Theory

Learning From Data

Map

Paradigms:

- supervised
- unsupervised
- reinforcement
- active
- online

Theory

- VC
- bias – variance
- complexity (computation, P, NP, ...)
- bayesian

Techniques

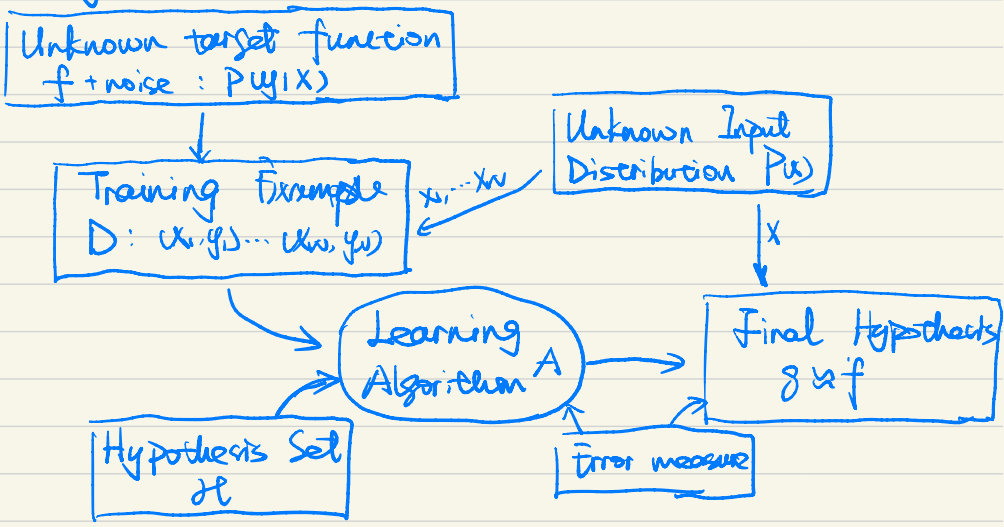
Model

- linear
- neural networks
- SVM
- nearest neighbors
- RBF
- Gaussian Process
- SVD
- graphical model

Method

- regularization
- validation
- aggregation
- input processing

Learning Problem



Learning model = hypothesis set + Learning Algorithm

Views
- Statistic: rigorous proof, idealized model in great detail
- ML: less assumption, weaker result, broader application
- Data Mining: More data analysis than prediction

Learning Feasibility

infer outside ↑ with D in a probalistic way

要了解
1. A pattern exists
2. Can't pin it down mathematically
3. We have data

Hoeffding's Inequality

Let $X_1, \cdots X_n$ be independent R.V bounded $[a_i, b_i]$ a.s.

$S_n = X_1 + \cdots + X_n$

$$\Rightarrow \begin{cases} P(S_n - E(S_n) \geq t) \leq \exp\left(-\frac{2t^2}{\sum (b_i - a_i)^2}\right) \\ P(|S_n - E(S_n)| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum (b_i - a_i)^2}\right) \end{cases}$$

In sample Error

$E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)]$

= fraction of D that f disagree with h

Out sample Error

$E_{out}(h) = P(h(x) \neq g(x))$, P is based on x

$\Downarrow \quad 0 \leq E \leq 1$

$P(|E_{in}(h) - E_{out}(h)| > \varepsilon) \leq 2e^{-2\varepsilon^2 N}$

$\Downarrow \quad \mathcal{H} = \{H_1, \cdots H_M\}$ → choose h based on E_{in}

$P(|E_{in}(h) - E_{out}(h)| > \varepsilon) \leq \sum_{h \in \mathcal{H}} P(|E_{in}(h_i) - E_{out}(h_i)| > \varepsilon)$

$(2Me^{-2\varepsilon^2 N})$ $\mathcal{H}$ complex 大 M 多大

Feasibility Question split into 2

1. Can we make sure $E_{out}(g)$ is close to $E_{in}(g)$
   Hoeffding's Inequality
2. Can we make $E_{in}(g)$ small enough
   Learning algorithm A

Error Measure

- In sample Performance over entire input space x
  $E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N} e(h(x_i), f(x_i))$
- Out of sample Performance in D
  $E_{out}(h) = E_x[e(h(x), f(x))]$

In a noisy set

$P(y|x)$ is what we are looking for

$P(x)$ is the relative importance of x in gauging performance

VC Dimension

Testing:

$$P(|E_{in} - E_{out}| > \varepsilon) \le 2e^{-2\varepsilon^2 N}$$

Training:

$$P(|E_{in} - E_{out}| > \varepsilon) \le 2Me^{-2\varepsilon^2 N}$$

Effective # of Hypothesis

1. Growth Function — group h based on its behavior on D
   - A hypothesis $h: \mathcal{X} \to \{-1, 1\}$ — all input space
   - A dichotomy $h: \{x_1, \dots x_N\} \to \{-1, 1\}$
     - Number of Hypothesis $|\mathcal{H}|$ can be infinite
     - Number of dichotomy $|\mathcal{H}(x_1, \dots x_N)|$ is finite

     $$|\mathcal{H}(x_1, \dots x_N)| \le 2^N$$

   growth function count the most dichotomy of $x \in \mathcal{X}$

   $$m_{\mathcal{H}}(N) = \max_{x_1 \dots x_N \in \mathcal{X}} |\mathcal{H}(x_1, \dots x_N)| \le 2^N$$

2. Bound of Growth Function (with break point)

   Break point: If no data set of size k can be shattered by $\mathcal{H}$, then k is a break point

   Bound: If $m_{\mathcal{H}}(N) < 2^N$ for some k

   then $m_{\mathcal{H}}(N) \le \sum_{i=0}^{k-1} \binom{N}{i}$ → polynomial in N of degree k-1

   VC dim (Vapnik-Chervonenkis) of a hypothesis set $\mathcal{H}$

   $d_{VC}(\mathcal{H})$ is the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$

   $$m_{\mathcal{H}}(N) \le \sum_{i=0}^{d_{VC}} \binom{N}{i}$$

   The most points that $\mathcal{H}$ can shatter

VC dimension

- $d_{vc} \geq N \iff$ there exists D of size N such that $\mathcal{H}$ shatters D
- $d_{vc} < N \iff$ No set of N points can be shattered by $\mathcal{H}$

ν degree of freedom : effective number of hypothesis

3 Replace M with $m_{\mathcal{H}}(N)$

$$P[|E_{in}(g) - E_{out}(g)| > \varepsilon] \leq 4 m_{\mathcal{H}}(N) e^{-\frac{1}{8}\varepsilon^2 N}$$

$$\Updownarrow$$

☆ **VC generalization bound : $E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4 m_{\mathcal{H}}(2N)}{\delta}}$ with $1-\delta$**

proof : estimate (Replace) $|E_{in}(g) - E_{out}(g)| > \varepsilon$

with an artificial event $|E_{in}(g) - E'_{in}(g)| > \varepsilon$

- A loose bound, but tends to be equally loose hence useful for comparing generalization performance of models
- $E_{out}(g) \leq E_{in}(g) + \Omega(N, \mathcal{H}, \delta)$

$$\Omega(N, \mathcal{H}, \delta) = \sqrt{\frac{8}{N} \ln \left(\frac{4 m_{\mathcal{H}}(2N)}{\delta}\right)}$$

$$\leq \sqrt{\frac{8}{N} \ln \left(\frac{4 (2N)^{d_{vc}} + 1}{\delta}\right)}$$

- A loose estimate of $E_{out}(g)$ as guideline
  Test sample → estimate $E_{out}(g)$
- Can extend to other Error Measure

Bias vs Variance

Approximation-Generalization tradeoff

Goal: small E_{out}, good approximation of f out of sample

complex $\mathcal{H}$ $\begin{cases} \text{better chance of approximating } f \\ \text{worse chance of generalizing out of sample} \\ \quad \text{(Have to navigate in } \mathcal{H} \text{ through } \mathcal{D} \text{ for } g\text{)} \end{cases}$

Bias vs Variance: decomposing E_{out} into:

1. How well $\mathcal{H}$ can approximate f (overall, not in current sample)
2. How well can zoom in on a good $g \in \mathcal{H}$

Apply to real value f and square error

$$E_{out}(g^{(\mathcal{D})}) = E_x[(g^{(\mathcal{D})}(x) - f(x))^2]$$

$\Downarrow$ remove dependance on $\mathcal{D}$ by expectation

$$E_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = E_{\mathcal{D}}[E_x[g^{(\mathcal{D})}(x) - f(x))^2]] = E_x[E_{\mathcal{D}}[g^{(\mathcal{D})}(x) - f(x))^2]]$$

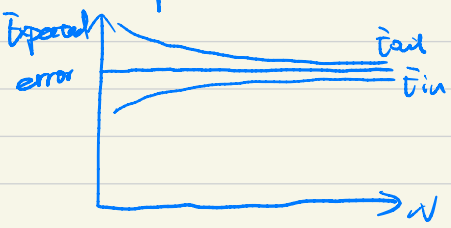
$\Downarrow\Downarrow$ average hypothesis $\bar{g}(x) = E_{\mathcal{D}}[g^{(\mathcal{D})}(x)] \approx \frac{1}{K}\sum_{i=1}^{K} g^{(\mathcal{D}_i)}(x)$ on many data set $\mathcal{D}_1 \cdots \mathcal{D}_K$ the best can get from the $\mathcal{H}$ (infinite data)

$$E_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2] = \underbrace{E[(g^{(\mathcal{D})}(x) - \bar{g}(x))^2]}_{\text{variance}} + \underbrace{(\bar{g}(x) - f(x))^2}_{\text{bias}}$$

$$E_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = E_x[\text{bias}(x) + \text{var}(x)] = \text{bias} + \text{var}$$

Learning Curve

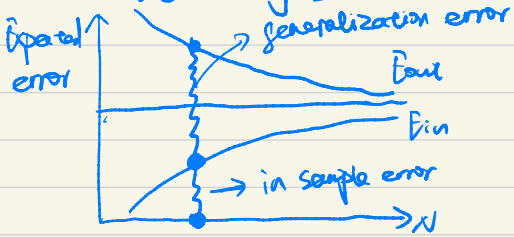
Simple Model



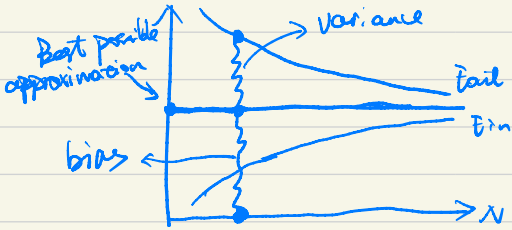
Complex Model

Match model complexity with data source not target complexity

VC analysis



bias-variance



Linear Case

$y = w^{*T} x + \text{noise}$

① Best approximation error σ^2 (= var(noise))

② Expected in sample error $\sigma^2 (1 - \frac{d+1}{N})$

③ Expected out of sample error $\sigma^2 (1 + \frac{d+1}{N})$

④ Expected generalization error $2\sigma^2 \frac{d+1}{N}$

Overfitting

Overfitting: $E_{in} \downarrow$ $E_{out} \uparrow$

fitting the data more than warranted (fitting the noise)

$$y = f(x) + \underbrace{\epsilon(x)}_{\sigma^2} \triangleq \sum_{q=0}^{Q_f} a_q x^q + \epsilon(x)$$

Overfitting:
- noise level: σ^2 + stochastic noise
- target complexity: Q_f + deterministic noise
- data set size: N -

Deterministic Noise: The part of f that $\mathcal{H}$ can't capture

Noise: something can't capture / deal with

Difference with stochastic noise.

1. depend on $\mathcal{H}$ (complex $\mathcal{H}$ → smaller)
2. fixed for a given X

Noise and bias-variance

$$E_D[(g^{(D)}(x) - f(x))^2] = E_D[(g^{(D)}(x) - \bar{g}(x))^2] + [\bar{g}(x) - f(x)]^2$$

$\Downarrow$ if $f(x)$ is noisy $y = f(x) + \epsilon(x)$, $E[\epsilon] = 0$

$$E_{D,\epsilon}[(g^{(D)}(x) - y)^2]$$

$$= E_{D,\epsilon}[(g^{(D)}(x) - \bar{g}(x))^2 + (\bar{g}(x) - f(x))^2 + (\epsilon(x))^2 + \underbrace{\text{cross term}}_{\downarrow 0 \text{ as } E[\epsilon(x)] = 0}]$$

$$\underbrace{E_{D,x}[(g^{(D)}(x) - \bar{g}(x))^2]}_{\text{var}} + \underbrace{E_x[(\bar{g}(x) - f(x))^2]}_{\text{bias}} + \underbrace{E_{\epsilon,x}[(\epsilon(x))^2]}_{\sigma^2}$$

var: impacted by both noise, capturing a model's susceptibility to being led astray by the noise

bias: deterministic noise

σ^2: stochastic noise

Not change with N

where overfit come in ← try to fit noise drag g away

Regularization

Constrain the learning algorithm to improve out-of-sample error
Necessary if model is too complicated for data amount/quality

View from VC

$E_{out}(h) \le E_{in}(h) + \Omega(\mathcal{H})$ for all $h \in \mathcal{H}$

Convert a measure $\Omega(h)$ for the complexity of an individual hypothesis
should fit better using simple h from $\mathcal{H}$

$\min_h E_{in}(h) \Rightarrow \min_h [E_{in}(h) + \Omega(h)]$

View from bias-var

sacrifice bias for smaller var

Unconstrained solution

$\min E_{in}(h) \longrightarrow w_{lin} = (X^T X)^{-1} X^T y$

Soft-order constraint / weight decay

① $\min E_{in}(h)$
$s.t \quad w^T w \le C \qquad C \uparrow \Leftrightarrow \lambda \downarrow$
$\Updownarrow$ KKT condition
② $\min E_{in}(h) + \frac{\lambda}{N} w^T w$ (augmented error) penalize large weight
$\longrightarrow w_{reg} = (X^T X + \lambda I)^{-1} X^T y$

Augmented Error

$E_{aug}(h) = E_{in}(h) + \frac{\lambda}{N} \tilde{\Omega}(h)$ regularizer
$\Uparrow$
$E_{out}(h) \le E_{in}(h) + \Omega(\mathcal{H})$

E_{aug} is a better proxy of E_{out} than E_{in}

Regularization is necessary since learning is sensitive to both stochastic and deterministic noise

Best way to constrain learning is in the direction of target

- stochastic noise : high frequency / non-smooth
- deterministic noise : non-smooth

⇓

Usually constrain learning toward smooth h hurt more to overfit noise than to hurt fit useful info

Regularizers:

1. weight decay
2. Weight elimination

$$\Omega(w) = \sum_{i,j,l} \frac{(w_{ij}^{(l)})^2}{\beta + (w_{ij}^{(l)})^2}$$

3. Early stop

choose hyper (λ, stop time) by validation

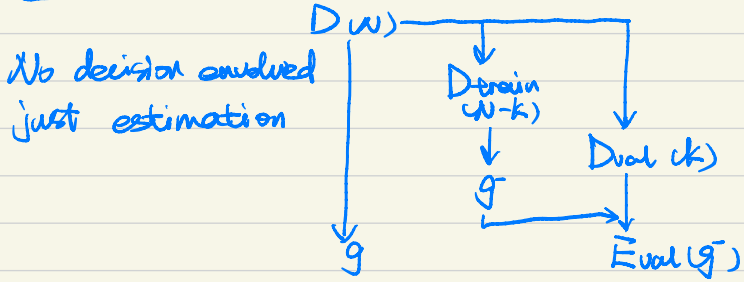
Validation

Both regularization and validation attempt to minimize E_{out}

$$E_{out} = E_{in} + \text{overfit penalty}$$

validation estimate this — regularization estimate this

Estimate E_{out} with Validation



No decision involved just estimation

$$E_{out}(g) \le E_{out}(g^-) \le E_{val}(g^-) + O\left(\frac{1}{k}\right)$$

$$E_{val}(g^-) = \frac{1}{k} \sum_{x_n \in D_{val}} e(g^-(x_n), y_n)$$

$E_{val}(E_{val}(g^-)) = E_{out}(g^-)$ ⇑ $E_{val}(e) = E_{x_n}(e) = E_{out}(g^-)$ — e depend only on x_n

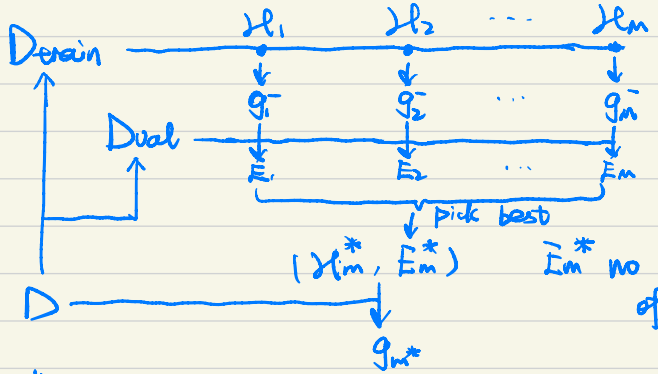
$\text{Var}(E_{val}(h)) = \frac{\sigma^2}{k}$

$E_{val}(h) = E_{out}(h) \pm O\left(\frac{1}{\sqrt{k}}\right)$

k 小: hard to exact $E_{out}(g^-)$ with $E_{val}(g^-)$

k 大: g^- is worse (less data point for training)

Model selection with validation



Decision Making involved

$(\mathcal{H}_m^*, \bar{E}_m^*)$ $\bar{E}_m^*$ no longer unbiased estimator of $\bar{E}_{out}(g_m^-)$

g_{m^*}

⇓ Hoeffding

$$\bar{E}_{out}(g_{m^*}^-) \le \bar{E}_{val}(g_{m^*}^-) + O\left(\sqrt{\frac{\ln M}{K}}\right)$$

bound is tighter than all Dtrain since $\mathcal{H}_{val} = \{g_1^- \cdots g_M^-\}$ is smaller

- Dtrain : totally contaminated
- Dval : slightly contaminated
- Dtest : totally clean

$$\bar{E}_{out}(g_{m^*}) \le \bar{E}_{out}(g_{m^*}^-) \le \bar{E}_{val}(g_{m^*}^-) + O\left(\sqrt{\frac{\ln M}{K}}\right)$$

Cross Validation

$$\bar{E}_{out}(g) \overset{\text{small } K}{\approx} \bar{E}_{out}(g^-) \overset{\text{large } K}{\approx} \bar{E}_{val}(g^-)$$

leave-one-out cross validation

① take data point (x_n, y_n) away and train with rest → g_n^-

② $e_n = \bar{E}_{val}(g_n^-) = e(g_n^-(x_n), y_n)$

③ $E_{cv} = \frac{1}{N}\sum_{n=1}^{N} e_n$

Why a good estimator?

Define $\bar{E}_{out}(N) \triangleq \bar{E}_D[\bar{E}_{out}(g)]$

$$\bar{E}_D[e_n] = \bar{E}_{D_n}\left[\bar{E}_{(x_n,y_n)}[e(g_n^-(x_n), y_n)]\right]$$
$$= \bar{E}_{D_n}[\bar{E}_{out}(g_n^-)]$$
$$= \bar{E}_{out}(N-1)$$

Cross validation can be used for model selection
For computational burden leave one out → k-fold

Theory vs Practice

Learning theory is guideline but not conclusive proof.

1. noise (stochastic + deterministic) leads to overfitting
2. Regularization helps to prevent overfitting by constraining the model, reducing the impact of the noise while still giving flexibility to fit the data.
3. Validation/CV are useful for estimating E_{out}. One important use is model selection

Kernel Method

Give 2 points x and $x' \in \mathcal{X}$

$k(x, x') = z^T z'$ for $z = \phi(x)$ in higher dimension

Compute $k(x, x')$ without ~~transforming~~ x and x'

Polynomial kernel

$\mathcal{X} = \mathbb{R}^d$, $\phi: \mathcal{X} \to \mathcal{Z}$ polynomial of order Q

$$k(x, x') = (1 + x^T x')^Q$$

$$= (1 + x_1 x_1' + x_2 x_2' + \cdots + x_d x_d')^Q$$

$$= z^T z'$$

stay here ↑ too many terms

As long as $k(x, x')$ is inner product in some $\mathcal{Z}$ space → good

Mercer's condition

$k(x, x')$ is a valid kernel iff

1. It is symmetric
2. Matrix $[k(x_i, x_j)]_{ij}$ is positive semi-definite for any $x_1, \cdots x_N$

Radial Basis Function

Each $(x_n, y_n) \in \mathcal{X}$ influences $h(x)$ based on $\|x - x_n\|$ radial

$$h(x) = \sum_{n=1}^{N} w_n \underbrace{\exp(-\gamma \|x - x_n\|^2)}_{\text{Basis Function}}$$

contribution of n on $h(x)$.

Solve for w_n

When $E_{in} = 0$: $h(x_n) = y_n$ for $n = 1, \cdots, N$

$$\sum_{m=1}^{N} w_m \exp(-\gamma \|x_n - x_m\|^2) = y_n \quad \text{for } n = 1, \cdots N$$

$$\Updownarrow$$

$$\begin{bmatrix} \exp(-\gamma\|x_1 - x_1\|^2) & \cdots & \exp(-\gamma\|x_1 - x_N\|^2) \\ \vdots & & \vdots \\ \exp(-\gamma\|x_N - x_1\|^2) & & \exp(-\gamma\|x_N - x_N\|^2) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

RBF with k centers

$N \gg k$: $u_1 \cdots u_k$ instead of $x_1 \cdots x_N$

$$h(x) = \sum_{k=1}^{k} w_k \exp(-\gamma \|x - u_k\|^2)$$

$\Downarrow$ estimate u_k with k-mean, not contaminated

$$\begin{bmatrix} \exp(-\gamma\|x_1 - u_1\|^2) & \cdots & \exp(-\gamma\|x_1 - u_k\|^2) \\ \vdots & & \vdots \\ \exp(-\gamma\|x_N - u_1\|^2) & \cdots & \exp(-\gamma\|x_N - u_k\|^2) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} \approx \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

$\tilde{\Phi}$ $\quad w$ $\quad y$

$$w = (\tilde{\Phi}^T \tilde{\Phi}) \tilde{\Phi}^T y$$

RBF can be derived purely on regularization

$$\sum_{n=1}^{N} (h(x_n) - y_n)^2 + \lambda \sum_{k=0}^{\infty} a_k \int_{-\infty}^{\infty} \left(\frac{d^k h}{dx^k}\right)^2 dx$$

smoothest interpolation

Learning Principles

Occam's Razor

The simplest model that fits the data is also the most plausible ☆

1. What's a simple model?
   - → Complexity of h: Minimum description length (MDL), order of polynomial
   - Complexity of $\mathcal{H}$: Entropy, VC dimension

   Link between 2 complexity

   l bits specify $f \rightarrow h$ is one of 2^l of a set $\mathcal{H}$

   exception: Looks complex but is one of few - SVM

2. Why is simpler better

   better: better out-of-sample performance

   - There're fewer simple hypotheses than complex ones
     ⇓ $m_{\mathcal{H}}(N)$
   - Less likely to fit a given data set
     ⇓ $m_{\mathcal{H}}(N) / 2^N$
   - More significance when it happens

Sampling Bias

If data is sampled in a biased way. Learning will produce a biased outcome ☆.

Data Snooping

If a data set has affected any step in the learning process its ability to assess the outcome has been compromised ☆

most common trap for practioners

When reusing data set: Trying one model after another on the same data set, you will eventually succeed

1. avoid data snooping: serict discipline
2. account for data snooping: how much data contamination

Discrete Latent Variable

Modeling of distribution over observed/latent variables

k-mean

Objective: $\min_{r_{nk}, \mu_k} \sum_n \sum_k r_{nk} \|x_n - \mu_k\|^2$

Iterative process:

① Initialize μ_k

② Iterate until no change (convergence)

- E: keep μ_k fixed, Min J w.r.t r_{nk}

$$r_{nk} = \begin{cases} 1, & k = \arg\min_j \|x_n - \mu_j\|^2 \\ 0, & \text{otherwise} \end{cases}$$

- M: keep r_{nk} fixed, Min J w.r.t μ_k

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

EM for Mixture Gaussian

1. Initialize μ_k, Σ_k and π_k
2. E-step

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

3. M-step

$$\begin{cases} \mu_k^{new} = \frac{1}{N_k} \gamma(z_{nk}) x_n \\ \Sigma_k^{new} = \frac{1}{N_k} \sum \gamma(z_{nk}) (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \\ \pi^{new} = N_k / N \end{cases}$$

$$N_k = \sum_i \gamma(z_{nk})$$

4. Check change in log-likelihood and stop if converge.

Mixture Gaussian

① Binary: $p(z_k=1) = \pi_k$, $\sum \pi_k = 1$

② Conditional Gaussian $p(x|z_k=1) = \mathcal{N}(x|\mu_k, \Sigma_k)$

$$\begin{cases} p(z) = \prod_{k=1}^{K} \pi_k^{z_k} \\ p(x|z) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k} \end{cases} \Rightarrow \boxed{\begin{aligned} p(x) &= \sum_z p(z)p(x|z) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \end{aligned}}$$

$z_k=1$ {
- prior: π_k
- posterior: $\gamma(z_k) = p(z_k=1|x) = \dfrac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}$

Log-Likelihood

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right)$$

$X: N \times D$ $\quad Z: N \times K$

EM in general

General technique to find MLE for probabilistic model with latent variables

Goal: Maximize likelihood over all latent variables

$$\max P(X|\theta) = \underbrace{\sum_Z P(X,Z|\theta)}_{\text{discrete } Z} = \underbrace{\int P(X,Z|\theta)dZ}_{\text{continuous } Z}$$

θ: parameter

Z: latent variables, X: observed variables

Idea:

optimization over $P(X|\theta)$ is difficult due to latent but optimization over $P(X,Z|\theta)$ is easier

$\Downarrow$ introduce $q(Z)$ on latent variable, for any $q(Z)$

$P(X,Z|\theta) = P(Z|X,\theta)P(X|\theta)$

$$\ln P(X|\theta) = \underbrace{\sum_Z q(Z)\ln\left(\frac{P(X,Z|\theta)}{q(Z)}\right)}_{\mathcal{L}(q,\theta)} - \underbrace{\sum_Z q(Z)\ln\left(\frac{P(Z|X,\theta)}{q(Z)}\right)}_{KL(P\|q)}$$

$$= \sum_Z q(Z)\ln\left(\frac{P(X,Z|\theta)}{P(Z|X,\theta)}\right)$$

$$= \sum_Z q(Z)\ln P(X|\theta) = \ln P(X|\theta)\underbrace{\sum_Z q(Z)}_{1}$$

$KL(P\|q)$: KL Divergence of $\begin{cases} \text{posterior } P(Z|X,\theta) \\ q(Z) \end{cases}$

$\underbrace{\geq 0}$

$\Downarrow$

$\mathcal{L}(q,\theta)$ is a lower bound on $\ln P(X|\theta)$

"=" hold when $KL(P\|q) = 0$

$\Updownarrow$

$q(Z) = P(Z|X,\theta)$

1. E-step

holding θ^{old} constant, changing $q(Z)$ to maximize lower bound $\mathcal{L}(q, \theta)$

$\Downarrow$ $p(X|\theta^{old})$ does not depend on $q(Z)$, reach maximum when $KL(p\|q) = 0$

$q(Z) = p(Z|X, \theta^{old})$

2. M-step

holding $q(Z)$ constant, changing θ to maximize lower bound $\mathcal{L}(q, \theta)$

$\theta^{new} = \arg\max \mathcal{L}(q, \theta)$

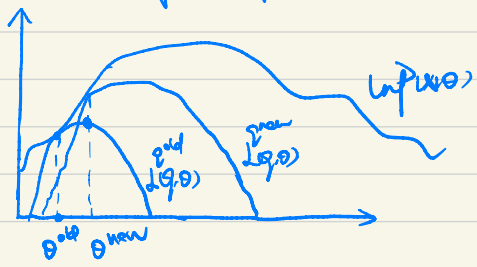
$\Downarrow q(Z) = p(Z|X, \theta^{old})$

$$\mathcal{L}(q, \theta) = \sum_Z p(Z|X, \theta^{old}) \ln p(X, Z|\theta) - \sum_Z p(Z|X, \theta^{old}) \ln p(Z|X, \theta^{old})$$

entropy of q

$$= Q(\theta, \theta^{old}) + \text{const}$$

expectation of complete data log likelihood



Another Perspective

Likelihood $L(\theta) = \log P(x|\theta) = \log \int P(x, z|\theta)\, dz$

$$= \log \int q(z) \frac{P(x,z|\theta)}{q(z)}\, dz \overset{\text{Jensen's}}{\geq} \underbrace{\int q(z) \log \frac{P(x,z|\theta)}{q(z)}\, dz}_{\text{Lower Bound}}$$

E-step

$$q_k(z) = \underset{q(z)}{\operatorname{argmax}} \int q(z) \frac{P(x,z|\theta_{k-1})}{q(z)}\, dz$$

$$\Rightarrow q_k(z) = P(z|x, \theta_{k-1})$$

M-step

$$\theta_k = \underset{\theta_k}{\operatorname{argmax}} \int q_k(z) \frac{P(x,z|\theta)}{q(z)}\, dz$$

$\Downarrow$

Local Maximum

Continuous Latent

Variables

PCA

Formulation

1. Maximum variance formulation

$\{x_n\}$ $n=1,\cdots,N$ $\quad x_n \in \mathbb{R}^D$ $\quad \bar{x} = \frac{1}{N}\sum x_n$

$u_1^T x_n$ projection as a scalar

var: $\frac{1}{N}\sum (u_1^T x_n - u_1^T \bar{x}) = u_1^T S u_1$, $\quad S = \frac{1}{N}\sum (x_n - \bar{x})(x_n - \bar{x})^T$

For following $u_2, \cdots$ optimizing orthogonal to previous transformation vectors

Max $u_1^T S u_1$

s.t $u_1^T u_1 = 1$ $\Rightarrow$ scale

$\Downarrow$ Lagrangian

$u_1^T S u_1 + \lambda_1 (1 - u_1^T u_1)$

$\Downarrow$ first order

$S u_1 = \lambda u_1$ must be an eigenvector

2. Minimum Error formulation

Introducing complete orthonormal basis vector $\{u_i\}$ $i=1,\cdots,D$

$x_n = \sum_{i=1}^{D} \alpha_{ni} u_i$

coordinate switch: $(x_{n1}, \cdots x_{nD}) \rightarrow (\alpha_{n1}, \cdots \alpha_{nD})$

$\Downarrow$ $\alpha_{nj} = x_n^T u_j$ new coordinate with basis u

$x_n = \sum_{i=1}^{D} (x_n^T u_i) u_i$

$\tilde{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$

z_{ni}: depend on particular data point

b_i: same for all

$\min_{u,z,b} \frac{1}{N}\sum \| x_n - \tilde{x}_n \|^2$

$\Downarrow$

$\frac{\partial}{\partial z} = 0 \Rightarrow z_{nj} = x_n^T u_j$

$\frac{\partial}{\partial b} = 0 \Rightarrow b_j = \bar{x}^T u_j$

$\tilde{x}_n$ definition

$x_n - \tilde{x}_n = \sum_{i=M+1}^{D} \{(x_n - \bar{x})^T u_i\} u_i$

$$\hat{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

$$\Downarrow \quad x_n = \sum_{i=1}^{D} (x_n^T u_i) u_i$$

$$\min \| x_n - \hat{x}_n \|^2$$

$$x_n - \hat{x}_n = \sum_{i=M+1}^{D} \{ (x_n - \bar{x})^T u_i \} u_i$$

$$\Downarrow$$

$$\min J = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=M+1}^{D} u_i^T S u_i$$

$$\Downarrow$$

$$S u_i = \lambda_i u_i, \quad J = \sum_{i=M+1}^{D} \lambda_i$$

Application

1. data compression

$$\hat{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

$$\Downarrow \quad z_{nj} = x_n^T u_j, \quad b_j = \bar{x} u_j$$

$$= \sum_{i=1}^{M} (x_n^T u_i) u_i + \sum_{i=M+1}^{D} (\bar{x}^T u_i) u_i$$

$$= \bar{x} + \underbrace{\sum_{i=1}^{M} (x_n^T u_i - \bar{x} u_i) u_i}$$ only consider deviation from mean for first M

2. normalization

$$S U = U \Lambda$$

$$[\cdots u_i \cdots) \quad (\lambda_1 \cdots \lambda_D)$$

$$y_n = \Lambda^{-\frac{1}{2}} U^T (x_n - \bar{x})$$

$$\begin{cases} \bar{y} = 0 \\ \frac{1}{N} \sum y_n y_n^T = I \end{cases}$$

3. Dimension Reduction

transform: $z_{n \times k} = X_{n \times p} U_{p \times k}$ first k eigenvector

decode: $\hat{x}_{n \times p} = z_{n \times k} U_{k \times p}^T$

Probabilistic PCA

$x = Wz + u + \varepsilon$

- $p(z) = \mathcal{N}(z|0, I)$ dim M
- $p(x|z) = \mathcal{N}(x|Wz+u, \sigma^2 I)$ dim D

Likelihood: $p(x) = \int p(x|z)p(z)dz = \mathcal{N}(x|u, C)$

$C = \mathrm{cov}(x) = E[(Wz+\varepsilon)(Wz+\varepsilon)^T]$

$= WW^T + \sigma^2 I$

Posterior: $p(z|x) = \mathcal{N}(z|M^{-1}W^T(x-u), \sigma^2 M^{-1})$

$M = W^TW + \sigma^2 I$

MLE

$\ln p(X|u, W, \sigma^2) = \sum_n \ln p(x_n|W, u, \sigma^2)$

set derivative $\hat{u} = \bar{x}$

$\hat{W} = U_M (L_M - \sigma^2 I)^{\frac{1}{2}} R$

$\hat{\sigma}^2 = \frac{1}{D-M} \sum_{i=M+1}^{D} \lambda_i$

EM

① Write down complete-data log likelihood

$\ln p(X, Z|u, W, \sigma^2) = \sum_n \{\ln p(x_n|z_n) + \ln p(z_n)\}$

② Take expectation with respect to posterior of latent

$E[\ln p(X, Z|u, W, \sigma^2)] = -\sum_n \{\frac{D}{2}\ln(2\pi\sigma^2) + \frac{1}{2}\mathrm{Tr}[E[z_n z_n^T]]$

$+ \frac{1}{2\sigma^2}\|x_n - u\|^2 - \frac{1}{\sigma^2}E[z_n]^T W^T(x_n - u)$

$+ \frac{1}{2\sigma^2}\mathrm{Tr}(E[z_n z_n^T]W^TW) + \frac{M}{2}\ln(2\pi)\}$

③ Use old param to evaluate (E-step)

$E[z_n] = M^{-1}W^T(x_n - \bar{x})$

$E[z_n z_n^T] = \sigma^2 M^{-1} + E[z_n]E[z_n]^T$

④ Maximize keep posterior fixed (M-step)

$W_{new} = [\sum_n (x_n - \bar{x})E[z_n]^T][\sum_n E[z_n z_n^T]]$

$\sigma^2_{new} = \frac{1}{ND}\sum_n \{\|x_n - \bar{x}\|^2 - 2E[z_n]^T W_{new}^T(x_n - \bar{x}) + \mathrm{Tr}(E[z_n z_n^T]W_{new}^T W_{new})\}$

Factor Analysis

$$y = \Lambda x + \varepsilon \qquad \varepsilon \sim \mathcal{N}(0, \Psi) \rightarrow \text{diagonal}$$

Λ: factor loading matrix

Non-uniqueness of Λ:

- orthogonal matrix U: $UU^T = I$
- $y = \Lambda x + \varepsilon = \Lambda U U^T x + \varepsilon = (\Lambda U)(U^T x) + \varepsilon$

Connection with PCA:

when $\Psi = \sigma^2 I$, Λ is made of the first k eigenvectors of covariance matrix of y.

EM

E: estimate x with parameters from last step

M: adjust parameters by maximizing the lower bound on the log-likelihood.

Sequential Data

Sequence Latent State Model

Markov Model

No assumption:

$$P(x_1, \cdots x_N) = P(x_1) \prod_{n=2}^{N} P(x_n | x_1, \cdots, x_{n-1})$$

2-nd order Markov

$$P(x_1, \cdots x_N) = P(x_1) P(x_2 | x_1) \prod_{n=3}^{N} P(x_n | x_{n-1}, x_{n-2})$$

State Space Model

$$P(x_1, \cdots x_N, z_1, \cdots z_N) = P(z_1) \prod_{n=2}^{N} P(z_n | z_{n-1}) \prod_{n=1}^{N} P(x_n | z_n)$$

Sequence latent state models $P(x|z)$

- parametric
  - State-space model: continuous z
  - Hidden Markov model: discrete z
- Non-parametric: RNN, LSTM, Bayesian neural networks

$$P(x_n | x_{n-1}) \rightarrow P(x_n | z_n)$$

HMM (Hidden Markov Model)

$$P(X, Z|\theta) = \underbrace{P(z_1|\pi)}_{\text{initial state}} \underbrace{\left[\prod_{n=2}^{N} P(z_n|z_{n-1}, A)\right]}_{\text{transition}} \underbrace{\prod_{m=1}^{N} P(x_m|z_m, \phi)}_{\text{conditional probability}}$$

$\theta = \{\pi, A, \phi\}$

$X = \{x_1, \cdots, x_N\}$ observed variable

$Z = \{z_1, \cdots, z_N\}$ state variable

EM for HMM

① E-step : find the posterior $P(Z|X, \theta^{old})$

$K \times 1$: $\gamma(z_n) = P(z_n|X, \theta^{old}) = \alpha(z_n)\beta(z_n)/P(X)$

$K \times K$: $\xi(z_{n-1}, z_n) = P(z_{n-1}, z_n|X) = \alpha(z_{n-1})P(x_n|z_n)P(z_n|z_{n-1})\beta(z_n)/P(X)$

$$\begin{aligned}
\alpha(z_n) &= P(x_1, \cdots x_n, z_n) \\
&= P(x_n|z_n) \sum_{z_{n-1}} P(x_1, \cdots, x_{n-1}, z_{n-1}) P(z_n|z_{n-1}) \\
&= P(x_n|z_n) \sum_{z_{n-1}} \alpha(z_{n-1}) P(z_n|z_{n-1})
\end{aligned}$$

$$\begin{aligned}
\beta(z_n) &= P(x_{n+1}, \cdots, x_N|z_n) \\
&= \sum_{z_{n+1}} P(x_{n+2}, \cdots, x_N|z_{n+1}) P(x_{n+1}|z_{n+1}) P(z_{n+1}|z_n) \\
&= \sum_{z_{n+1}} \beta(z_{n+1}) P(x_{n+1}|z_{n+1}) P(z_{n+1}|z_n)
\end{aligned}$$

② M-step : Maximize posterior expectation of the log complete likelihood

$$\begin{aligned}
Q(\theta, \theta^{old}) &= \sum_z P(z|X, \theta^{old}) \ln P(X, z|\theta) \\
&= \sum_k \gamma(z_{1k}) \ln \pi_k + \sum_n \sum_j \sum_k \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\
&\quad + \sum_n \sum_k \gamma(z_{nk}) \ln P(x_n|\phi_k)
\end{aligned}$$

⇓ Lagrangian Multiplier

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_j \gamma(z_{1j})}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_k \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}$$

ϕ_k only depend on the last term

$$\mu_k = \frac{\sum_n \gamma(z_{nk}) x_n}{\sum_n \gamma(z_{nk})}$$

$$\Sigma_k = \frac{\sum_n \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T}{\sum \gamma(z_{nk})}$$

State Space Model

$$z_n = Az_{n-1} + w_n \qquad w \sim \mathcal{N}(w|0,\Gamma)$$
$$x_n = Cz_n + v_n \qquad v \sim \mathcal{N}(v|0,\Sigma)$$
$$z_1 = \mu_0 + u \qquad u \sim \mathcal{N}(u|0,P_0)$$
$$\theta = \{A, \Gamma, C, \Sigma, \mu_0, P_0\}$$

EM { E: inference problem: posterior marginals for latent
M: learning problem: update θ

Inference : E step

posterior marginal $\hat{\gamma}(z_n) = p(z_n|x_1,\cdots x_n) = \mathcal{N}(z_n|\mu_n, V_n)$

$$\Downarrow$$

$$\mu_n = \underbrace{A\mu_{n-1}}_{\text{prediction}} + \underbrace{k_n(x_n - CA\mu_{n-1})}_{\text{correction}}$$
$$V_n = (I - k_nC)P_{n-1}$$
$$p(z_{n-1}|x_1,\cdots x_{n-1}) \to p(z_n|x_1,\cdots x_{n-1}) \xrightarrow{x_n} p(z_n|x_1,\cdots x_n)$$
$$c_n = p(x_n|x_1,\cdots x_{n-1})$$
$$= \mathcal{N}(x_n|CA\mu_{n-1}, CP_{n-1}C^T + \Sigma)$$

- $P_{n-1} \equiv AV_{n-1}A^T + \Gamma$
- Kalman Gain : $k_n = P_{n-1}C^T(CP_{n-1}C^T + \Sigma)^{-1}$

Marginal $\gamma(z_n) = p(z_n, X)/p(X) = \alpha(z_n)\beta(z_n)/p(X)$
$$= \mathcal{N}(z_n|\hat{\mu}_n, \hat{V}_n)$$

$$\Downarrow$$

$$\hat{\mu}_n = \mu_n + J_n(\hat{\mu}_{n+1} - A\mu_n)$$
$$\hat{V}_n = V_n + J_n(\hat{V}_{n+1} - P_n)J_n^T$$

- $J_n = V_nA^T(P_n)^{-1}$
- $\mathrm{cov}(z_{n+1}, z_n) = J_{n-1}\hat{V}_n$

Learning : M step

$$\ln p(X, Z|\theta) = \ln p(z_1|\mu_0, P_0) + \sum_{n=2}^{N} \ln p(z_n|z_{n-1}, A, \Gamma) + \sum_{n=1}^{N} \ln p(x_n|z_n, C, \Sigma)$$

↓ expectation with $p(Z|X, \theta^{old})$ posterior distribution

$$Q(\theta, \theta^{old}) = E_{Z|\theta^{old}}[\ln p(X, Z|\theta)]$$

⇓ maximize

$$\mu_0^{new} = E(z_1)$$

$$P_0^{new} = E(z_1 z_1^T) - E(z_1)E(z_1)^T$$

$$A^{new} = \left(\sum_{n=2}^{N} E(z_n z_{n-1}^T)\right)\left(\sum_{n=2}^{N} E(z_{n-1} z_{n-1}^T)\right)^{-1}$$

$$\Gamma^{new} = \frac{1}{N-1}\sum_{n=2}^{N}\{E(z_n z_n^T) - A^{new}E(z_{n-1} z_n^T) - E(z_n z_{n-1}^T)(A^{new})^T + A^{new}E(z_{n-1} z_{n-1}^T)(A^{new})^T\}$$

$$C^{new} = \left(\sum_{n=1}^{N} x_n E(z_n^T)\right)\left(\sum_{n=1}^{N} E(z_n z_n^T)\right)^{-1}$$

$$\Sigma^{new} = \frac{1}{N}\sum_{n=1}^{N}\{x_n x_n^T - C^{new}E(z_n)x_n^T - x_n E(z_n^T)C^{new} + C^{new}E(z_n z_n^T)C^{new}\}$$

- $E(z_n) = \hat{\mu}_n$
- $E(z_n z_{n-1}^T) = \hat{V}_n J_{n-1}^T + \hat{\mu}_n \hat{\mu}_{n-1}^T$
- $E(z_n z_n^T) = \hat{V}_n + \hat{\mu}_n \hat{\mu}_n^T$

High Dimensional Data

① PCA: projection that spreads data
② Multidimensional-scaling: retain original distance
③ Stochastic neighbor embedding: non linear method to keep close points together

MDS (multidimensional scaling)

$$Y = \arg\min_{Y} \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} - \|y_i - y_j\|^2)$$

$$\begin{cases} D_{ij} = \text{dist}(x_i, x_j) & x_i \in \mathbb{R}^p \\ y_i \in \mathbb{R}^q & q < p \end{cases}$$

$$\Updownarrow \; D_{ij} = \|x_i - x_j\|_2^2$$

$$\min_{Y} \text{trace}(XX^T - YY^T)^2$$

$$\Rightarrow Y = V\Lambda, \quad \Lambda \text{ eigenvalue of } XX^T$$

SNE (stochastic neighbor embedding)

① probability of picking j as neighbor of i

$$P_{ij} = \frac{\exp(-D_{ij}^2)}{\sum_{k \neq i} \exp(-D_{ki}^2)}$$

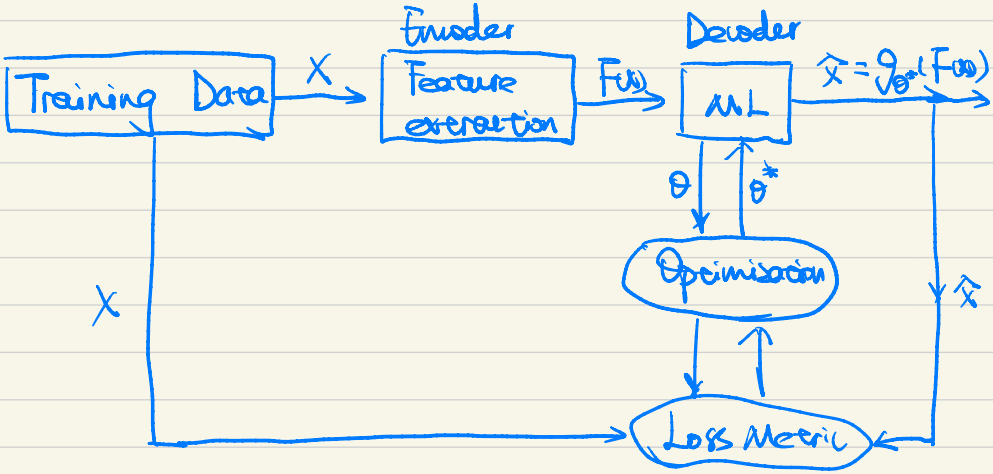
② probability in low dimension

$$\begin{cases} q_{ij} = \dfrac{\exp(-\|y_i - y_j\|_2^2)}{\sum_{k \neq i} \exp(-\|y_k - y_i\|_2^2)} & \text{SNE} \\ q_{ij} = \dfrac{(1 + \|y_i - y_j\|_2^2)^{-1}}{\sum_{k \neq i} (1 + \|y_k - y_i\|_2^2)^{-1}} & \text{t-SNE / reduce crowding} \end{cases}$$

$$\min KL(P\|Q) = \sum_{i \neq j} P_{ij} \log \frac{P_{ij}}{q_{ij}}$$

good to remain local pattern: KL not symmetric

Auto encoder



For PCA, keep both F and g linear

$F(x) = XV$ eigen

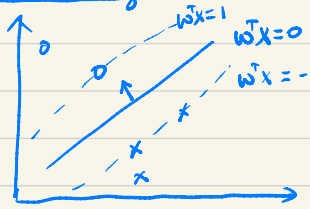
$\hat{x} = F(x)V^T$

Min construction error $\hat{x}, x$

Support Vector Machine

S V M

Hard Margin SVM



idea: look for fat margin

↓

few dichotomies → better generalization

$$\text{Min } \frac{1}{2}w^Tw$$

$$\text{s.t } y_n(w^Tx_n+b)\geq 1 \quad \text{for } n=1,\cdots,n$$

Lagrangian:

$$L(w,b,\alpha)=\frac{1}{2}w^Tw-\sum_{n=1}^{N}\alpha_n(y_n(w^Tx_n+b)-1)$$

$$\text{s.t } \alpha_n\geq 0$$

plug ⇓

$$\frac{\partial L}{\partial w}=w-\sum_{n=1}^{N}\alpha_n y_n x_n=0$$

$$\frac{\partial L}{\partial b}=-\sum_{n=1}^{N}\alpha_n y_n=0$$

$$L(w,b,\alpha)=L(\alpha)=\sum_{n=1}^{N}\alpha_n-\frac{1}{2}\sum_{n}\sum_{m}y_n y_m\alpha_n\alpha_m x_n^Tx_m$$

⇓ max against $\begin{cases}\alpha_n\geq 0\\ \sum\alpha_n y_n=0\end{cases}$

solve with a quadratic programming

$$\min_{\alpha}\frac{1}{2}\alpha^T\begin{bmatrix}y_1y_1x_1^Tx_1 & \cdots & y_1y_Nx_1^Tx_N\\ \vdots & & \vdots\\ y_Ny_1x_N^Tx_1 & & y_Ny_Nx_N^Tx_N\end{bmatrix}\alpha+(-1^T)\alpha$$

$$\text{s.t } y^T\alpha=0$$

$$0\leq\alpha\leq\infty$$

replace $x_n^Tx_m$ with $k(x_n,x_m)$ for kernel

⇓ α_n

$$w=\sum_{n=1}^{N}\alpha_n y_n x_n$$

⇓ KKT condition

$\alpha_n>0 \Rightarrow x_n$ is a support vector

Generalization depend on SV: $\mathbb{E}[E_{out}]\leq\frac{\mathbb{E}[\#\text{ of SV}]}{N-1}$

Soft Margin SVM

$$\text{Min } \frac{1}{2}w^T w + C\sum \xi_n$$

$$\text{s.t } \quad y_n(w^T x_n + b) \geq 1 - \xi_n$$

$$\xi_n \geq 0$$

Lagrangian

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2}w^T w + C\sum \xi_n - \sum \alpha_n (y_n(w^T x_n + b) - 1 + \xi_n) - \sum \beta_n \xi_n$$

$$\frac{\partial L}{\partial w} = w - \sum \alpha_n y_n x_n = 0$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_n y_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = C - \alpha_n - \beta_n = 0$$

$$L(\alpha) = \sum \alpha_n - \frac{1}{2}\sum_n \sum_m y_n y_m \alpha_n \alpha_m x_n^T x_m$$

$$\text{s.t } \quad 0 \leq \alpha_n \leq C$$

$$\sum \alpha_n y_n = 0$$

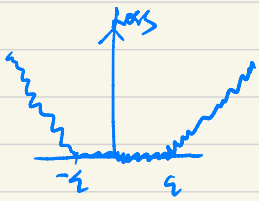
$\Downarrow \alpha_n$

$$w = \sum \alpha_n y_n x_n$$

SV

- $\xi_n = 0 \ (0 < \alpha_n < C) \ : \ y_n(w^T x_n + b) = 1$
- $\xi_n > 0 \ (\alpha_n = C) \ : \ y_n(w^T x_n + b) < 1$

SVM regression



Does not care error smaller than ε

Make function f as flat as possible → less sensitive to noise

$f(x) = w^T x + b$

$$\min \frac{1}{2} w^T w + C \sum (\xi_n + \xi_n^*)$$

$$\text{s.t} \quad y_i - w^T x - b \le \varepsilon + \xi_n \qquad n = 1 \cdots N$$

$$w^T x + b - y_i \le \varepsilon + \xi_n^*$$

$$\xi_n, \xi_n^* \ge 0$$

⇓ Lagrangian

$$L := \frac{1}{2} w^T w + C \Sigma (\xi_n + \xi_n^*) - \Sigma (\eta_n \xi_n + \eta_n^* \xi_n^*)$$
$$- \Sigma \alpha_n (\varepsilon + \xi_n - y_n + w^T x_n + b)$$
$$- \Sigma \alpha_n^* (\varepsilon + \xi_n^* + y_n - w^T x_n - b)$$

where $\alpha_i, \alpha_i^* \ge 0$

$$\frac{\partial L}{\partial b} = \Sigma (\alpha_n^* - \alpha_n) = 0$$
$$\frac{\partial L}{\partial w} = w - \Sigma (\alpha_n - \alpha_n^*) x_n = 0$$
$$\frac{\partial L}{\partial \xi_n} = C - \alpha_n - \eta_n = 0$$

dual: $\max L = -\frac{1}{2} \Sigma_n \Sigma_m (\alpha_n - \alpha_n^*)(\alpha_m - \alpha_m^*) x_n^T x_m$
$$- \varepsilon \Sigma (\alpha_n + \alpha_n^*) + \Sigma y_n (\alpha_n - \alpha_n^*)$$

$$\text{s.t} \quad \Sigma (\alpha_n - \alpha_n^*) = 0$$

⇓ α_n

$$f(x) = \Sigma (\alpha_n - \alpha_n^*) x_n^T x + b$$

vanish for nonbinding

Ensemble Learning

Ensemble: a group of predictors. Aggregate the prediction of a group of weak learners

1. Bagging: sample data and train model independently
   (- variance) Usually homogeneous model

2. Boosting: learn sequentially in a adaptive way
   (- bias) Usually homogeneous model
   ① Adaboost
   for t in 1, ..., T
   a. weak learner: $h_t(x)$
   $h_t(x) = \arg\min \varepsilon_t = \arg\min \sum w_{i,t} \varepsilon_{i,t}$
   b. update weight, learner
   $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
   $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
   c. update error rate → larger for failed sample
   $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$, renormalize
   ② Gradient Boosting
   Fit new learner on previous residual
   $F_{m+1}(x) = F_m(x) + h_m(x) = y_i$
   ⇕
   $h_m(x) = y_i - F_m(x)$

3. Stacking (- prediction error) learn a meta model to combine weak learners. Usually on heterogeneous weak learns

Tree Model

Decision Tree

Divide into subset

Measure of a node

classifier:

① Gini impurity

$$G_i = 1 - \sum_k p_{i,k}^2$$

$p_{i,k}$: proportion of class k in node i

② Entropy

$$H_i = -\sum_{\substack{k \\ p_{i,k} \neq 0}} p_{i,k} \log(p_{i,k})$$

regression

$$MSE_n = \sum (y_i - \bar{y}_n)^2$$

CART Algorithm

classification and regression tree — greedy method

At each step, choose feature and threshold to minimize

$$J = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

Random Forest

Decision tree + bagging

feature importance → average depth

Network Analysis

Spectral Clustering

- Spanning Tree: connect all nodes while minimizing sum of edges
- Steiner Tree: connect selected nodes while minimizing edges
  - NP-problem: need to decide which nodes to include
  - Polynomial Approximation: with spanning tree over those selected nodes

- Community Detection: detect subset of nodes that're more densly connected
- Clustering: determine subsets of points that are 'close' to each other given a similarity measure

Quality function: function that assigns a number to partition

Modularity: $Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(c_i, c_j)$

① P_{ij}: expected edge density between i and j under null

② Erdos-Renyi: $P_{ij} = 2m/n(n-1)$

③ Configuration: $P_{ij} = k_i k_j / (2m-1)$

Modularity maximization: NP

- Louvain Method

Ideal Cluster: identify nodes that share edges more among themselves compared to outside of themselves.

$s_i = \begin{cases} 1 & : \text{group } 1 \\ -1 & : \text{group } 2 \end{cases} \Rightarrow \quad C = \min_{s \in \{-1,1\}^n} \sum_{ij} A_{ij} (1 - s_i s_j)$

n_1, n_2 size of 2 cluster s.t. $\sum s_k = n_1 - n_2$

Relaxation of clustering

① remove integer constraint for s

$$\hat{C}_1 = \min_{X \in \mathbb{R}^n} X^T L X, \quad \|X\| = 1$$

$L = D - A$ = adjacent matrix − degree matrix

⇓

spectral clustering

Time Series

Time Series Data

Time Series: a collection of realizations of distinct variable at equally spaced points in time.

R.V: $X_1 \cdots X_t \cdots X_n \quad (\Omega, P)$

$\Downarrow w$

$x_1 = X_1(w) \cdots x_t = X_t(w) \cdots x_n = X_n(w)$

① time series observations 可看作 stochastic process 下 single realization

② dependence 可帮助 predict，但也含有更多 statistical info

③ time interval 为当是 deterministic 下

Deterministic Dependence

常见体现于 mean $\mu_X(t) = E(X_t)$

① trend: $m_X(t)$

② seasonality: $S_X(t)$

Basic Stat

- Marginal Mean: $\mu_X(t) = E(X_t)$
- Marginal Variance: $Var(X_t) = E[(X_t - \mu_X(t))^2]$
- Autocovariance

$$\gamma_X(s,t) = cov(X_s, X_t) = E[(X_s - \mu_X(s))(X_t - \mu_X(t))]$$

现实世界中，R.V 只有 1 个 realization

$\Downarrow$ assumption ☆☆

stationary

✓ weak version:

$\mu_X(t) = \mu_X$, $Var(X_t) = \sigma_X^2$, $cov(X_s, X_t) = \gamma_X(s-t)$

✓ strong version:

distribution $X_{t+h}, \cdots X_{t+n+h}$ same for all t, n, h

Stochastic Dependence

$P_{X_t|X_s}$,

$\gamma_X(s,t)$ describe linear stochastic dependence

Stochastic Dependence

$P_{X_t|X_s}$

Why we need stationary

1. same distribution / stats so we can estimate from time series observations
2. Generalization of CLT / LLN when RV with large gap are kind of independent
3. Rules out trend, seasonality

$$X_t = T_t + S_t + \underbrace{Y_t}_{\text{stationary}}$$

How to get stationary

1. Remove linear regression on t → remove linear trend
2. Remove nonparametric regression on t
   i.e: kernal smoothing, polynomial, k-neighbor
   → remove trend / seasonality
3. Remove periodic regression on t → remove seasonality
4. Differenciation
   1st → remove linear trend / expanding variance
   2nd → remove polynomial trend / expanding variance
   other interval → remove seasonality
   ...
5. Transformation
   log → remove expanding variance

Empirical Estimator

$$\hat{\mu} = \bar{X}_n = \frac{1}{n}\sum_{t=1}^{n} X_t$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{t=1}^{n}(X_t - \hat{\mu})^2$$

$$\hat{\gamma}(h) = \frac{1}{n}\sum_{t=1}^{n-h}(X_t - \hat{\mu})(X_{t+h} - \hat{\mu}) \quad \text{for} \quad 1 \le h < n$$

$$\star\ \mathrm{var}(\hat{\mu}) = \frac{1}{n}\sigma^2 + \frac{2}{n^2}\sum_{h=1}^{n-1}(n-h)\gamma(h)$$

In order for $\mathrm{var}(\hat{\mu}) \to 0$, the second term need to decay sufficiently fast. $\Rightarrow$ estimate marginal mean from single path of realization

{ stationary
  dependence decay fast

$\Rightarrow$ can estimate expectation from mean in one realization path

Models

Autoregressive Process

AR(p): $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + w_t$

- ACF decay exponentially and never 0
- stationary under conditions

Random walk

No drift: $X_t = X_{t-1} + w_t = X_0 + w_1 + \cdots + w_t$

With drift: $Y_t = \delta t + X_t$

$= \delta + Y_{t-1} + w_t$

$= \delta t + Y_0 + w_1 + \cdots + w_t$

stat:

$\mu_X(t) = E(X_t) = E(X_0)$

$\gamma_X(t) = \mathrm{Var}(X_t) = \mathrm{Var}(X_0) + t\sigma_w^2$

$\gamma_X(s,t) = \mathrm{cov}(X_s, X_t) = \mathrm{Var}(X_0) + \min(s,t)\sigma_w^2$

$\mu_Y(t) = \delta t + E(Y_0)$

$\gamma_Y(t) = \mathrm{Var}(Y_0) + t\sigma_w^2$

$\gamma_Y(s,t) = \mathrm{Var}(Y_0) + \min(s,t)\sigma_w^2$

Moving Average

MA(q) $X_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q}$

$= \sum_{h=0}^{q} \theta_h w_{t-h}$

- $\gamma(h) = \sum_{j=0}^{q-h} \theta_j \theta_{j+h} \sigma_w^2$ for $0 \le h \le q$ $\Rightarrow$ stationary

ARMA / ARIMA

ARMA(p, q) $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p}$

$+ w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \theta_q w_{t-q}$

ARIMA: ARMA with differencing

Fit TS Regression

Overview

1. transform to stationary
   - log-transform
   - remove trend / seasonality
   - differentiate successively
2. check for white noise (ACF)
3. If stationary
   ① finite lag ⇒ MA(q)
      - compute ACF to get order
      - estimate with ML
      - check residual
   ② Otherwise ⇒ AR(p)
      - compute PACF to get order
      - estimate ϕ_k and residual variance $\hat{\sigma_w}$ via Yule-Walker
      - check residual
   ③ ARMA(p, q)
      - attempt to fit AR and compute residual
      - attempt to fit MA to residual
      - fit ARMA with p, q from previous steps
      - check residual

	ACF	PACF
AR(p)	decays	0 after p
MA(q)	0 after q	decays
ARMA(p,q)	decays	decays

AR(p) Parameter Estimation

AR(p): $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + w_t$

- p+1 parameters to estimate: $\phi_1, \cdots \phi_p, \sigma_w^2$
- If X_t stationary $\Rightarrow$ estimate with method of moment

$$(\gamma_X(0), \gamma_X(1), \cdots, \gamma_X(p)) = f(\phi_1, \phi_2, \cdots \phi_p, \sigma_w^2)$$

$$\Downarrow$$

Yule-Walker equation

p+1 unknow and p+1 equations

$$(\hat{\phi}_1, \hat{\phi}_2, \cdots \hat{\phi}_p, \hat{\sigma}_w^2) = f^{-1}(\hat{\gamma}_X(0), \cdots \hat{\gamma}_X(p))$$

$$\Downarrow$$

continuous mapping
Delta theorem

consistent estimator

Yule-Walker:

$$\begin{cases} E(X_t X_t) = E[\phi_1 X_t X_{t-1} + \phi_2 X_t X_{t-2} + \cdots + \phi_p X_t X_{t-p} + X_t w_t] \\ E(X_{t-1} X_t) = E[\phi_1 X_{t-1} X_{t-1} + \phi_2 X_{t-1} X_{t-2} + \cdots + \phi_p X_{t-1} X_{t-p} + X_{t-1} w_t] \\ \vdots \\ E(X_{t-p} X_t) = E[\phi_1 X_{t-p} X_{t-1} + \phi_2 X_{t-p} X_{t-2} + \cdots + \phi_p X_{t-p} X_{t-p} + X_{t-p} w_t] \end{cases}$$

$$\Updownarrow$$

$$\begin{cases} \gamma_X(0) = \phi_1 \gamma_X(1) + \phi_2 \gamma_X(2) + \cdots + \phi_p \gamma_X(p) + \sigma_w^2 \\ \gamma_X(1) = \phi_1 \gamma_X(0) + \phi_2 \gamma_X(1) + \cdots + \phi_p \gamma_X(p-1) \\ \vdots \\ \gamma_X(p) = \phi_1 \gamma_X(p-1) + \phi_2 \gamma_X(p-2) + \cdots + \phi_p \gamma_X(0) \end{cases}$$

Time Series Analysis

Partial Correlation

- Partial correlation of X, Y given Z
  - regress X, Y on Z
  - $P_{XY|Z} = \text{corr}(X - \hat{X}, Y - \hat{Y})$
- Partial correlation of X_t, X_{t-h} for stationary series
  - ACF: $\rho_X(h) = \text{corr}(X_h, X_0) = \frac{E[(X_h - E(X_0))(X_0 - E(X_0))]}{\text{Var}(X_0)}$
  - PACF: $\alpha_X(h) = \text{corr}(X_h - \hat{X}_h^{\text{lin } h-1}, X_0 - \hat{X}_0^{\text{lin } h-1})$
    linear projection on $X_1, \cdots X_{h-1}$
  - Frisch-Waugh-Lovell Theorem

    $X_t = \phi_1 X_{t-1} + \cdots + \phi_h X_{t-h} + \hat{X}_t$ is best linear prediction

    $\Downarrow$

    $\alpha_X(h) = \phi_h$

Information Criterion

$AIC(k) = \underbrace{-2 \log\text{-likelihood}}_{\text{model fit}} + \underbrace{2k}_{\text{complexity penalty}}$

$BIC(k) = -2 \log\text{-likelihood} + k \ln(n)$

$BIC(k) > AIC(k)$

Gaussian Process

Gaussian Process

N-dim multivariate Gaussian

joint:

$$X = \begin{bmatrix} X_1 & X_1 \in \mathbb{R}^d \\ X_2 & X_2 \in \mathbb{R}^{N-d} \end{bmatrix}$$

$$\mu = \begin{bmatrix} \mu_1 & \mu_1 \in \mathbb{R}^d \\ \mu_2 & \mu_2 \in \mathbb{R}^{N-d} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \Sigma_{11} \in \mathbb{R}^{d \times d} & \Sigma_{21} \in \mathbb{R}^{d \times (N-d)} \\ \Sigma_{21} \in \mathbb{R}^{(N-d) \times d} & \Sigma_{22} \in \mathbb{R}^{(N-d) \times (N-d)} \end{bmatrix}$$

conditional:

$$\mu_{X_1|X_2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2)$$

$$\Sigma_{X_1|X_2} = \Sigma_{11} - \Sigma_{21}\Sigma_{22}^{-1}\Sigma_{21}$$

$$X_1|X_2 \sim \mathcal{N}(\mu_{X_1|X_2}, \Sigma_{X_1|X_2})$$

Spacial Covariance

observed value: $y_1, \cdots y_N$

location: $x_1, \cdots x_N$

RBF: $\text{cov}(Y_i, Y_j) = k(x_i, x_j) = \exp\left[-\frac{\|x_i - x_j\|^2}{2\nu}\right]$

$$\Sigma_N = \begin{bmatrix} \text{cov}(Y_1, Y_1) & \cdots & \text{cov}(Y_1, Y_N) \\ \vdots & & \vdots \\ \text{cov}(Y_N, Y_1) & & \text{cov}(Y_N, Y_N) \end{bmatrix} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$$

$$k_*^T = [k(x_*, x_1), \cdots k(x_*, x_N)]$$

conditional:

$$\mu_{*|1:N} = \mu_* + k_*^T K_N^{-1}(y_{1:N} - \mu_{1:N})$$

$$\sigma^2_{*|1:N} = \sigma^2_* - k_*^T K_N^{-1} k_*$$

predict:

$$\hat{y}_* = \hat{f}(x_*) = \underbrace{\mu_*}_{\text{assume } 0} + \underset{1\times N}{k_*^T} \underset{N\times N}{K_N^{-1}} \underset{N\times 1}{y_{1:N}}$$

$=$

① $\sum_1^N w_i^* y_i$ weighted average of $y_{1:N}$

② $\sum_1^N k(x_*, x_i)\alpha_i$ linear combination of N non-linear features, same α for all prediction

Gaussian Process

a collection of random variables, any finite number of which is Gaussian

$$\begin{cases} m(x) = \mu_x = E[f(x)] \\ k(x, x') = \text{cov}(f(x), f(x')) \end{cases}$$

Measurement Noise

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim N(0, \tilde{v})$$

$$\Rightarrow \begin{cases} \mu_{*|1:N} = \mu_* + k_*^T (K_N + \tilde{v} I)^{-1} y_{1:N} \\ \Sigma_{*|1:N} = k(x_*, x_*) - k_*^T (K_N + \tilde{v} I)^{-1} k_* \end{cases}$$

adjust less due to noise

Entropy

Quantify uncertainty

$$H(Y) = \sum_Y P(y) \log \frac{1}{P(y)} = -\sum_Y P(y) \log P(y)$$

$\log \frac{1}{P(y)}$: measure of surprise

$H(Y)$: expected surprise

- Joint Entropy:

$$H(Y, Z) = -\sum_{Y,Z} P(y, z) \log P(y, z) \leq H(Y) + H(Z)$$

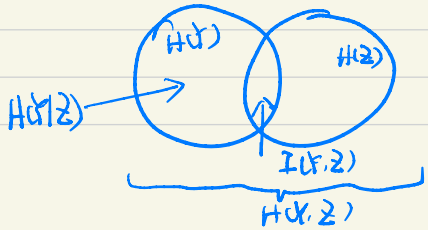
- Conditional Entropy:

$$H(Y|Z) = H(Y, Z) - H(Z)$$

- Mutual Information

$$I(Y, Z) = H(Y) - H(Y|Z) = \sum_{y,z} P(y, z) \log \frac{P(y, z)}{P(y) P(z)}$$

$\frac{P(y,z)}{P(y)P(z)}$: distance from independance → if independent



Neural Network

Forward Propagation

$$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

Vectorize over multiple samples

$$Z^{[l]} = [\, z^{[l](1)} \cdots z^{[l](m)} \,]$$

$b^{[l]}$ with broadcasting

$$Z^{[l]}_{(n^{[l]}, m)} = w^{[l]}_{(n^{[l]}, n^{[l-1]})} A^{[l-1]}_{(n^{[l-1]}, m)} + b^{[l]}_{(n^{[l]}, 1)}$$
$$A^{[l]} = g^{[l]}(Z^{[l]})$$

dim: $w^{[l]}, dw^{[l]} : (n^{[l]}, n^{[l-1]})$; $b^{[l]}, db^{[l]} : (n^{[l]}, 1)$

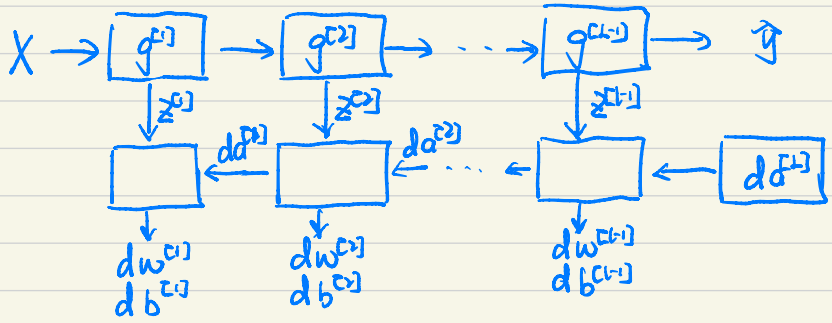
Loop through l for forward propagation

Backward Propagation

$$dz^{[l]} = da^{[l]} g^{[l]\prime}(z^{[l]}) = w^{[l+1]T} dz^{[l+1]} g^{[l]\prime}(z^{[l]})$$
$$dw^{[l]} = dz^{[l]} a^{[l-1]T}$$
$$db^{[l]} = dz^{[l]}$$
$$da^{[l-1]} = w^{[l]T} dz^{[l]}$$

Vectorize

$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$
$$dw^{[l]} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$$
$$db^{[l]} = \frac{1}{m} np.sum(dZ^{[l]}, axis=1, keepdims=True)$$
$$dA^{[l-1]} = w^{[l]T} dZ^{[l]}$$



Exploding / Vanishing Gradients

i.e. $g(z) = z$, $b^{[l]} = 0$

$$y = w^{[L]} w^{[L-1]} w^{[L-2]} \cdots w^{[3]} w^{[2]} w^{[1]} X$$

$$z^{[1]} \to z^{[1]}$$

$$z^{[2]} \to z^{[2]}$$

w large → y explode → gradient explode

w small → y vanish → gradient vanish

Solution:

Scale initialization

$$w^{[l]} = np.random.rand(shape) * scale$$

$$scale = \begin{cases} ① \ \sqrt{1/n^{[l-1]}} \\ ② \ \sqrt{2/(n^{[l-1]} + n^{[l]})} \quad \text{Xavier} \end{cases}$$

Gradient Descent with momentum

almost always faster than gradient descent

$$\begin{cases} V_{dw} = \beta V_{dw} + (1-\beta)\, dw \\ V_{db} = \beta V_{db} + (1-\beta)\, db \end{cases}$$

RMSprop

b ↕ slow w ↔ fast

On iteration t:

Compute dw, db on batch

$$S_{dw} = \beta S_{dw} + (1-\beta) dw^2 \quad \leftarrow \text{relative small}$$

$$S_{db} = \beta S_{db} + (1-\beta) db^2 \quad \leftarrow \text{relative large}$$

$$w = w - \alpha \frac{dw}{\sqrt{S_{dw}}}$$

$$b = b - \alpha \frac{db}{\sqrt{S_{db}}}$$

Adam optimization adaptive moment estimation

- Initialization $V_{dw} = 0,\ S_{dw} = 0,\ V_{db} = 0,\ S_{db} = 0$
- On iteration t

momentum:
$$V_{dw} = \beta_1 V_{dw} + (1-\beta_1) dw$$
$$V_{db} = \beta_1 V_{db} + (1-\beta_1) db$$

+ RMSprop:
$$S_{dw} = \beta_2 V_{dw} + (1-\beta_2) dw^2$$
$$S_{db} = \beta_2 S_{db} + (1-\beta_2) db^2$$

$$w := w - \alpha \frac{V_{dw}}{\sqrt{S_{dw}} + \varepsilon}, \qquad b := b - \alpha \frac{V_{db}}{\sqrt{S_{db}} + \varepsilon}$$

Batch Normalization

normalize $z^{[l]}$ or sometime $a^{[l]}$

$$z^{[l]}_{norm} = (z^{[l]} - u) / \sqrt{\sigma^2 + \varepsilon}$$

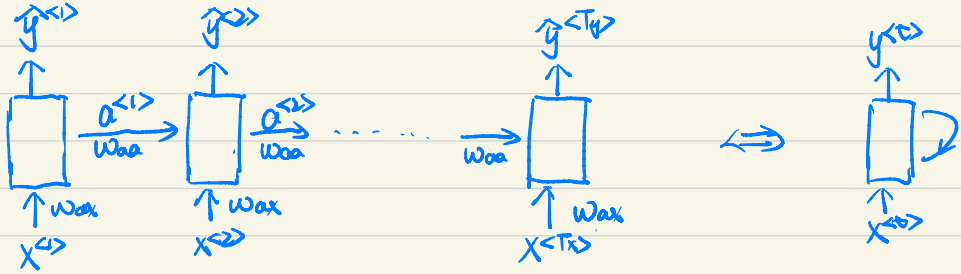
$$\tilde{z}^{[l]} = \gamma z^{[l]}_{norm} + \beta \qquad \gamma = \sqrt{\sigma^2 + \varepsilon},\ \beta = u$$

γ, β: learnable parameter

RL

Reinforcement Learning

Recurrent Neural Network



prediction based on new input $x^{<2>}$ as well as past information $a^{<1>}$

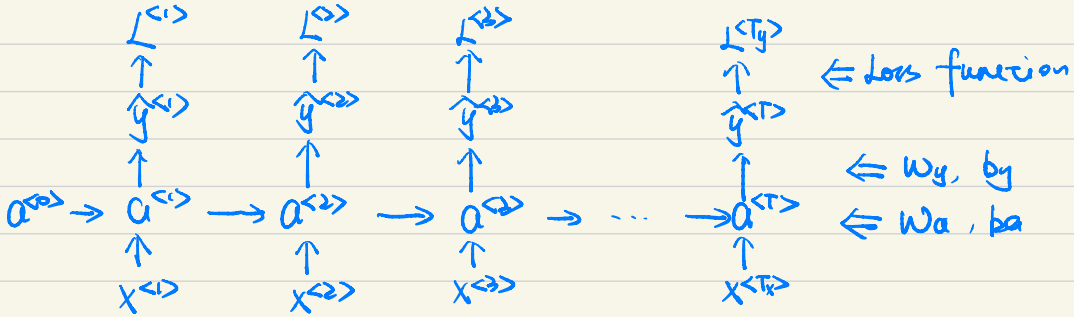
$$\begin{cases} a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a) = g(W_a [a^{<t-1>}, x^{<t>}] + b_a) \\ \hat{y}^{<t>} = g'(W_{ya}\, a^{<t>} + b_y) \\ a^{<0>} = \vec{0} \end{cases}$$

$$W_a = [W_{aa} : W_{ax}]$$

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ \cdots \\ x^{<t>} \end{bmatrix}$$

parameter

Backpropagation

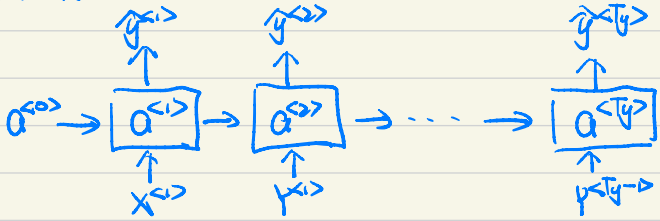


T_x and T_y can be different

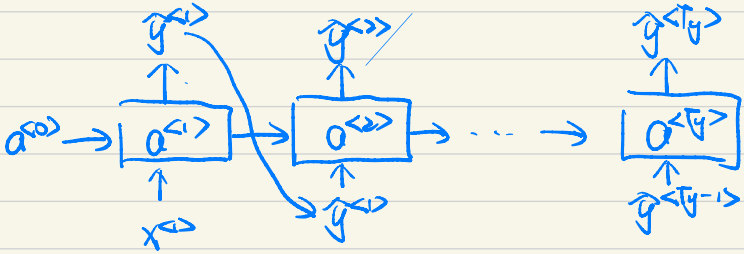
- many to many
- many to one
- one to many
- one to one

Language model

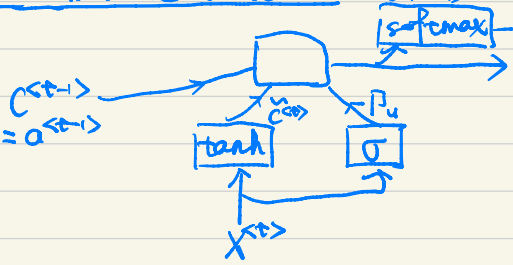
Training:



Sampling:



Gated recurrent unit (GRU) → better at capture long term dependence



GRU

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[\Gamma_r \odot c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$
$$\Gamma_u = \sigma(W_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$
$$\Gamma_r = \sigma(W_r[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_r)$$
$$c^{\langle t \rangle} = \Gamma_u \odot \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) \odot c^{\langle t-1 \rangle}$$
$$a^{\langle t \rangle} = c^{\langle t \rangle}$$

LSTM

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$
$$\Gamma_u = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u) \quad \text{update}$$
$$\Gamma_f = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f) \quad \text{forget}$$
$$\Gamma_o = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o) \quad \text{output}$$
$$c^{\langle t \rangle} = \Gamma_u \odot \tilde{c}^{\langle t \rangle} + \Gamma_f \odot c^{\langle t-1 \rangle}$$
$$a^{\langle t \rangle} = \Gamma_o \odot c^{\langle t \rangle}$$

Deep RNN