notation: 간략하게 표현 한다.

해보자
Inference Question 1: Likelihood
P(B=true, MC=true)=?

Inference Question 2: Conditional Probability
P(A|B=true, MC=true)=?

Inference Question 3: Most Probable Assignment
argmax_aP(A|B=true, MC=true)=?



B	E	P(A|B,E)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

A	P(J|A)
T	0.90
F	0.05

A	P(M|A)
T	0.70
F	0.01

- General form
  - $P(x_V) = \sum_{X_H} P(X_H, X_V)$
  $= \sum_{x_1} \dots \sum_{x_k} P(x_1 \dots x_k, x_V)$
  - Likelihood of x_V

- General form
  - $P(Y|x_V) = \sum_z P(Y, Z = z|x_V)$
  $= \sum_z \frac{P(Y, Z, x_V)}{P(x_V)}$
  $= \sum_z \frac{P(Y, Z, x_V)}{\sum_{y,z} P(Y = y, Z = z, x_V)}$
  - Conditional probability of Y given x_V

P(B, E| A) -> 진단!
P(A | B , E) -> 예측!
- Applications of *a posteriori*
  - Prediction
    - B, E → A
  - Diagnosis
    - A → B, E

- Marginalization and Elimination

- $P(a=true,b=true,mc=true)=\sum_{JC}\sum_{E}P(a,b,E,JC,mc)$

$$=\sum_{JC}\sum_{E}P(JC|a)P(mc|a)P(a|b,E)P(E)P(b)$$

이건 계산이 너무 빡세니

$$P(a,b,mc)=\sum_{JC}\sum_{E}P(a,b,E,JC,mc)$$

$$=P(b)P(mc|a)\sum_{JC}P(JC|a)\sum_{E}P(a|b,E)P(E)$$

이렇게 해야하며 이게 훨 쉽게해줌 계산을

- Preliminary
  - $P(e|jc,mc)=\alpha P(e,jc,mc)$
- Joint probability (e=jc=mc=true)
  - $P(e,jc,mc,B,A)=$ $\alpha P(e)\sum_B P(b)\sum_A P(a|b,e)\,P(jc|a)P(mc|a)$
    - Line up the terms by the topological order
    - Consider a probability distribution as a function
      - $f_E(E=t)=0.002$
  - $=\alpha f_E(e)\sum_B f_B(b)\sum_A f_A(a,b,e)\,f_J(a)f_M(a)$

A	$f_{JM}(A)$
T	0.63
F	0.0005

A	$f_J(A)$
T	0.90
F	0.05

A	$f_M(A)$
T	0.70
F	0.01

  - $=\alpha f_E(e)\sum_B f_B(b)\sum_A f_A(a,b,e)\,f_{JM}(a)$

- $=\alpha f_E(e)\sum_B f_B(b)\sum_A f_{AJM}(a,b,e)$
- $=\alpha f_E(e)\sum_B f_B(b)\,f_{\bar{A}JM}(b,e)$
- $=\alpha f_E(e)\sum_B f_{B\bar{A}JM}(b,e)$
- $=\alpha f_E(e)f_{\bar{B}\bar{A}JM}(e)$
- $=\alpha f_{E\bar{B}\bar{A}JM}(e)$

B	E	$f_{\bar{A}JM}(B,E)$
T	T	0.95*0.63+0.05*0.0005
T	F	0.94*0.63+0.06*0.0005
F	T	0.29*0.63+0.71*0.0005
F	F	0.001*0.63+0.999*0.0005

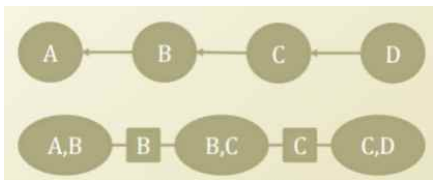
이렇게 만들고 푸는 것이다.

왜 function notation?
조건부와 같은 것들의 의미를 없애 계산을 쉽게하려고!

- Potential Function

probabilistic graphical models로 만들고 싶은데 아직 그렇게 못한다. 잠재력은 있지만 아직 아닌
P(A,B,C,D) = P(A|B)P(B|C)P(C|D)P(D) 인 거에서
맨 오른쪽과 같이 만들고,
오른쪽과 같이 함수를 정의해준다.

- Potential function on nodes
  - $\psi(a,b),\psi(b,c),\psi(c,d)$
- Potential function on links
  - $\phi(b),\phi(c)$



프사이와 파이를 요렇게하면
맨 위식이 만족이 된다.

- $P(A,B,C,D)=P(U)=\frac{\prod_N\psi(N)}{\prod_L\phi(L)}=\frac{\psi(a,b)\psi(b,c)\psi(c,d)}{\phi(b)\phi(c)}$
  - $\psi(a,b)=P(A|B),\psi(b,c)=P(B|C),\psi(c,d)=P(C|D)P(D)$
  - $\phi(b)=1,\phi(c)=1$
- $P(A,B,C,D)=P(U)=\frac{\prod_N\psi(N)}{\prod_L\phi(L)}=\frac{\psi^*(a,b)\psi^*(b,c)\psi^*(c,d)}{\phi^*(b)\phi^*(c)}$
  - $\psi^*(a,b)=P(A,B),\psi^*(b,c)=P(B,C),\psi^*(c,d)=P(C,D)$
  - $\phi^*(b)=P(B),\phi^*(c)=$P(C)

또 다른 방법은 이렇게 할 수 있다.
얘는 근데 결합확률로 잡아서 ㅈ고ㅁ
문제가 있다.

(cf) 또 마지날라이제이션도 정의할 수 있다.)

- Absorption in Clique Graph: 위에걸 발전해보자.
업솔션은 일종의 오퍼레이션을 이용해 인퍼런스 할 것이다.
오른쪽과 같은 걸 가정해보자.

- Let's assume
  - $P(B) = \sum_A \psi(A,B)$
  - $P(B) = \sum_C \psi(B,C)$
  - $P(B) = \phi(B)$

- How to find out the ψs and the ϕs?
  - When the ψs change by the observations: P(A,B) → P(A=1,B)
  - A single ψ change can result in the change of multiple ψs
  - The effect of the observation propagates through the clique graph
  - Belief propagation!

Why does t

특정 프사이가 (a,b 여기서) evidenc random variable로 바뀌면 belief가 바뀌어서 BELIEF PROPA가 된다. 변화하면서 전체 프사이가 바뀌는 결과. 관측이 클리프 그래프로 바뀌는 것 이것이 BELIEF PROPA!

- How to propagate the belief?
  - Absorption (update) rule
  - Assume $\psi^*(A,B)$,$\psi(B,C)$, and $\phi(B)$
  - Define the update rule for separators
    - $\phi^*(B) = \sum_A \psi^*(A,B)$
  - Define the update rule for cliques
    - $\psi^*(B,C) = \psi(B,C)\frac{\phi^*(B)}{\phi(B)}$

어떻게 하는거지? Absorption Rule에 의해

과정들 스타들로 바뀔 것이다. Absorption Rule에 의해
Separator는 이렇게 클리크는 아래처럼 업데이트 된다.

왜 이게 그러는지는 노트를 봐라 시간 없다.
- Local consistency 가 되는 간단한 예시도 노트 나중에 강의 다시 ㄱㄱ

cf)
생성모델은 전체적인 입력공간 분포를 모델링합니다. Unsupervised learning이 가능하고(P(x) 학습) supervised learning도 가능합니다.(P(x|y) 학습). Sample의 classification 뿐 아니라 생성 및 변환도 할 수 있습니다.
판별모델은 category간 경계를 주로 모델링하고 대부분 supervised learning (또는 semi-supervised learning)에 의해 학습합니다. 보통 classification이나 regression에 사용됩니다.
생성 모델은 패턴의 분포 전체를 학습하는 반면, 판별 모델은 category 구분과 무관한 특징은 무시하는 경향이 있습니다. 그래서 생성모델을 "전체 모습을 기억하는 화가의 눈"에 판별모델을 "적군/아군 구분에 최적화된 보초병의 눈"에 비유할 수 있습니다.
판별모델은 유사패턴을 더 잘 구분하는 반면, outlier에 약합니다. 생성모델은 unsupervised learning이 가능하다는 장점과 함께 overfitting이 적게 발생하는 장점도 있습니다.
GAN의 generator는 좀 특별합니다. FFN(Feed Forward Network)으로 random noise를 sample으로 변환하는데, 구조는 판별모델이고 역할은 생성모델입니다.

K-means
cf) 언수퍼바이즈드는 클러스터링, 필터링으로 크게 2개로 (차원축소도 맞지 않나?) 나뉘며, CRF, MRF는 필터링이다.

필터링은 신호와 잡음의 혼합으로부터 기초적인 신호와 기초적인 신호를 추정하는 것.

클러스터링: Latent Factor에 대한 Optimal Assignment!

K-means : Latent Factor가 k개 되겠다. 그리고 두 단계의 문제를 풀어야함 센트로이드 찾고 점을 어싸인.

- $J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}||x_n - \mu_k||^2$
- Minimize J by optimizing
  - r_{nk}: the assignment of data points to clusters
  - μ_k: the location of centroids

rnk와 뮤k를 알아 내야한다.

- EM 알고리즘!

  Expectation : Expectation of the log-likelihood given the parameters
  Assign the data points to the nearest centroid (옵티마이즈 알엔케이)
  완성되지 않은 센트로이드에 어싸인

  Maximization : Maximization of the parameters with respect to the likelihood
  Update the centroid positions given assignment (센트로이드를 포지션을 다시 계산 어싸인)
  rnk를 기준으로 다시 센트로이드 변경 이거 두 개를 반복

미분, = 0로 계산함 뮤케이는 그러면 평균값이 된다. 거리들을 계산하면 로컬 옵티마에 빠질 가능성이 있다.
- 사용이유 : 당시, 알고 있던 가장 간단한 클러스터링 방법, 결과를 해석하기 쉽다고 판단.
- 단점:

  a. 클러스터 개수가 불확실하다: 베이지안 넘 파라미터를 통해 나중에 정해줄 수 있다.
  b. 센트로이드 초기값에 취약하다. c. 유클리디안 거리는 제한적이다. d. 하드클러스터링은 위험
  c와 d는 gmm으로 극복가능 // b와 a에 대해서는다른 다양한 테크닉은 있다.

GMM

- 다항분포 (Multinomial Distribution)

- X=(0,0,1,0,0,0) when K=6 and selecting the third option
- $\sum_k x_k = 1, P(X|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k{}^{x_k}$ *such that* $\mu_k \geq 0, \sum_k \mu_k = 1$

우식과 같이 X의 값이 지수로 올라가며 계산됨. 이항분포의 제너럴 버전

- $P(X|\boldsymbol{\mu}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mu_k{}^{x_{nk}} = \prod_{k=1}^{K} \mu_k{}^{\sum_{n=1}^{N} x_{nk}} = \prod_{k=1}^{K} \mu_k{}^{m_k}$
  - When $m_k = \sum_{n=1}^{N} x_{nk}$
  - Number of selecting k^{th} *option* out of N selections
- How to determine the maximum likelihood solution of $\boldsymbol{\mu}$?
  - Maximize $P(X|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k{}^{m_k}$
  - Subject to $\mu_k \geq 0, \sum_k \mu_k = 1$

이렇게 표현해

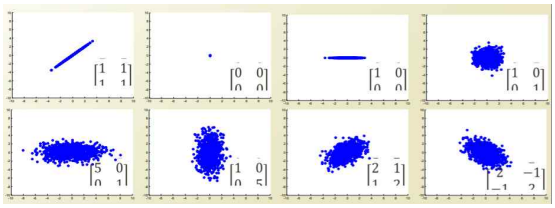
아래와 같이 뮤케이를 계산할 수 있다.

이때, 목적함수에 제약조건이 있는 MLE방법이므로 라그랑주 승수로 기릿. (SVM에서 & PRML 보충자료)
라그랑지 목적함수에 상수랑 올려 람다와 목적함수를 정리해 구한다.

그렇게 나온 결과를 해석해보면 다항분포에서의 MLE parameter는 (특정 선택지가 선택된 개수)/(선택지개수)
가 되며 이항분포랑 같은 개념으로 되는구나.

- 멀티베리에이트 가우시안 분포:

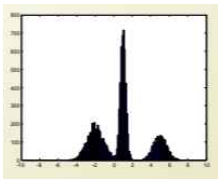
쭉 전개하고 trace trick을 사용하고 계산하여 평균과 분산을 MLE한다.
Covariance Matrix의 특성 -> correation과 연관이 있음
(선대에서도 나옴)

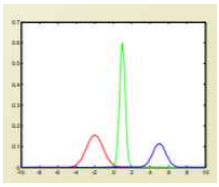


- Mixture Model: 위의 두 분포를 연결하자.

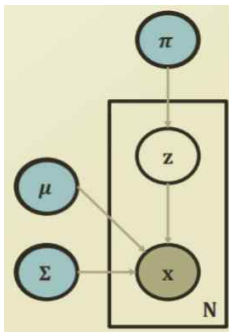
여기는 3개의 근원인 다변수 가우시안 분포에서 왔구나.

믹싱 코이피션트를 추가한다.





- $P(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \sigma_k)$
  - Mixing coefficients, π_k: A normal distribution is chosen out of K options with probability
    - Works as weighting
    - $\sum_{k=1}^{K} \pi_k = 1, 0 \le \pi_k \le 1$
    - This is a probability (as well as weighting!)
    - Then, which distribution?
    - New variable? Let's say Z!
  - Mixture component, $N(x|\mu_k, \sigma_k)$: A distribution for the subpopulation
- $P(x) = \sum_{k=1}^{K} P(z_k) P(x|z)$
  - Why this ordering of variables?



- 이제 GMM을 정의 해보자
그리고 위의 믹싱 코이피션트가 rnk와 비슷하지만 과정을 soft하게(확률로) 해준다.
파란색은 파라미터 형태로, x는 관측이 되어 찐한색

- 장점: soft하고, 더 많은 정보를 알 수 있다. latent factor에 대한 이해를 할 수 있다.
(Unsuper에서 super와의 Essense는 Latent Factor)
Latent factor에 대한 driving force를 더 잘 이해
- 단점: 코베리언스 모델링을 해야하는 등에 대한 계산량,
지역최소에 빠지는 위험 ,
K를 정해야하는(이는 GMM을 베이지안GMM으로 만들면 해결 가능)

- K-means와 비교:
k-means는 모든 데이터 포인트를 특정 센트로이드에 옵팀러하게 어싸인 해 거리를 최소화하는 식으로 했다면,
GMM은 확률이라는 프레임 워크안에 라이클리후드를 최대화하는 방향으로 파라미터를 옵티마이즈 하겠다.

Expectation : 완성되지 않은 분포(처음엔 랜덤한 분산, 평균, 믹싱코이피션트)에 데이터를 확률적으로 계산해 어싸인 (znk들 계산)
Maximization : 계산된 znk 값들에 대하여 파라미터 처음엔 랜덤한 분산, 평균, 믹싱코이피션트를 업데이트 미분과, 라그랑지를 사용해

GMM에서 covar. mat. = eI로 fix해보자: 그리고 e를 0으로 계속 보내 사고실험을 하면
znk가 도미넌트해진다. -> 하드 클러스터링으로 되는 것.
즉 Soft Assignment와 공분산 행렬이 가장 큰 차이임을 알 수 있다.

- Classification과 Clustering의 차이는 LATENT VARIABLE의 유무

- EM algorithm
  - Initialize θ^0 to an arbitrary point
  - Loop until the likelihood converges
    - Expectation step
      - Assign Z by $P(Z|X,\theta)$
      - $\gamma(z_{nk}) \equiv p(z_k = 1|x_n) = \frac{P(z_k=1)P(x|z_k=1)}{\sum_{j=1}^{K} P(z_j=1)P(x|z_j=1)} = \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_j, \Sigma_j)}$
    - Maximization step
      - Same optimization of ordinary MLE
      - $\frac{d}{d\mu_k} \ln P(X|\pi,\mu,\Sigma) = 0, \frac{d}{d\Sigma_k} \ln P(X|\pi,\mu,\Sigma) = 0, \frac{d}{d\pi_k} \ln P(X|\pi,\mu,\Sigma) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right) = 0$
      - $\widehat{\mu_k} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \widehat{\mu_k})(x_n - \widehat{\mu_k})^T}{\sum_{n=1}^{N} \gamma(z_{nk})}, \pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$

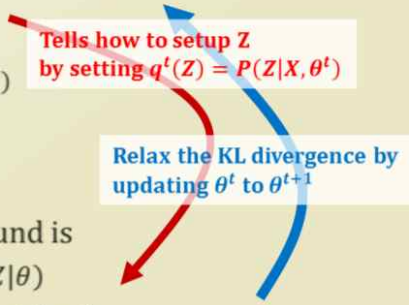
- EM:
분류에는 Z(latent)가 없는 상태로 했었지만 클러스터링에는 있다.

Z를 summation out해야 하는데 ln안에 있다. 어려워짐

- Difference between classification and clustering
- Let's say
  - {X,Z}: complete set of variables
  - X: observed variables
  - Z: hidden (latent) variables
  - θ: parameters for distributions
  - $P(X|\theta) = \sum_Z P(X,Z|\theta) \rightarrow \ln P(X|\theta) = \ln\{\sum_Z P(X,Z|\theta)\}$

- 이해를 위한 밑재료 옌센 부등식:
log는 concave함을 이용해 옌센 부등식을 이용해
(이 때 q(z)를 introduce 하여)
lower bound를 maximize하게 한다.
그리고 이를 활용하면 inequality를 만드는 factor를 찾고 이를 사라지게 해야겠다 생각할 수 있다.
그리고 그 factor는 KL Divergence의 형태와 유사.

- $l(\theta) = \ln P(X|\theta) = \ln\left\{\sum_Z q(Z)\frac{P(X,Z|\theta)}{q(Z)}\right\} \geq \sum_Z q(Z)\ln\frac{P(X,Z|\theta)}{q(Z)} = Q(\theta,q)$
  - $Q(\theta,q) = E_{q(Z)}\ln P(X,Z|\theta) + H(q)$
  - $L(\theta,q) = \ln P(X|\theta) - \sum_Z\{q(Z)\ln\frac{q(Z)}{P(Z|X,\theta)}\}$
- Why do we compute $L(\theta,q)$?
  - We do not know how to optimize $Q(\theta,q)$ without further knowledge of $q(Z)$
  - The second term of $L(\theta,q)$ tells how to set $q(Z)$
    - The first term is fixed when θ is fixed ***at time t***
    - The second term can be minimized to maximize $L(\theta,q)$
      - $KL(q(Z)||P(Z|X,\theta)) = 0 \rightarrow q^t(Z) = P(Z|X,\theta^t)$
  - Now, the lower bound with optimized q is
    - $Q(\theta,q^t) = E_{q^t(Z)}\ln P(X,Z|\theta^t) + H(q^t)$
- Then, optimizing θ to retrieve the tight lower bound is
  - $\theta^{t+1} = argmax_\theta Q(\theta,q^t) = argmax_\theta E_{q^t(Z)}\ln P(X,Z|\theta)$
    - $q^t(Z)$ → Distribution parameters for latent variable is at time t
    - $\ln P(X,Z|\theta)$ → optimized log likelihood parameters is at time t+1



그리고 보면 a. KL divergence를 활용해 Latent Variable의 확률 분포를 알아내는 step과
세타^t를 업데이트를 하는(KL.D가 다시 만족이 안되게 됨)는 step을
반족하는 것이 EM 알고리즘이다.

BUT EM은 지역최소값이나 saddle point에 빠질 수 있으며 실전에서는 괜찮게 한다.

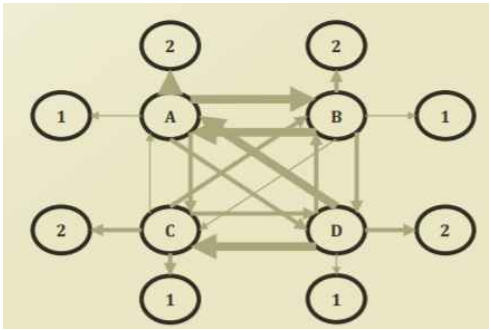
다시 정리
전 페이지에
EM GMM꺼

- EM algorithm
  - Initialize θ^0 to an arbitrary point
  - Loop until the likelihood converges
    - Expectation step
      - $q^{t+1}(z) = argmax_q Q(\theta^t,q) = argmax_q L(\theta^t,q) = argmin_q KL(q||P(Z|X,\theta^t))$
      - $\rightarrow q^t(z) = P(Z|X,\theta) \rightarrow$ Assign Z by $P(Z|X,\theta)$
    - Maximization step
      - $\theta^{t+1} = argmax_\theta Q(\theta,q^{t+1}) = argmax_\theta L(\theta,q^{t+1})$
      - → fixed Z means that there is no unobserved variables
      - → Same optimization of ordinary MLE

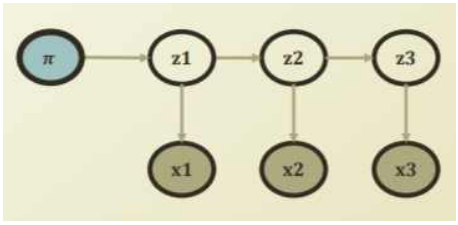
cf)
k 개수 정하는 방법: 실루엣: 같은 군집내의 원소들의 거리와 다른 군집과의 거리를 활용하는 지표
엘보우: SSE가 급격히 줄어드는 지점

HMM : GMM에서 시간의 간격을 두고 나올 때, Time이 들어갈 때의 방법! 나중에 더 다 공부!
- Dynamic Clustering이라고도 하며, pos tagger에 사용되기도 함. (discrete인 모델을 보면)

A,B,C,D 는 latent factor (상태들)
1,2,3,4 observation.



z: latent state



- Hmm의 Main Questions
파이 = initail state Prob (
a=Transition Prob z1 -> z2로 가는 관계 (이전 i번째 클러스터에서 j로 가는 확률)
b = 어떤 특정 state에서 observation이 나온 확률
(i번째 클러스터에 있을 때 j번째 관측을 보일 확률)
X = 관측데이터
M = topology of BN, HMM / HMM의 sturcture

- Transition probabilities
  - $P(z_t|z_{t-1}^i = 1) \sim Mult(a_{i,1}, \dots, a_{i,k})$
  - Or, $P\left(z_t^j = 1 \middle| z_{t-1}^i = 1\right) = a_{i,j}$

- Emission probabilities
  - $P(x_t|z_t^i = 1) \sim Mult(b_{i,1}, \dots, b_{i,m}) \sim f(x_t|\theta_i)$
  - Or, $P\left(x_t^j = 1 \middle| z_t^i = 1\right) = b_{i,j}$

a. Evaluation Q.
관측된 데이터가 얼마나
likely한지

b. Decoding Q.
관측된 데이터의 흐름을
가장 likely한 latent factor
의 sequence를알아내 보라.
find argmax_{z}P(Z|X,M,a,b,파이)

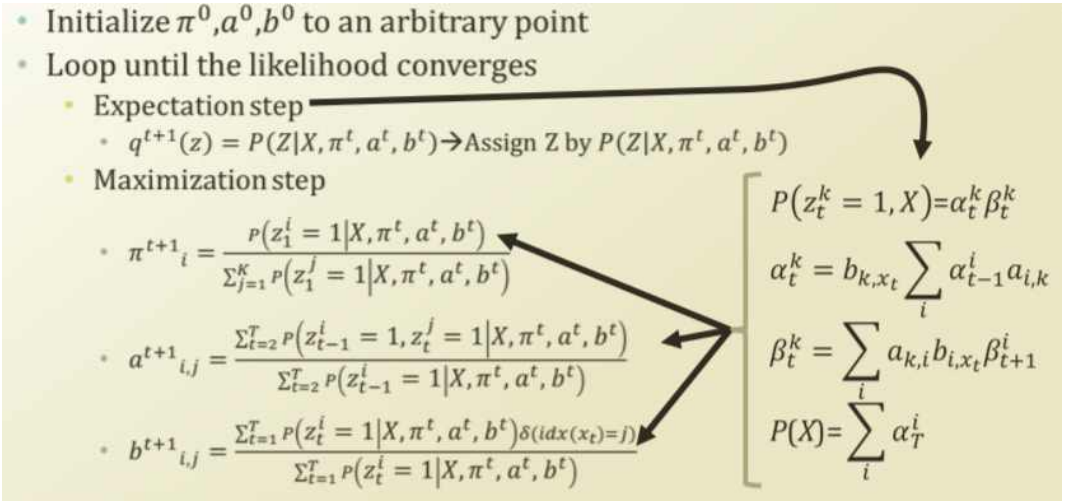
- Evaluation question
  - Given $\boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b}$, X
  - Find P(X|M, $\boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b}$)
  - How much is X likely to be observed in the trained model?
- Decoding question
  - Given $\boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b}$, X
  - Find $argmax_Z P(Z|X, M, \boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b})$
  - What would be the most probable sequences of latent states?
- Learning question
  - Given X
  - Find $argmax_{\pi,\, a,\, b} P(X|M, \boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b})$
  - What would be the underlying parameters of the HMM given the observations?

c. Learning Q.
이게 어려움. 파에비가 없어서
Given X only에서
주어진 데이터가 보일 확률을 가장 maximize하는 파에비를 찾아보아라. parameter를 inference해봐라
find argmax_{a,b,파이}P(X|M,a,b,파이)
(GMM도 X만 있듯이 이래서 EM을 미리 소개했던 것이다.)

a. Evaluation은 쉽다. counting을 통해 Joint prob으로 쉽게 풀 수 있다.
b. X의 관측에서 z는 무엇이었을까, loaded를 썼는지 fair를 는지.
(supervised와 비슷하게 training 데이터를 통해 파라미터를 학습하고 sequence의 latent factor를 예측.)
이는 DP를 활용해 Forward Backward Prob,를 만들고 DP와 trace를 bottum up 활용한 Viterbi Decoding으로 푼다. underflow prob은 로그 도메인으로 하여 해결한다. 이를 통해 Pos Tagger를 만들 수 있다.

c. EM과 유사한 바움웰치알고리즘으로 푼다.
파이 a, b를 임의로 고르고 Z를 Assign,

MLE(미분으로)로 다시 파라미터 optimize



Sampling based Inference: 문서에서 soft 클러스터링과 같은것

Forward, Rejection, Importance Sampling
Markov Chain
Stationary Dist.
Markov Chain Monte Carlo (MCMC)
Metropolis-Hastings Algorithm (Random walk M-H)
Gibbs Sampling (Special Case of MH Algorithm): PGM모델의 파라미터 inference를 하는 한축으로 이용
- LDA (Case Study)
https://github.com/aailabkaist/Introduction-to-Artificial-Intelligence-Machine-Learning
week11

###############################3###############################3###############################3

장바구니 분석
1. 지지도(Support): 지지도 = P(A∩B) : A와 B가 동시에 포함된 거래 수 / 전체 거래 수
그냥 MLE한 확률

2. 신뢰도(Confidence): 상품 A를 포함하는 거래 중 A와 B가 동시에 거래되는 비중으로,
상품 A를 구매 했을 때 상품 B를 구매할 확률
신뢰도 = P(A∩B) / P(A) = P(B|A) : A와 B가 동시에 포함된 거래 수 / A가 포함된 거래 수
그냥 조건부확률

3. 향상도(Lift): A를 살 때 B를 살 확률 / B를 사는 확율
P(A∩B) / P(A)*P(B) = P (B|A) / P (B) $\frac{P(B|A)}{P(B)}$
P(A|B)에 사전확률이 없는 것?

◎ bias and variance trade-off:

Error의 원천: Approximation & Generalization

a. Variance: (우리가 가진 데이터셋과 모든 데이터 셋을 봤을 때의 차이에 생긴다)

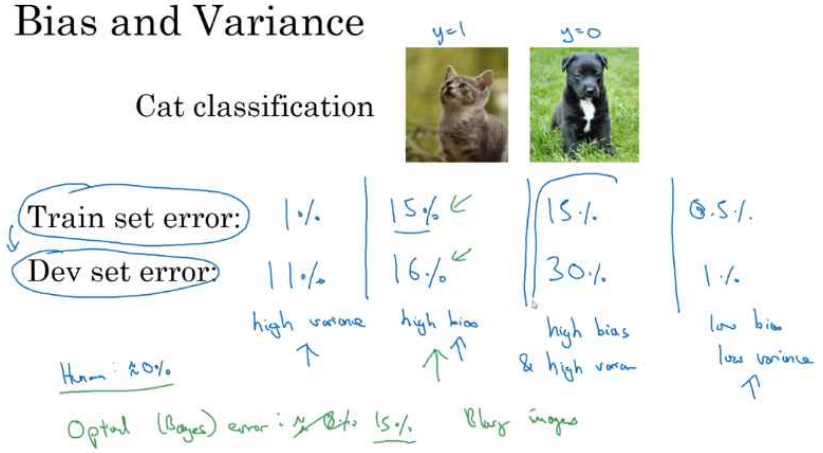
해결방안: generalization과 관련. 이는 regularization(data aug, batch norm 등)으로 해결한다.

b. Bias: (데이터 셋에는 문제가 없다.) 우리가 가진 모델(Approximation)의 한계점.

해결방안: 다른 hypothesis(hyper-parameter 변경, 모델 변경(comlex) 등)를 사용.

- 앤드류응님

이를 training set error와 dev set error를 보고 판단한다.



두 번째 예시는 사람이 (Optimal Error)가 거의 0일 일 때 high bias

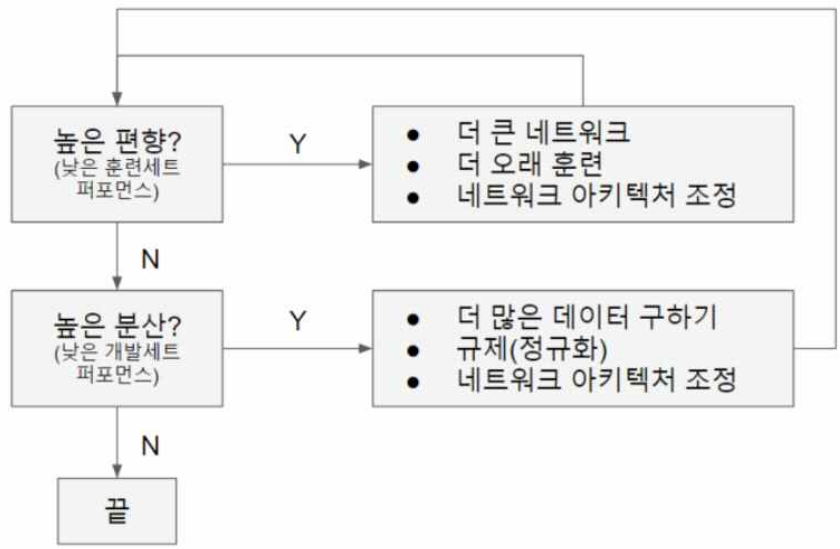
가정: 인간 수준의 성능이 기본으로 되어야 합니다. 이번 예제에서는 개와 고양이를 분류할 때 인간 수준의 성능은 0% 에 가까울 것입니다. 조금 더 일반적으로 이야기 하면, 베이지안 최적 오차가 0% 라는 가정이 깔려 있습니다.

베이지안 최적 오차 개념은 차후에도 배웁니다. 너무 걱정안하셔도 됩니다.

초기 머신러닝 시대에는 Bias and variance trade-off 관계가 있었다.

현재 딥러닝 시대에는 정규화를 잘했다면, 더 큰 네트워크와 데이터는 trade off없이 둘 다 감소 시킨다.

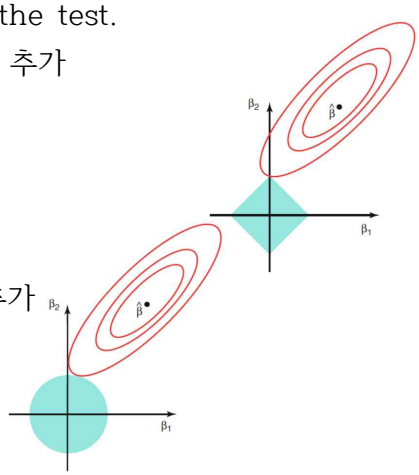
이게 딥러닝의 유용한 점이다.



Occam's Razor: hypothesis 중에서 에러가 유사하면 심플한 모델이 좋다.
Cross-Val: sampling을 따라 해보는 것.
average hypothesis를 구해 bias and variance trade-off를 조절하려고!

◎ L1, L2 Regualarization:
Regualarization은 기본적으로 perfect fit을 포기하는 것.
complex한 모델을 쓰되 모델이 있는 데이터 셋에 둔감하게 만들어 주는 것.
by reducing training acc increase potential fit in the test.

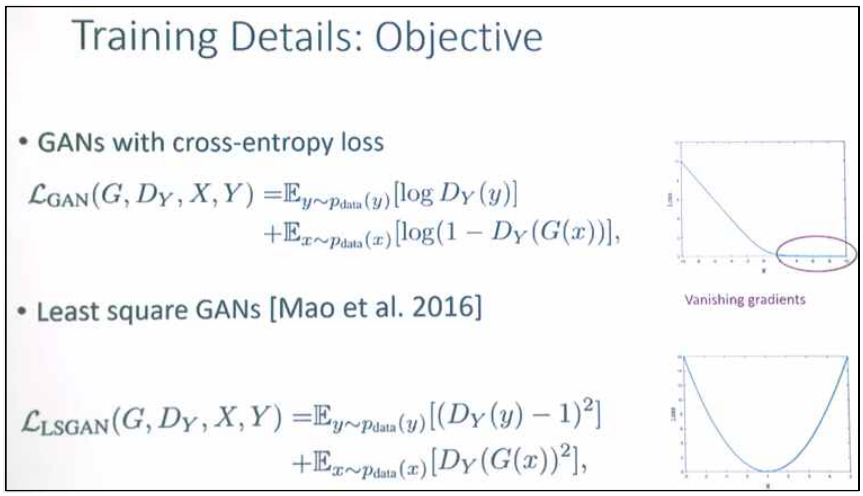
- a. L1 Regualarization == 라쏘(Lasso) // 절대값항 추가
극단적으로 만난다. 파라미터가 0이 될 수 도 있다.
참고로 싸클간은 이거씀. Guiding Force
원래 간은 log를 씌운거



- b. L2 Regualarization == 릿지(Ridge) // 제곱항 추가
미분이 쉬워서 릿지를 가장 많이 사용.

cf) 중요 내 플젝 일부이니까
싸이클간은 vanishing gradient 문제를 해결하기 위해 기존 간의 log cross entropy 대신에
LSGAN의 Loss를 사용해 그냥 1일 0으로 해서 제곱으로 하는 (아래) 모드 콜랩스같은 것도 없어졌다.
cycle loss와 identity loss에 L1 loss를 추가하면 guiding force를 한다!

L1, L2는
이런건 모델이나 데이터의
특성에 따라 다르다.



L2에서는 이렇게되며 closed sol 나옴.

$$\frac{d}{dw}E(w) = \frac{d}{dw}\left(\frac{1}{2}\|train - Xw\|^2 + \frac{\lambda}{2}\|w\|^2\right)$$

$$= -X^T \cdot train + X^T Xw + \lambda w = 0$$

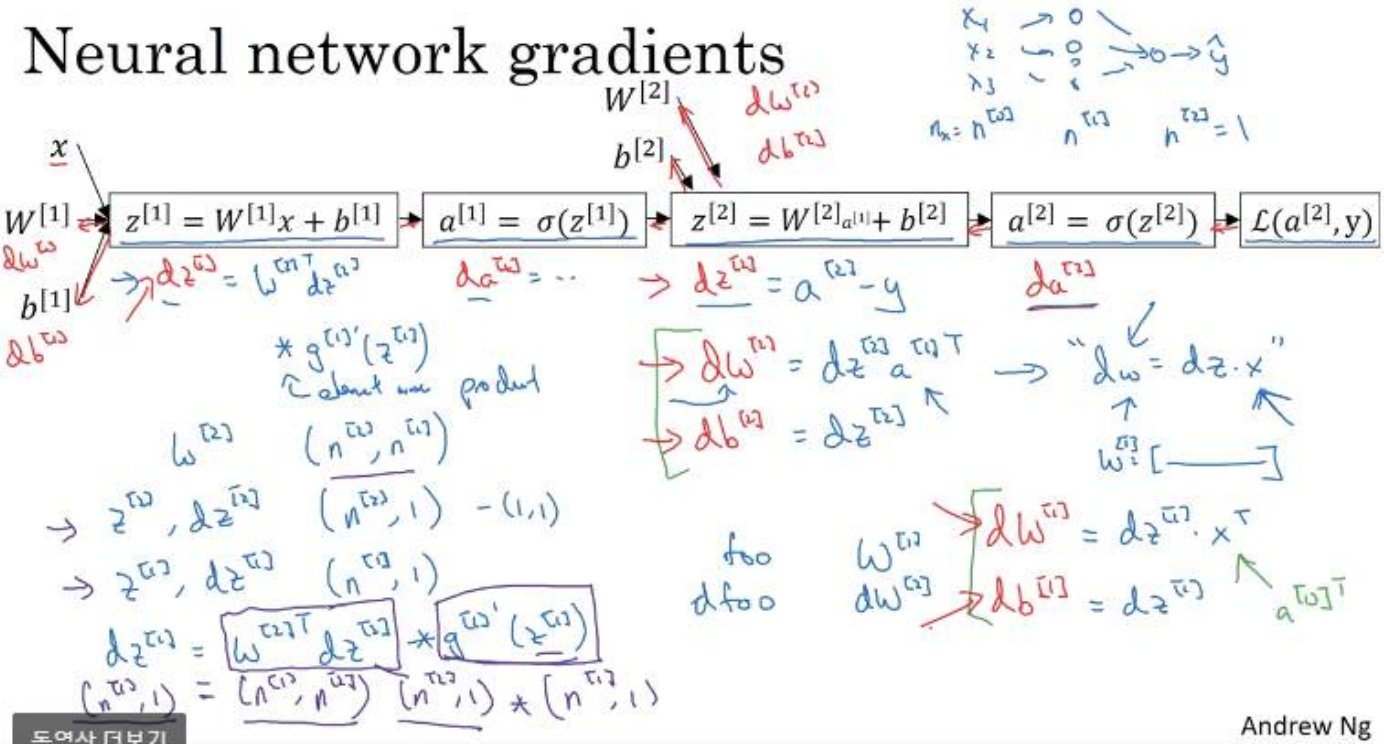
train은 y_train w에 이렇게 추가됨 $w = (X^T X + \lambda I)^{-1} X^T \cdot train$

◎ 회귀 / 분류시 의 metric:

		Actual Value	
		True	False
Estimated Value	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

RECALL: VIP를 분류할 때, TP / TP + FN
True == Vip False == Non VIP
PRECISION: SPAM을 분류할 때 , TP / TP + FP
True == Spam False == ham
F-Measure: F2일 때 recall이 중요, F0 precison에 초점

◎ Back-propagation:
뉴럴 넷에서 백프로파할 때 차원이 일치하는지 확인해라



Summary of gradient descent

$dz^{[2]} = a^{[2]} - y$

$dW^{[2]} = dz^{[2]}a^{[1]T}$

$db^{[2]} = dz^{[2]}$

$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$

$dW^{[1]} = dz^{[1]}x^T$

$db^{[1]} = dz^{[1]}$

$dZ^{[2]} = A^{[2]} - Y$

$dW^{[2]} = \frac{1}{m}dZ^{[2]}A^{[1]T}$

$db^{[2]} = \frac{1}{m}np.sum(dZ^{[2]}, axis = 1, keepdims = True)$

$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$ (element-wise product)

$dW^{[1]} = \frac{1}{m}dZ^{[1]}X^T$

$db^{[1]} = \frac{1}{m}np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

◎ 최적화 방법들:

학습알고리즘을 훈련할 때 평균값의 중심을 0으로 할 때가 있다.
-> 다음 층에서 학습을 더 잘할 수 있다.

시그모이드와 tanh의 단점: z가 너무 크거나 작으면 함수의 도함수가 너무 작아지는 것.
경사 하강이 느려짐. 시그는 이진분류아니면 안씀.

Tanh 장점:
Tanh 의 값이 [-1, 1] 사이에 있고 평균이 0이기 때문에,
데이터를 원점으로 이동하는 효과가 있습니다.
이는 평균이 0.5인 Sigmoid 보다 더 효율 적입니다.
자세한 내용은 다음에 이야기 할 것입니다.

ReLU 의 장점:
0보다 큰 활성화 함수의 미분값이 다른 함수에 비해 많아서 빠르게 학습 할 수 있습니다.
왜 비선형 활성화 함수가 필요한지는 다음 시간에 이야기 할 것입니다.
가장 기본으로 사용한다.

Why Non-linear Activation Functions (C1W3L07)

딥러닝으로 회귀를 풀 때는 마지막 층에 비선형함수를 안 쓴다. 그래도 그전에는 다써야한다. 막층에 렐루를 사용할 수 있다 회귀에서 음수가 없을 때는.

예를 들어 g(z)=z 라는 선형 활성화 함수를 사용한다고 가정했을 때, 3개의 은닉층을 쌓아도 g(g(g(z)))=z 로 아무런 혜택을 얻지 못했습니다. 따라서 은닉층에는 비선형 활성화 함수를 사용해야 합니다.

◎ Batch Normalization:

학습 속도가 개선된다 (학습률을 높게 설정할 수 있기 때문)
가중치 초깃값 선택의 의존성이 적어진다 (학습을 할 때마다 출력값을 정규화하기 때문)
과적합(overfitting) 위험을 줄일 수 있다 (드롭아웃 같은 기법 대체 가능)
Gradient Vanishing 문제 해결

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Current explanation of BatchNorm's success
현재까지의 BatchNorm의 성공에 대한 설명으로는 internal convariate shift(ICS)
internal convariate shift 이전 layer로 인한 다음 layer의 input distribution의 변화임
internal convariate shift가 training시에 부정적 영향을 줄 것이라고 짐작되어 왔음
BatchNorm의 목적은 internal convariate shift를 없애는 것이였음

- Smoothing Effect of Batchnorm
Batch Norm의 핵심 효과는 training process 있음
Batch Norm은 optmization problem의 land scape를 significantly smooth 하게 만들어 준다.
Lipschitzness of the loss function를 향상 시킴
Lipschitzness improvement로인한 효과
a. loss changes at a smaller rate
b. magnitude of gradient are smaller
좀 더 함수가 립시츠 해짐??(Lipschitzness가 커진다는 말인듯)
즉 loss의 효과가 더 effective 해진다고 해석 가능
립시츠 연속 함수는 두 점 사이의 거리를 일정 비 이상으로 증가시키지 않는 함수

진짜 핵심은 바로 gradient를 좀더 reliable하고 predictive 하게 만든다는 것
Lipschitzness가 크면 계산된 gradient의 방향으로 larger step을 밟더라도 step 후의 gradient 방향과도 꽤 정확하게 맞는다. (step 전 후의 gradient가 비슷하다는 뜻인 듯)

- 원래 설명 CS231N
BN을 쓰면 Layer Output이 더이상 deterministic하지 않고 jittering 하면서 stochastical한 효과를 준다고 Serena Yeung 누나 설명

a. Train time
BN : 하나의 데이터가 여러 무리(minibatch)와 무작위로 묶임. 어떤 친구들과 휩쓸리냐에 따라 매번 정규화 양상이 달라짐(Stochastic Jittering)

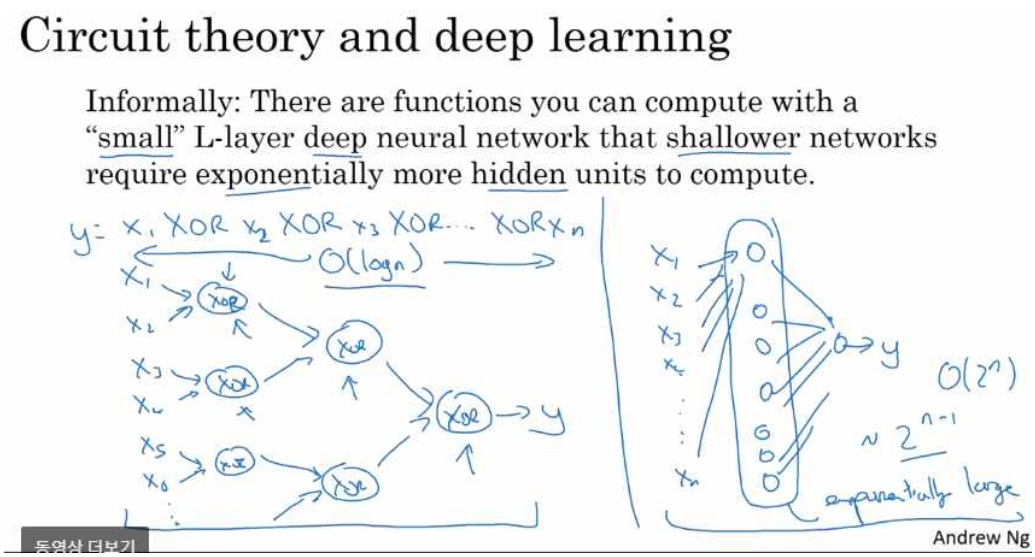
b. Test time
BN의 경우 Global normalization

◎ 딥러닝이 잘되는 이유

직관 1: 네트워크가 더 깊어 질 수록, 더 많은 특징을 잡아낼 수가 있습니다. 낮은 층에서는 간단한 특징을 찾아내고, 깊은 층에서는 탐지된 간단한 것들을 함께 모아 복잡한 특징을 찾아낼 수 있습니다.

직관 2: 순환 이론에서 따르면, 상대적으로 은닉층의 개수가 작지만 깊은 심층 신경망에서 계산할 수 있는 함수가 있습니다. 그러나 얕은 네트워크로 같은 함수를 계산하려고 하면, 즉 충분한 은닉층이 없다면 기하급수적으로 많은 은닉 유닛이 계산에 필요하게 될 것입니다.

순환 이론: 로직 게이트의 서로 다른 게이트에서 어떤 종류의 함수를 계산할 수 있을지에 관한 것입니다.



◎ Probability Theory:

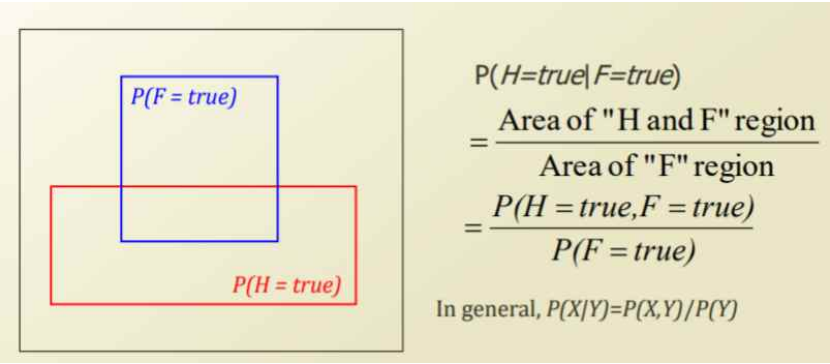
확률이란:

Frequentistic View: It is the relative frequency with which an outcome would be obtained if the process were repeated a large number of times under similar conditions.

Bayesian View: Probability is your degree of belief in an outcome.

확률 자체로 사용보다 조건부 확률, 무엇이 Given 일 때의 확률을 더 많이 사용한다.

결합확률 또한 중요한데 이는 많은 정보를 주기 때문



* Law of Total Prob.

- Law of Total Probability
  - a.k.a "summing out" or marginalization
  - $P(a) = \sum_b P(a, b) = \sum_b P(a \mid b)\, P(b)$
  - When B is any random variable

Joint를 알면 Marginalization을 통해 개별에 대해서 알 수 있다.

- Also, consider this case
  - given a joint distribution (e.g., P(a,b,c,d))
  - We can obtain any conditional probability of interest
  - $P(c \mid b) = \sum_a \sum_d P(a, c, d \mid b) = 1/P(b) \sum_a \sum_d P(a, c, d, b)$
  - Where 1 / P(b) is just a normalization constant

즉 joint를 알면 개별 individual prob이나 conditional prob도 알 수 있지만

근데 지수적으로 알아야하는 값들이 늘어남 나이브 베이즈 때와 비슷한 얘기 했다.

* Chain Rule & Factorization

- We can always write
  - $P(a, b, c, \ldots z) = P(a \mid b, c, \ldots. z)\, P(b, c, \ldots z)$
  - by definition of joint probability
- Repeatedly applying this idea, we can write
  - $P(a, b, c, \ldots z) = P(a \mid b, c, \ldots. z)\, P(b \mid c, .. z)\, P(c \mid .. z) .. P(z)$

우변 뒤쪽으로 나누면 정의

이렇게 펙토라이즈 가능.

- Variables *A* and *B* are independent if any of the following hold:
  - P(A | B) = P(A)
    - P(A,B) = P(A)P(B)
    - P(B|A) = P(B)

원래 동전도 앞 뒤 상황 상관 없이 독립이라 쉽게 확률을 계산 할 수 있었다.

이것이 Marginal independence의 예시 이며, 아래와 같으면 아니다. 조건부 독립은 나이브 베이즈!

- P(OfficerA=Go|OfficerB=Go) > P(OfficerA=Go)
- **This is not marginally independent!**

- Conditional independence
  - P(OfficerA=Go|OfficerB=Go,Commander=Go) =P(OfficerA=Go|Commander=Go)
  - **This is conditionally independent!**

◎ Information Theory:

◎ Decision Theory:

1. Sentence Representation

- A sentence is a variable-length sequence of tokens: X=(x1,...,xT)X\
- Each token could be any one from a vocabulary: xt∈Vxt∈V
    - ex) space-separated, 형태소(Morphs) 단위, 모든 syllable(어절) 단위,
      predefined된 딕셔너리의 term 등등

- 컴퓨터에게 단어를 숫자로 표현하기 위해서, 단어장(Vocabulary)을 만들고, 중복되지 않는 인덱스(index)로 바꾸고 문장을 일련의 정수로 표현.(Encoding)
    - Ex) One hot Encoding(One of K Encoding)
    - Pros&Cons: 모든 토큰 간의 거리가 같다.(arbitrary함을 나타낼 수 있다.)
    - BUT 의마가 없다. 개와 늑대의 거리가 가까우면 좋겠다.(의미를 Capture할 수 있도록)

→ 각 토큰을 Continuous vector space에 투영하자! 임베딩(Embedding)! → Table Look-up을 통해

이러면 sentence는 sequence of vector가 된다.
BUT sentnece의 size는 고정되어 있지 않으니 어떻게 fixed size representation을 찾을 수 있는지.
토큰에 대한 의미를 갖고있는 vector를 찾는 것이 table lookup이었다면, 문장에 대한 의미를 갖고있는 vector를 찾는 것을 이 장에서 다루어 보자.

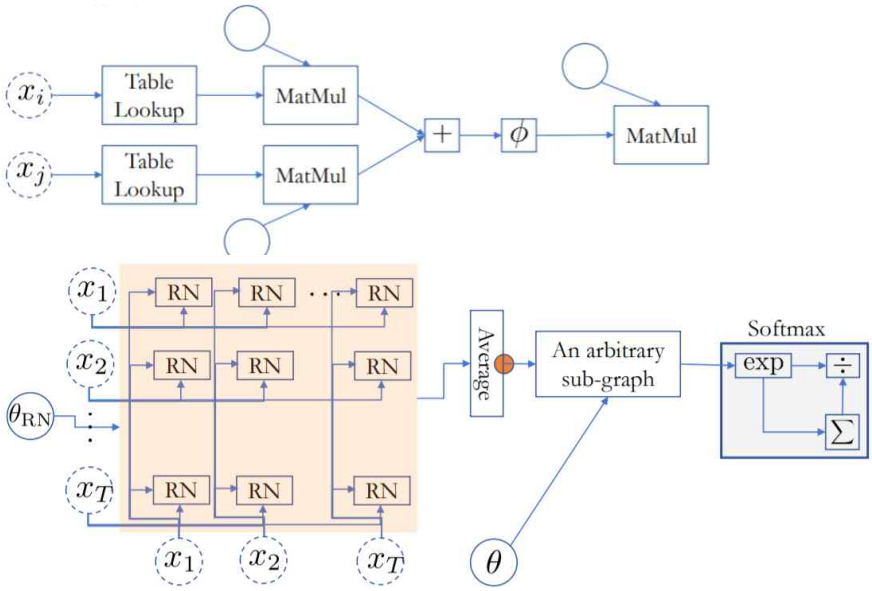
1) CBoW(Continuous Bag-of-words)
단지, 어떤 단어가 들어있는지만 고려하며 각 단어, 토큰들은 이미 table lookup을 통해 vector로 바뀐 상태에서의 average를 보자.
- 각 토큰들이 hi-dimensitional(300d, 3000d, ...의)하게 고차원에서 점들로 있으며, 문장에 있는 점들의 average point가 문장을 표현(의미)하는 것.
- Continuous Bag-of-N-gram으로 generalization도 가능.
- order를 무시했음에도 성능이 좋다.
- averaging된 결과값은 representation space안에서 비슷한 의미를 가진 문장들은 가까운 위치에 , 다른 의미를 가진 문장들은 멀리 있는 형태가 될 것. (Text Classification 문제를 풀기에 적합한 상태)

정형화 된 Represenation 방법은 없다.

- 결국 sentence represenation은 NN이나 머신러닝 관점에서보면, Universial한 representation(모든 문제를 포용하는 표현 방법)이 있다기 보다는 내가 가진 문제를 푸는 것에 가장 적합한 represenation이 무엇인지를 의미한다!
- sentiment analysis를 하는 툴을 만들었다고 하면, representation은 sentiment analysis을 하는데 적합한 representation이 나오고 만약 category classification을 하는 툴을 만들었다고 하면, representation이 category classification에 적합한 representation이 나온다.
- 이 두 representation은 compatible할 수도 있고, 안 할 수도 있다. 다시 말해, training 하는 모델을 어떻게 만드냐에 따라서 representation이 따라간다.
 이게 좀 이해 안되네. 첨에 임베딩한거에서 변화한다는 것인가.

2) Relation Network : Skip Bigrams

- One subgraph in the DAG / 파이는 비선형함수 relu, tanh

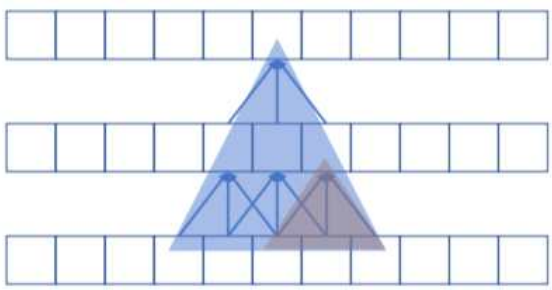


- 각 pair에 대해 Relation Neural Network를 만들어서 각 pair의 continuous vector representation을 섞어서 하나의 벡터로 나타낸다. (pair의 representation을 찾는 것이다.)
- 그리고 문장에서 RN을 통해 나온 벡터에 Average를 취한다.
- 영어와 같은 구조 SVO의 문법 구조에서 좋을지 의문이 들지만, 한국어 독어와 같은 SOV 구조에선 ㄱㅊ

3) Convolutional Networks

- Captures k-grams hierarchically // One 1-D convolutional layer: considers all k-grams
- 1d conv layer에서는 k-gram을 hierarchical하게 gradually growing 하는 structure를 보는 것으로 생각할 수 있다. 직관에 잘 맞아 보인다.
- Fits our intuition of how sentence is understood:

tokens → multi-word expressions → phrases → sentence



4) Self Attention

- CNN을 Weighted RN라고 생각할 수 있다.

RN: $h_t = f(x_t, x_{t-k}) + \cdots + f(x_t, x_t) + \cdots + f(x_t, x_{t+k})$

CNN:

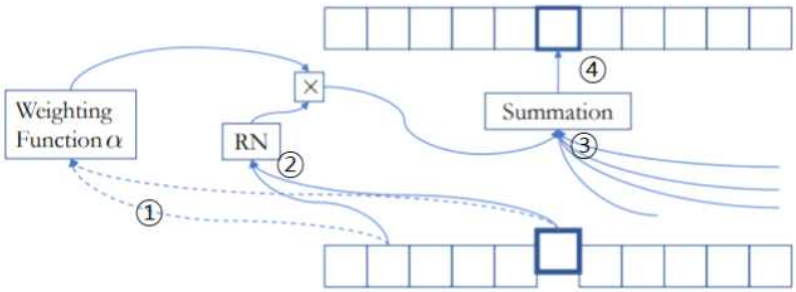
$$h_t = \sum_{t'=1}^{T} \mathbb{I}(|t' - t| \leq k) f(x_t, x_{t'}) \quad \text{where } \mathbb{I}(S) = 1, \text{ if } S \text{ is true, and } 0, \text{otherwise.}$$

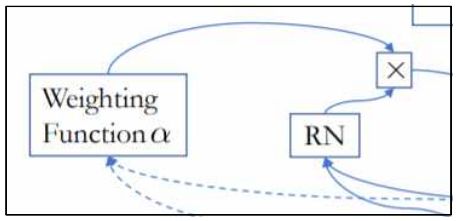
- weighting된 0,1과 같은 상수가 아니라 NN이 알아서 계산하도록하여
  토큰내의 k근처만 고려하는 것이 아니라 중요한 단어를 알아서 고르도록!!

$$h_t = \sum_{t'=1}^{T} \alpha(x_t, x_{t'}) f(x_t, x_{t'})$$

- The weighting function could be yet another neural network
  - Just another subgraph in a DAG: easy to use! (NN subgraph (RN) 하나 더 생긴것으로 볼 수 있다.)
    $\alpha(x_t, x_{t'}) = \sigma(\text{RN}(x_t, x_{t'})) \in [0, 1]$
  - Perhaps we want to normalize them so that the weights sum to one (더해서 1이 되도록 할 수 있다.)
    $$\alpha(x_t, x_{t'}) = \frac{\exp(\beta(x_t, x_{t'}))}{\sum_{t''=1}^{T} \exp(\beta(x_t, x_{t''}))}, \text{where } \beta(x_t, x_{t'}) = \text{RN}(x_t, x_{t'}))$$

- 장점: Able to capture long-range dependencies within a single layer. (CNN에 비해)
  Able to ignore irrelevant long-range dependencies. (RN에 비해)





- Graphical 과정
  모든 Possible pair에 대해 weight을 계산 (Weighting Function α)
  그리고 그 둘의 Interection을 계산 (RN을 쓰겠죠.)
  그 두개 를 곱해서 (RN은 interaction, Weight은 그것의 가중치)
  다 더하여 나온다.

- Further generalization via multi-head and multi-hop attention.
  multi-head attention: 우측 상단의 Head라는 부분을 늘려 multi-head로 만들어, Long 과 short을 따로 수행하도록 할 수 있다.
  multi-hop attention: 계산된 결과를 다시 input으로 넣어(위 그림의 위 벡터) weighting을 바꿔주는 등의 generalization 방법이 있다.

(이런 것이 end2end memory network, Attention is all you need 에서 쭉 나왔으며 multi-head and multi-hop attention은 약간 art로 가는 느낌이다(?!).)

단점: Quadratic computational complexity(forward path만 보면) O(T2)O(T2)
Some operations cannot be done easily:
e.g., counting, (counting같은게 쉽지 않다. 문장, 단어간의 거리와 같은 것, 필할 때 있음)

5) RNN

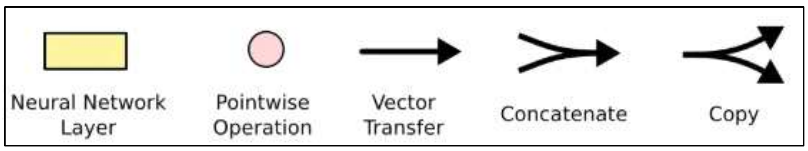
- Online compression of a sequence O(T)O(
self attention은 compress를 안하고 매번 interaction을 보는 것에 반해 sentence를 linear한 타임에 online으로 compress 하는 방법

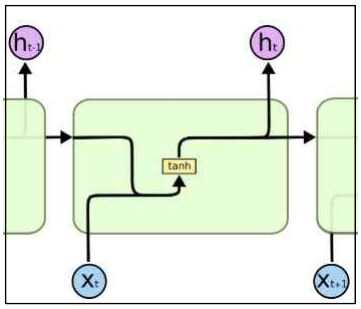
$$h_t = \mathrm{RNN}(h_{t-1}, x_t), \text{where } h_0 = 0.$$

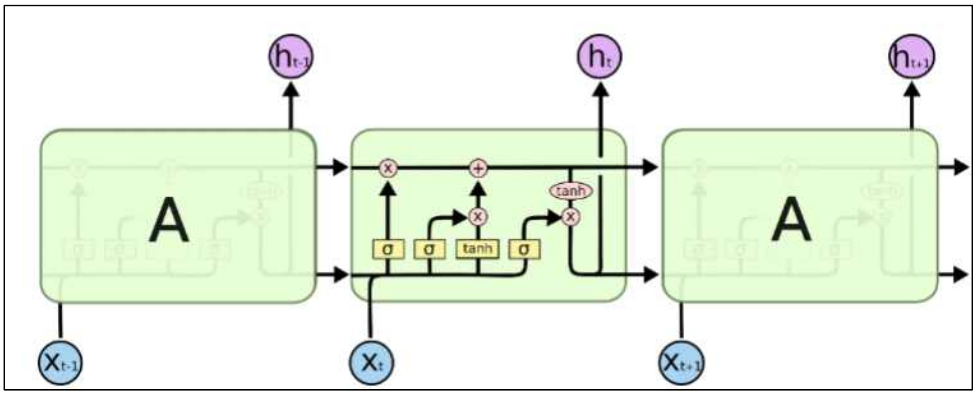
- 과정:
매번 한 토큰을 읽을 때마다 h_t 라는 메모리를 업데이트를 하고 그 토큰을 버리고 다음 토큰으로 가고 또 업데이트 하고 그러면 메모리에 지금까지 읽은 전체 토큰의 정보가 들어간다.
문장 하나를 읽을 때 마지막에 나온 h_T 전체 문장을 summarize해주는 애가 된다.
compress를 하다보니 벡터사이즈는 한정되어 있고 length가 길수록 compress가 어렵고 information loss가 생긴다. 이를 overcome 하기 위해 bidirectional RNN을 쓰기도한다. (양방향으로 읽어 합쳐 t번째 token의 representation을 만든다.)

- 단점
구조가 Sequential하게 하나씩 읽는 구조다 보니 Modern hardware랑 어긋난다.
(동시에 계산이 안되는 parallelize하는게 trivial한 점으로 인한 속도 저하)
그 외의 training할 때 여러 문제들이 있는데 LSTM과 GRU와 같은 걸 사용하면 대부분 없어진다.
major framework에서 노드로 많이 들어가 있으니 LSTM ,GRU 갔다 쓰자.

- LSTM
기존 RNN의 한계: Unfortunately, as that gap grows, RNNs become unable to learn to connect the information. 그리고 이론적으로는 원래 되는데, 실전에서 안되는 근본적인 이유는 논문에서 찾았었다. 그리고 LSTM 등장.
- 개요: 원래 Long term을 위해 개발되었고, 오랬동안 정보를 기억하는 것을 애를 쓰는 것이 아니라 LSTM의 기본 행동이다. 기본 RNN은 오른쪽과 같고 LSTM은 아래와 같이 4개의 모듈이 있는 형태로 있다.
아래 그림에서 각 라인은 entire vector를 운반하고 하고 노란 박스가 학습하는 것이다.







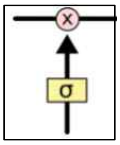
- 코어아이디어

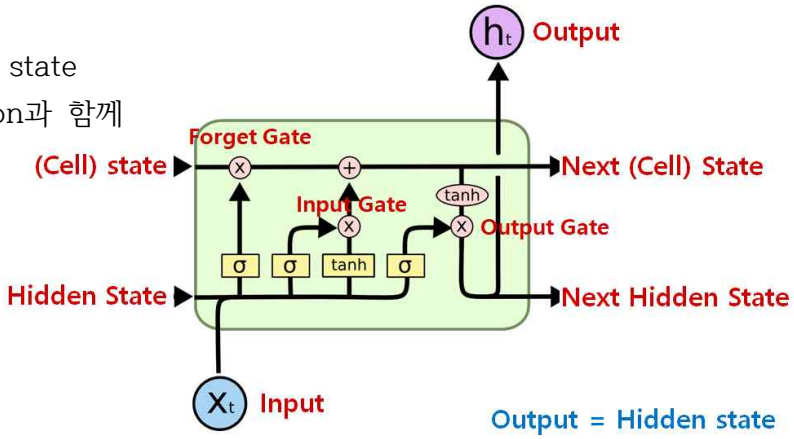
!! Cell State: The key to LSTMs is the cell state

컨베이어 벨트와 비슷하게 가고 선형 interaction과 함께 흘러감. 그리고 정보는 게이트라는 구조에 의해 제거하고 추가한다.

이전 히든스테이트에서 시그모이드를 통과해 엘먼와이즈 곱으로 제거!

이러한 구조는 3개로 셀스테이트를 보호하고 컨트롤한다.





- 과정

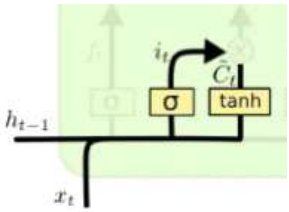
a. Forget gate: 셀스테이트의 어떤 정보를 버릴지 정한다.

b. Input gate: 셀스테이트에 어떤 새로운 정보를 저장할 것인지.

씨 틸다는 얼마나 셀스테이트의 반영될 후보가 되어 얼마나 반영할지를 정한다.

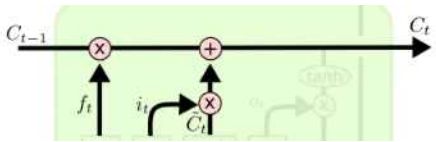
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

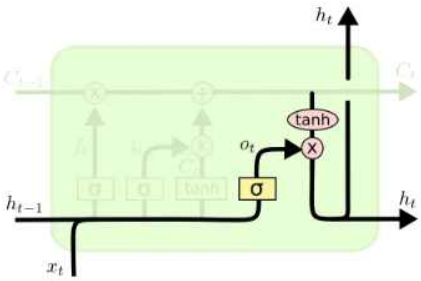
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



c. Update: 정했던 것들을 셀스테이트에 반영한다.

d. output: 그리고 이렇게 얻어진 셀스테이트를 어떻게 output으로 낼 것인지 정한다.





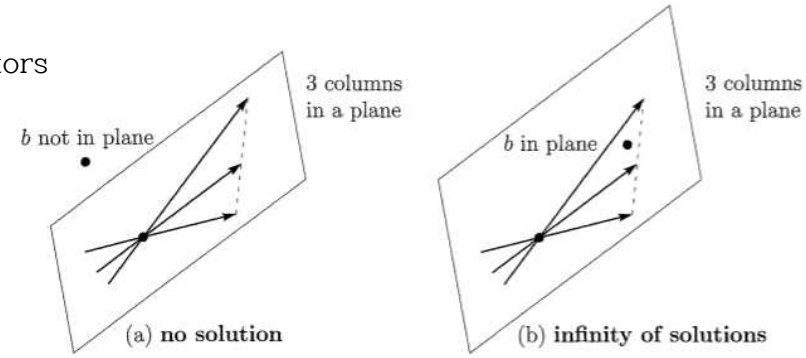
- Geometry of Linear Equation

1) row from: Intersection of lines, planes, hyperplanes

Singular Cases: no solution: parallel / infinite solutions: line, plane of intersection

column from: Combination of column vectors

Singular Cases: 열공간에 없을 때



Linear Independence

식 $c_1 v_1 + c_2 v_2 + \cdots + c_n v_n = 0$ 에서 모든 $c_i = 0$ 일 때만 성립하면 벡터들 끼리 서로 Linearly independent하다고 한다. 벡터들 끼리 dependent 하다는 것은 One vector is a combination of the others 라는 것이다.

Projections and Least Squares

Square system → G.E. → Upper Triangular. (Chapter1)

Underconstrained System: Rectangular system when # of Equation < # of Unknowns → G.E. → Reduced Row Echelon form (Chapter2)

Overconstrained System: Rectangular system when # of Equation > # of Unknowns, usually no solution.

Ax = b를 만족하는 해가 없으니 (inconsistent) $\|Ax - b\|$를 최소화하는 제일 적절한 Column space의 solution을 찾자. (Use Projection! $min\|Ax - b\|^2$ 하자 → Least Square Solution)

Gaussian Elimination

Forward Elimination → Triangular System (Upper Triangular Matrix) → Back-substitution

Do pivoting when Zero appears in a pivot position. When all pivots are non-zero after G.E then It has unique solution.

Elementary Matrix in Elimination Steps

- Elemantary Matrix

$$E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -l_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad E_{21}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Triangular Factors and Row Exchanges

For each Elementary Matrix $E_{32}E_{31}E_{21}A = U$

$A = E_{32}E_{31}E_{21} = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}U = LU$

→ Triangular Factorization, LU Decomposition, LU Factorization

L has history of G.E. det(A) = det(L) * det(U)

LU Factorization are uniquely determined by A. (LU 분할이 되면, 시스템은 일대일 대응이다!)

Linear Regression 선대느낌으로.

보통 제약조건이 더 많은 것이 우리가 주어진 데이터와 그 것을 위한 line fitting하는 것이 기본인 ML에서 Ax = b를 만족하는 x벡터들 즉 파라미터들은 없으니, || Ax - b ||를 최소화하는 제일 적절한 Column Space의 솔루션을 찾자.

컬럼으로 보는 것은 어떻게 보면 featuere들로 보는 것. 아래로

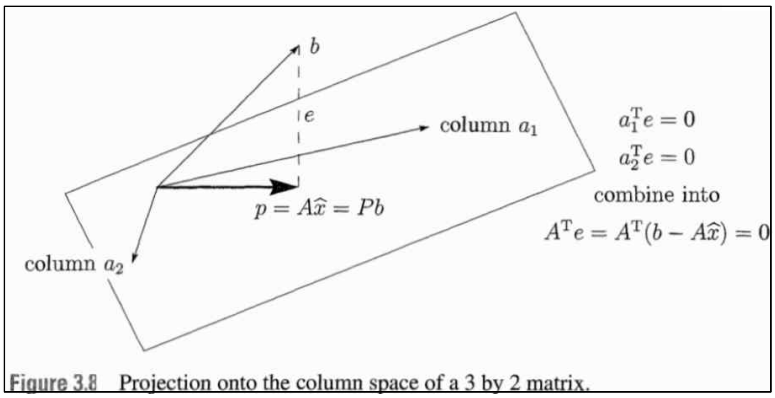


Figure 3.8 Projection onto the column space of a 3 by 2 matrix.

방법 1. 선대 맛 수직임을 이용

Finding $\hat{x}$ and the projection is so fundamental the we do it in 2 ways:

1) The error vector must be orthogonal to each column vector

$$\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \left[b - A\hat{x} \right] = 0$$

2) Since Column space is perpendicular to left null space, The error vector ($e = b - A\hat{x}$ must be in the nullspace of A^T (left null space).

$A^T(b - A\hat{x}) = 0$ or $A^T A\hat{x} = A^T b$

- Normal form ($A^T A$의 역행렬이 항상 존재하지 않으니) : $A^T A\hat{x} = A^T b$
- Best estimate : $\hat{x} = (A^T A)^{-1} A^T b$ and $(A^T A)^{-1}$ can be pseudo inverse
- Projectioin Matrix: $\mathbf{P} = A(A^T A)^{-1} A^T$, Point: $p = A\hat{x} = \mathbf{P}b = A(A^T A)^{-1} A^T b$

방법 2. 편미분

$y_1 = ax_1 + b,\ y_2 = ax_2 + b,\ \ldots y_n = ax_n + b$ 에서 error를 최소화 하는 직선 a,b

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

error의 형태는 Ax-b의 형태이므로

$$\begin{bmatrix} ax_1 + b - y_1 \\ \vdots \\ ax_n + b - y_n \end{bmatrix} \rightarrow \|Ax - b\|^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

또는 미분을 활용해 $\|Ax - b\|^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2 = f(a,b)$ 이니까.

각각의 편미분을 활용해, $\partial f/\partial a = 0, \partial f/\partial b = 0$ a,b를 구해도 결과는 위와 같다.

Chapter 5 Eigenvalues and Eigenvectors

벡터 공간을 간단하게 표현하는 데 초점.

$Ax = \lambda x$

Ax의 결과를 람다라는 스칼라(Eigenvalue)와 벡터(Eigenvector)로 표현하자! (A는 n x n)

$\rightarrow (A - \lambda I)x = 0$ for non-zero x, $det(A - \lambda I) = 0$ 을 만족해야 (singular 여야)

이 때 람다에 대해 n차 식이 나오고 이 람다의 근을 구하는 것이, Eigenvalue를 찾는 것이다. 그리고 Eigenvector는 Nullspace of $(A - \lambda I)$이다. (Redueced row echelon form!) 행렬의 크기에 상관없이 eigenvalue 하나에 대해, nullspace를 이루는 free variable은 1개씩 밖에 나오지 않아 다행이다. 또한 eigenvalue에 따라 나오는 eigenvector들 끼리 independent해서 벡터공간을 해석하게 할 수 있는 또 하나의 방법을 제시한다.

Singular Value Decomposition

Chapter 5에서 배운 matrix A에 대한 diagonalization은($A = S\Lambda S^{-1}$) nxn 행렬 그리고 S의 inverse 가 존재했어야만 했다.

m x n의 임의 행렬일 때도, eigenvalue 가 n개가 나오지 않아도 decomposition 할 수 있도록 하는 것이 SVD이다.

$\rightarrow A = U\Sigma V^T$

U (m x m)와 V (n x n)는 Unitary, Orthonormal matrix이고, Σ는 m x n Diagonal matrix이지만 모든 diagonal term이 non-zero는 아닌 행렬이다.

그리고 이 람다가 큰 아이젠벡터로 행렬을 표현하는 것이 PCA이다.

과정: 먼저 A^TA 로 대칭행렬을 만들어 아이겐 벨류를 만들고 여기에 루트를 취해 singular value로 시그마를 구한다.

이후에는 (A^T의 rowspace의 basis)와 (A의 Nullspace의 basis)를 통해 V를 만들고, (column space of A의 basis)와 Nullspace of A^T의 basis를 통해 U를 구한다!

SVD를 통해 압축하여 아이겐 벨류(싱귤러벨류가 큰 값으로) 표현할 수 있으며 이는 PCA와 유사하다.