Machine Learning, Neural Network, Genetic Programming, Deep Learning, Reinforcement Learning Review

Ron Wu

Last update: 8/6/16

Table of Contents

ML algo comparison:

http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/MLAlgo_Comparisons.pdf

The notes are primarily based on Aarti Singh and Tom Mitchell, ML-701 courses; also Bishop, pattern recognition and machine learning, and Andrew Ng, machine learning notes

Preliminary

1. Bayesian vs Frequentist Probability

Observable data $D = \{t_1, \dots, t_N\}$, model **W**, Bayesian uncertainty of the model, $p(w|D)$, is given by from

$$\overset{\text{posteriori}}{p(\mathbf{w}|\mathcal{D})} = \frac{\overset{\text{likelyhood func}}{p(\mathcal{D}|\mathbf{w})}\,\overset{\text{a priori}}{p(\mathbf{w})}}{\underset{\underset{\int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})\,\mathrm{d}\mathbf{w}}{}}{\underset{\text{normalization}}{p(\mathcal{D})}}}$$

i.e. uncertainty of w is a distribution function.

In frequentist view, w is given by the estimator, hence it is deterministic for given observables. To find uncertainty of w, one would first find uncertainty of $\mathcal{D}$. (see my computational finance note page 20-22 for frequentists regression error and bootstrap/Monte Carlo for other estimation.)

Above equation provides another way to find W; this is W maximizes $p(D|W)$— maximizes likelihood estimate, (*MLE)*. In fact, the negative log of the likelihood function is the error function. Often times likelihood function is made of products and error function is made of sums, so it is easier to differentiate it.

Above Bayes formula gives uncertainty of the model. There is another form of Bayes for prediction models. Let X be data, Y be label, W model

$$p(Y|X;W) \sim p(X|Y;W)p(Y)$$

This is useful for generative/discriminative classifier problem

- ## Discrete Parametric Model

Flip a coin N times get α number of heads. What is the probability of getting head from such coin? Suppose freq $= \theta$, there are $\binom{N}{\alpha}$ possible configuration, and each has the same probability

$$P(D|\theta) = \theta^{\alpha}(1-\theta)^{N-\alpha} \tag{BD-1}$$

Then MLE says $\frac{\partial P(D|\theta)}{\partial\theta} = 0 => \theta_{MLE} = \frac{\alpha}{N}$.

How good is our θ_{MLE}? By Hoeffding inequality (see later section)

For any $\gamma > 0$,

$$P(|\theta_{MLE} - \theta_{true}| > \gamma) \leq 2e^{-2\gamma^2 N}$$

It gives a way to estimate the number of N needed in order to guarantee certain confidence level.

From Bayes formula, it's natural to propose $P(\theta), P(\theta|D)$ have same distribution—same physics. Hence we are looking for conjugate prior. For

$$P(D|\theta) = \theta^{\alpha}(1-\theta)^{N-\alpha}$$

We let $P(\theta) = Bern(\theta; a, b)$, then $P(\theta|D) = Bern(\theta; \alpha + a, N - \alpha + b) \sim \theta^{\alpha+a-1}(1-\theta)^{N-\alpha+b-1}$, then maximizing prior gets

$$\frac{\partial P(\theta|D)}{\partial\theta} = 0 => \theta_{MAP} = \frac{\alpha + a - 1}{N + a + b - 2}$$

So when N, α large, $\theta_{MAP} \sim \theta_{MLE}$. Hence a prior knowledge becomes irrelevant.

- ## Continuous Parametric Model

Assume $P(D|w)$ is product of iid Gaussian $x \sim N(\mu, \sigma)$ and one can show that the σ in Gaussian is indeed the standard deviation by its definition

$$\sigma^2 = E[x^2] - (E[x])^2$$

(for proof e.g. see my intro QM I note pages 6-8). Therefore

$$P(X_i|\mu,\sigma) = \prod \frac{1}{\sqrt{2\pi}\sigma} e^{-(X_i-\mu)^2/2\sigma^2}$$

Then MLE gives

$$\mu_{MLE} = \frac{\sum X_i}{N}, \quad \sigma_{MLE}^2 = \frac{\sum(X_i - \mu_{MLE})^2}{N}$$

Notice the sample variance in above form is biased, because

$$E[\sigma_{MLE}^2] = \frac{\sum_i(E[X_i^2]) - 2E[\sum(X_i\mu)] + \sum_i E[\mu^2]}{N} = \frac{\sum(E[X_i^2] - E[\mu^2])}{N} = \frac{\sum(\mu^2 + \sigma^2 - \mu^2 - \sigma^2/N)}{N} = \frac{N-1}{N}\sigma^2$$

Above we used these facts repeatedly,

$$E[X^2] = E^2[X] + var[X]$$

And

$$var[\mu] = \sigma^2/N$$

Therefore the unbiased σ is

$$\sigma^2 = \frac{\sum(X_i - \mu_{MLE})^2}{N-1}$$

This is the general idea behind linear regression of (x_n, t_n) of N data points, using $y_n = \sum^M w_k x_n^k$, M-polynomial (M+1 coefficients), find $\{w_n\}$ to minimize the error function

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2$$

Root mean square

$$E_{RMS} = \sqrt{\frac{2E(w)}{N}}$$

Modified error function, taking into account of overfitting

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}|w|^2$$

Where $|w|^2 = w_1^2 + \cdots + w_M^2$, called Ridge Regression. How to pick λ? Cross-validation.

Assuming data t_n fits the models w_n, which are normal with variation, $1/\alpha$, pluses extra noise iid with variation $1/\beta$,

$$p(t|x, w, \beta) = \prod_{n=1}^{N} N(t_n|y(x_{n,}, w), 1/\beta)$$

$$p(w|\alpha) = N(w|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} e^{-\frac{\alpha}{2}w^T w}$$

If we maximize—(*MAP*)

$$p(w|x, t, \alpha, \beta) \propto p(t|x, w, \beta)p(w|\alpha)$$

or minimize (ignore terms not involving w)

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_N\}^2 + \frac{\alpha}{2}w^T w$$

which is the modified error function above. If use Laplace prior $p(w) = Laplace(0, t) \sim \prod e^{-\frac{|w_i|}{t}}$, the regulator becomes

$$\frac{\alpha}{2}|w|_1$$

call Lasso regression.

These are aiming at overfitting, and making larger weight unfavorable. The general scheme is called regulation

$$\ln p(D|w_{ML}) - M$$

M: number of adjustable parameters in the model. Common M are AIC and BIC

$$AIC = \#parameters\ in\ model - 2 * \log(L)$$

$$BIC = \#parameters\ in\ model * \log n - 2 * \log(L)$$

n # data points, L maximized value of the likelihood function of the model $p(x|w)$. They penalize complex models.

- Entropy & Information Theory

This leads to the definition of *information*

$$h(x) = -\log_2 p(x)$$

Moreover define

$$H(x) = -\sum_x p(x)\log_2 p(x)$$

Entropy of random variable x. Borrowed from physics: Gibbs entropy is a logarithmic measure of the number of states with probability of being occupied.

Suppose there are N identical particles (bosons) and some energy states. For each energy state *i*, there are n_i particles. Given a set of occupation number $\{n_i\}$, how likely the particles will end up in this macrostate? Equivalently what is the multiplicity of this configuration, i.e. number of microstates in this macrostate?

$$\Omega = \frac{N!}{\prod_i n_i!}$$

Proof: suppose there were distinguishable particles. There were $N!$ ways to line them up along a line. Each line-up fixed one way to put the particles in the microstate, imagining cropping them by the order of n_i. This explains $N!$. Then dividing n_i made them indistinguishable.

Boltzmann's entropy formula $H = k \ln \Omega$. When $\forall n_i = \{1,0\}$, H is maximum and it's the most chaotic.

$$H = \frac{\ln \Omega}{N} \approx \underbrace{\sum \frac{n_i}{N}}_{1} (\ln N - 1) - \sum \frac{n_i}{N} (\ln n_i - 1) = - \sum \frac{n_i}{N} \ln \frac{n_i}{N} = - \sum p_i \ln p_i$$

using Stirling: $\ln N! \approx N \ln N - N$ as $N \to \infty$.

To see H retains its maximum not min, take second derivative

$$\frac{\partial^2 H}{\partial p_i \partial p_j} = - \frac{\delta_{ij}}{p_i} \leq 0$$

To find H max, subject to the constraint, use Lagrange multiplier

$$H = - \sum_i p(x_i) \ln p(x_i) + \lambda \left(\sum_i p(x_i) - 1 \right)$$

Letting above discuss be continuous, we get differential entropy, introduced by Claude Shannon, the father of information theory

$$H[X] = - \int p(X) \ln X \, dX$$

For more of interplay between Entropy, coding, information processing see: Thomas Cover, Joy Thomas, *Elements of Information Theory, Wiley 2006*

2. Probability Distributions

- ### Binary

    - Bernoulli – parametric with one parameter model, $x \in \{0,1\}$

$$\mathrm{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x} = \begin{cases} \mu & x = 1 \\ 1-\mu & x = 0 \end{cases}$$

$$E[x] = \mu, \qquad var[x] = \mu(1-\mu)$$

- Binomial

$$Bin(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m}$$

$$E[m] = \sum_{m=0}^{N} mBin(m|N,\mu) = N\mu$$

$$var[m] = \sum_{m=0}^{N}(m-E[m])^2 Bin(m|N,\mu) = N\mu(1-\mu)$$

- Beta – conjugacy to Bern, b/c setting $p(w) =$ beta → Since $p(Data|W) =$ Bin, by Bayes, $p(w|data) =$ beta, same form of a priori $p(w)$

$$p(w) \sim Beta(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}$$

And

$$E[\mu] = \frac{a}{a+b}, var[\mu] = \frac{ab}{(a+b)^2(a+b+1)}$$

The Gramma functions there to ensure

$$\int_0^1 Beta(\mu|a,b)d\mu = 1$$

So

$$p\left(\mu \middle| \underbrace{m,N}_{\text{data}}, a, b\right) = \frac{\Gamma(a+N+b)}{\Gamma(m+b)\Gamma(N-m+b)}\mu^{m+a-1}(1-\mu)^{N-m+b-1}$$

- Multinomial

$$\boldsymbol{x} \coloneqq K - \text{vector}, \{x_k\}_1^K \in \{0,1\} \text{ and } \sum x_k = 1$$

Let $\mu_k = p(x_k = 1)$, then $\sum \mu_k = 1$, b/c $x_k = 1$ are mutually exclusive and there is one and only one equal to one. The distribution of x

$$p(x|\mu) = \prod_{k=1}^{K} \mu_k^{x_k}$$

Consider a data set D of N independent observations $x_1, \dots, x_N$

$$p(D|\mu) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mu_k^{x_{n,k}} = \prod_{n=1}^{N} \mu_k^{\overbrace{\sum_n x_{n,k}}^{m_k}}$$

where m_k is the sum of the value of the kth component of the N data points.

Then the joint distribution of the m_k's is multinomial

$$p(w) \sim Mult(m_1, .., m_K|\mu, N) = \frac{N!}{m_1!\, m_2! \dots m_K!} \prod_{k=1}^{K} \mu_k^{m_k}$$

- Dirichlet

We show Dirichlet distribution

$$p(D|w) \sim Dir(\mu|\alpha) = \frac{\Gamma(\sum \alpha_k)}{\Gamma(\alpha_1) \dots \Gamma(\alpha_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1}$$

is the conjugate of multinomial, that is

$$p(\mu|D, \alpha) \propto p(D|\mu) p(\mu|\alpha) \propto \prod_{k=1}^{K} \mu_k^{\alpha_k + m_k - 1} = Dir(\mu|\alpha + m)$$

$$= \frac{\Gamma(\alpha_0 + N)}{\Gamma(\alpha_1 + m_1) \dots \Gamma(\alpha_K + m_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k + m_k - 1}$$

- Gaussian
  - Normal

$$N(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

multivariate normal of D dimensional with, Σ, covariance and means, μ , is

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)} \qquad \text{(Gauss.1)}$$

If they are iid, Σ becomes $I(\sigma^2)$

$$p(x|\mu, \sigma^2) = \prod_{n=1}^{D} N(x_n|\mu, \sigma^2)$$

And error function

$$\ln p(x|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln 2\pi$$

Define $\Delta^2 = (x-\mu)^T\Sigma^{-1}(x-\mu)$, this gives the contour line, i.e. constant Gaussian on "distance" centered to μ.

Since $\sum$ is self-adjoint and positive-definite,

$$\Sigma = UVU^T$$

That is one can introduce new coordinate **y** in which the exponential of (Gauss.1) looks like $\mathbf{y^Ty}$. This allows us to reduce the number parameters in the model (via Σ), and treat it like diagonal.

Furthermore, one can show

$$E[x] = \mu, E[xx^T] = \mu\mu^T + \Sigma, cov[x] = \Sigma$$

- Conditional Gaussian & Marginal Gaussian

Split data set $\boldsymbol{x} = \{\boldsymbol{x_a}, \boldsymbol{x_b}\}$, and

$$\mu = \begin{pmatrix}\mu_a\\ \mu_b\end{pmatrix}, \Sigma = \begin{pmatrix}\Sigma_{aa} & \Sigma_{ab}\\ \Sigma_{ba} & \Sigma_{bb}\end{pmatrix}$$

then $p(x_a|x_b)$ is conditional Gaussian with

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b-\mu_b)$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \qquad \text{(CG-MG)}$$

And $p(x_a)$ is marginal Gaussian

$$p(x_a) = N(x_a|\mu_a, \Sigma_{aa})$$

Reverse given $\mu_{a|b}, \Sigma_{a|b}, \mu_a, \Sigma_{aa},$ one can solve for $\mu_{b|a}, \Sigma_{b|a}, \mu_b, \Sigma_{bb}$ and thus get $p(x_b|x_a)$ and $p(x_b)$.

- Gamma Distribution

The conjugacy of normal is a Gamma distribution

$$p(w) \sim Gam(\lambda|a,b) = \frac{1}{\Gamma(a)}b^a\lambda^{a-1}e^{-b\lambda}$$

with

$$E[\lambda] = \frac{a}{b}, \qquad var[\lambda] = \frac{a}{b^2}$$

Because

$$p(\lambda|X) \propto \lambda^{a_0-1}\lambda^{\frac{N}{2}}e^{-b_0\lambda-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2}$$

- Wishart

Consider N Gaussian data – multivariate normal

$$p(X|\lambda) = \prod_{n=1}^{N} N(x_n|\mu,\lambda^{-1}) \propto \lambda^{\frac{N}{2}}e^{-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2}$$

The conjugate prior on mean is normal.

The conjugate prior on covariance is inverse Wishart.

See Murphy https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf

- Student T

If we integrate above over λ, creating Gaussian Mixture and increasing robustness, we get student T distribution

$$St(x|\mu,\Lambda,\nu) = \int_0^\infty N(x|\mu,(\eta\Lambda)^{-1})Gam\left(\eta\middle|\frac{\nu}{2},\frac{\nu}{2}\right)d\eta = \frac{\Gamma(D/2+\nu/2)}{\Gamma(\nu/2)}\frac{|\Lambda|^{1/2}}{(\pi\nu)^{D/2}}\left[1+\frac{\Delta^2}{\nu}\right]^{-\frac{D}{2}-\frac{\nu}{2}}$$

With

$$E[x] = \mu, cov[x] = \frac{\nu}{\nu-2}\Lambda^{-1}$$

- Circular Gaussian

Consider

$$p(x_1,x_2) = \frac{1}{2\pi\sigma^2}e^{-\frac{(x_1-\mu_1)^2+(x_2-\mu_2)^2}{2\sigma^2}}$$

Substitute $= r\cos\theta$, $x_2 = r\sin\theta$, $\mu_1 = r_0\cos\theta_0$, $\mu_2 = r_0\sin\theta_0$ We get circular Gaussian

$$p(\theta|\theta_0,m) \sim e^{m\cos(\theta-\theta_0)}$$

its normalization is 0th order Bessel function of the 1st kind.

3. Learning Theory

• Finite Hypotheses

Example of finite H: How many distinct decision trees (Boolean functions) of n Boolean attributes?

$$f(x_1, \dots, x_n) = y \text{ where } x_i, y = boolean$$

It equals to the number of distinct truth tables with 2^n rows = $2^{(2^n)}$. A smarter way is to write each f in logical operator form

$$f = x_1 \wedge \neg x_2 \wedge x_3 \dots$$

Since *b* number of bits can represent 2^b different integers, so hypothesis complexity $|H| = 2^n$. If we allow learning algorithms to have the ability to ignore some attributes, $|H| = 3^n$.

How many distinct decision trees with depth 2 of N Boolean variable?

$$|H| = N(N-1)(N-2)2^4$$

Clearly $|H|\ finite => the\ set\ of\ instances\ (train + test\ data)\ |X|\ finite$

Active Learner: learner to pick what data to train. Training Scenario: trainer pick data to learner.

Here we consider random training set D. Training instances as well as the testing set are drawn from a fixed probability distribution PD, independent of hypothesis.

<u>Consistent trainer</u>: For each training data D, if x=y, the trainer will classify the result c(x)=c(y). The setup of all such hypotheses is called

$$version\ space = \{h \in Hypotheses: Consistent(h, D)\}$$

Learner tries to come up with h hypothesis. h(x) estimates value of x. c(x) correct value of x for both training and testing data.

Training error (h) = $P_{x\in D}\big(c(x) \neq h(x)\big) = \frac{\sum_x \delta(c(x)\neq h(x))}{|D|}$

True error (h) = $P_{x\in PD}(c(x) \neq h(x))$

Because variance of the binomial distribution is p(1-p), 95% confidence intervals of True error (h) lies in

$$error_{train}(\mathrm{h}) \pm 1.96\sqrt{\frac{[1 - error_{train}(\mathrm{h})]error_{train}(\mathrm{h})}{|D|}}$$

• Haussler Theorem

A version space of H & D is said to be <u>ϵ-exhausted</u> iff

$$error_{true}(h) < \epsilon \;\; \forall h \in VS_{H,D}$$

Haussler Theorem: $|H|$ finite, D is sequence of random instances with classification c, for any $\epsilon \in [0,1]$, the probability that the version space wrt H, D is not ϵ-exhausted is less than

$$|H|e^{-\epsilon|\{D\}|}$$

Proof: Let $h_1, \dots, h_k \in H$ be the ones whose error $\geq \epsilon$. The probability that the version space wrt H, D is not ϵ-exhausted is equal to the probability of at least one of $h_1, \dots, h_k$ sneaks into the version space, i.e. it makes correct estimate of all training data, which is

$$\leq \binom{k}{1}(1-\epsilon)^{|\{D\}|} \leq |H|e^{-\epsilon|\{D\}|}$$

because $(1-\epsilon) \leq e^{-\epsilon}$. QED.

We have shown

$$\underbrace{P(\exists h \in H \ni error_{train}(h) = 0 \;\wedge error_{true}(h) \geq \epsilon)}_{=\delta} \leq |H|e^{-\epsilon|\{D\}|} \qquad \text{(HT-1)}$$

Let LHS = $\delta = \text{failure probability} < 1$, type-1 error.

$$|\{D\}| \geq \frac{\ln|H| - \ln\delta}{\epsilon} \qquad \text{(HT-2)}$$

This also implies

$$error_{true}(h) \geq \epsilon \geq \frac{\ln|H| - \ln\delta}{|\{D\}|}$$

In other words, for $h \ni error_{train}(h) = 0$, with probability at least $1-\delta$,

$$error_{true}(h) \leq \frac{\ln|H| - \ln\delta}{|\{D\}|}$$

We are interested in PTIME algorithm (CPU complexity), i.e. probably approximately correct (PAC) learning

Let C be a class of possible classifications defined over a set of instances ($|X| = n$), and hypotheses space H. C is PAC-learnable iff

For all $c \in C$, distribution PD over X, with $\epsilon, \delta \in (0,1/2)$. A learner will with probability at least $1-\delta$ output a $h \in H$ such that $error_{true}(h) \leq \epsilon$ in time that is polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, n$ and $|c|$.

- ### Hoeffding's Inequality

Above discussion focuses on cases in which $h \ni error_{train}(h) = 0$. What if that is not achievable we now consider $error_{train}(h) \neq 0$. We need Hoeffding inequality

Let $X_1, \dots, X_m$ be independent random variables bounded by interval $[a_i, b_i]$, $i = 1, \dots, m$ respectively. Let $\hat{\theta} = \frac{1}{m}\sum X_i$. Then for any $t > 0$,

$$P\left(\left(E\left(\hat{\theta}\right) - \hat{\theta}\right) \geq t\right) \leq e^{-\frac{2mt^2}{\sum(a_i - b_i)^2}}$$

$$P\left(\left|E\left(\hat{\theta}\right) - \hat{\theta}\right| \geq t\right) \leq 2e^{-\frac{2mt^2}{\sum(a_i - b_i)^2}}$$

Proof: We first show Hoeffding Lemma. X random variable taking value in [*a,b*], with $E(X) = 0$, then

$$E(e^{tX}) \leq e^{\frac{t^2(b-a)^2}{8}}$$

Proof of claim: $e^x, E(.)$ are convex

$$e^{tX} \leq \frac{b - X}{b - a}e^{tb} + \frac{X - a}{b - a}e^{ta} => E(e^{tX}) \leq \frac{b}{b - a}e^{tb} - \frac{a}{b - a}e^{ta} = e^{\phi(u)}$$

where $u = t(b - a)$, $\phi(u) = -\lambda u + \log(1 - \lambda + \lambda e^u)$, $\lambda = -\frac{a}{b-a}$.

$$\phi(u) = \underbrace{\phi(0)}_{0} + \underbrace{\phi'(0)}_{0}u + \underbrace{\phi''(v)}_{\leq\frac{1}{4}}\frac{u^2}{2}, \qquad for\ some\ v \in t(a, b)$$

Proved the claim. By Markov's inequality: if X is non-negative rv, for any $t > 0$

$$E[X] = \int_0^\infty XP(X)dX \geq \int_t^\infty XP(X)dX \geq t\int_t^\infty P(X)dX = tP(X \geq t)$$

By the way the corollary of this is the Chebyshev's inequality

$$P(|X - E[X]| \geq k) = P((X - E[X])^2 \geq k^2) \leq \frac{E\left[\left(X - E(X)\right)^2\right]}{k^2} = \frac{\sigma^2}{k^2}$$

Back to Hoeffding inequality, for $s > 0$. It satisfies assumptions of Markov and Hoeffding lemma, because assuming $E\left[\hat{\theta}\right] \geq \hat{\theta}$ (otherwise $P\left(\left(E\left[\hat{\theta}\right] - \hat{\theta}\right) \geq t\right) = 0$, the inequality is satisfied automatically) and $E[X_i] - X_i$ is in the range of $[0, (b_i - a_i)]$ or $[-(b_i - a_i), 0]$ thus

$$P\left(\left(E\left[\hat{\theta}\right]-\hat{\theta}\right)\geq t\right)=P\left(e^{s\left(E\left[\hat{\theta}\right]-\hat{\theta}\right)}\geq e^{st}\right)\leq\frac{E\left[e^{s\left(E\left[\hat{\theta}\right]-\hat{\theta}\right)}\right]}{e^{st}}=e^{-st}\prod E\left[e^{\frac{s}{m}(E[X_i]-X_i)}\right]$$

$$\leq\ e^{-st}\prod e^{\frac{s^2(b_i-a_i-0)^2}{8m^2}}=e^{-st+\sum\frac{s^2(b_i-a_i)^2}{8m^2}}$$

Taking max over s, we get

$$P\left(\left(E\left[\hat{\theta}\right]-\hat{\theta}\right)\geq t\right)\leq e^{-\frac{2m^2t^2}{\sum(b_i-a_i)}}$$

QED.

- ## Agnostic Learning

Now we are ready to generalize (HT-1). Apply Hoeffding inequality, let X_i be 1 when the learner get correct estimate and 0 when it gets wrong, m the number of training sample. Hence $b_i-a_i=1$, so for any $h\in H$

$$P(error_{true}(h)-error_{train}(h)\geq t)\leq e^{-2mt^2}$$

For all $h\in H$

$$\underbrace{P(\exists h\in H\ni error_{true}(h)-error_{train}(h)\geq t)}_{=\delta}\leq|H|e^{-2mt^2}$$

Similar as before with probability at least $1-\delta$, all h

$$error_{true}(h)\leq error_{train}(h)+\sqrt{\frac{\ln|H|-\ln\delta}{2m}}\qquad\text{(AL-1)}$$

- ## Infinite Hypotheses

What happens when $|H|=\infty$? We use Vapnik–Chervonenkis (VC) dimension to characterize $|H|$. In words,   dimension of H is the size of the largest subset of X for which H can guarantee zero training error, regardless of what the trainer classification function c is.

Formally, a dichotomy of a set $S\subset X$ is a partition of S into two disjoint subsets. A set of instance S is shattered by H iff for every dichotomy of S there exists some $h\in H$ consistent with this dichotomy. The VC dimension of H over X is the size of the largest finite sets of X can be shattered by H. If arbitrary large finite sets of H can be shattered by H, then $VC(H)=\infty$.

Example: SVM, VC dimension of lines in a plane R^2? VC=3. 3 non-collinear points are always binary separable, 1/0 loss function. By induction, one can show VC(linear separating hyperplanes in n dim) = n + 1.

Reversely given $VC(H)$, we know there are $VC(H)$ points that can be separated by the hypotheses, so the hypotheses space is at least

$$|H| \geq 2^{VC(H)}$$

Vapnik Theorem:

$$(\mathrm{HT}-2) \text{ changes to } m \geq \frac{1}{\epsilon}\left(4\log\frac{2}{\delta} + 8VC(H)\log\frac{13}{\epsilon}\right)$$

$$(\mathrm{AL}-1) \text{ changes to } \ error_{true} \geq error_{train} + \sqrt{\frac{VC(H)\left[\log\frac{2m}{VC(H)}+1\right]+\log\frac{4}{\delta}}{m}}$$

- Mistake Bound

Question: how many mistakes can the learner make before the learner finds the best hypothesis? Consider halving algorithm for binary classifications:

-For a given data point, output the estimate value by each hypothesis in the hypotheses space.

-Kick out hypotheses that misclassify instance from the hypotheses space, and repeat. A mistake is recorded when majority hypotheses (If tie, pick them at random) misclassify the instance.

The worst case, maximum number of mistakes, occurs when every decision turns out to be wrong and only the smallest number of hypotheses being eliminated, i.e. $\frac{|H|}{2}$. It can be achieved when H is the power set of X. After kth mistake,

$$|H| \ becomes \frac{|H_{initial}|}{2^k}$$

Hence the maximum mistake is

$$\log_2|H_{initial}|$$

Let $M_A(h)$ be the maximum number of mistake made by algorithm A for some hypothesis h over all possible training sequence (worst case). The optimal mistake bound is

$$opt(H) = \min_{\substack{A\in learning\\ algorithms}} \max_{h\in H} M_A(h)$$

Optimal Mistake Bound Theorem:

$$VC(H) \leq opt(H) \leq M_{halving}(H) \leq \log_2|H|$$

The first inequality on the left is due to the fact that regardless of what algorithm is in use, there is a set $S \ni |S| = VC(H)$ and each point in S is distinguishable from the rest, hence there is h that contains at least $|S|$ number of classifying functions that can misclassify them.

Supervised Learning—Parametric

4. Discriminative: Generalized Linear Models: Least Mean Squares (LMS), logistic regression

$$p(t|x, w, \beta) = \prod_{n=1}^{N} N(t_n|w^T\phi(x_n), 1/\beta)$$

W model, ϕ base function. When $\phi(x) = x$, we have linear regression; when

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

Logistic regression. Negative log gives error function

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - w^T\phi(x_n)\right)^2$$

And minimizing it gives

$$w_{ML}(\phi^T\phi)^{-1}\phi^T t$$

If above batch calculation becomes computationally intense, use stochastic gradient descent

$$w^{(\tau+1)} = w^{(\tau)} - \eta\nabla E_n$$

$$w^{(\tau+1)} = w^{(\tau)} + \eta\left(t_n - w^{(\tau)T}\phi_n\right)\phi_n$$

Choose η, learning rate, so that it converges. Decay learning rate or use momentum.

- Regression error

Bias-variance tradeoff: increase variance in observation will reduce bias in determining the slope.

x features, D data distribution, t model, y label, E[t|X] estimates

$$E_D[(y(x,D) - E[t|x])^2] = \underbrace{(E_D[y(x,D)] - E[t|x])^2}_{(bias)^2} + \underbrace{E_D[(y(x,D) - E_D[y(x,D)])^2]}_{variance}$$

- ### Exponential Family Distribution

$$p(y;\eta) = b(y)\exp\left(\eta^T T(y) - a(\eta)\right)$$

y label, x feature/attributes-measurement , $T(y)$ prediction, η exponential family parameter, θ model. Let $h(x) = E[y|x]$ be learned hypothesis. Linear model says model coefficients are linear

$$\eta = \theta^T x$$

- Linear regression

$p(y;\mu,\sigma) \sim \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right) = \exp\left(-\frac{1}{2\sigma^2}y^2\right)\exp\left(\underbrace{(\mu)}_{\eta}y - \frac{1}{2}\mu^2\right)$, hence

$$h = E[y|x] = \mu = \eta = \theta^T x$$

Thus log likelihood

$$l(\theta) = \log \prod p(y|x;\theta) \sim -\frac{1}{2}\sum (y-\theta^T x)^2 \sim \sum (h-y)^2$$

Algorithm: steepest decent

For data $i = 1, \ldots, m$, $x \in R^n$, repeat until convergence , for every j

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

The traditional route to derive linear regression starting from

$$y = \theta^T x \ \ subject\ to \min \sum (y - \theta^T x)^2$$

We can also propose non-linear functions, such as Fourier, wavelets,… other orthogonal

The problem is exact solvable. Let $X = cbind\{(1,1,\ldots,1)^T, (X_1,\ldots,X_n)^T\}$ be $n \times (D+1)$ matrix. Its columns are n data points of dimension D. Let Y be label.

$$X = \begin{pmatrix} 1 & X_1 \\ 1 & X_2 \\ \ldots & \ldots \\ 1 & X_n \end{pmatrix}$$

Our goal is to learn W, vector of $(D+1)$ dimension, such that

$$\min_W (\|Y - XW\|^2 + \lambda \|W\|^2)\,, \qquad \lambda \geq 0$$

Solution

$$W = (X^T X + \lambda I)^{-1} X^T Y$$

Projection of test data x

$$\hat{x} = \langle x, W \rangle$$

W is in fact in the row space of X, since

$$(X^T X + \lambda I)W = X^T Y => W = X^T \underbrace{\frac{1}{\lambda}(Y - XW)}_{\alpha}$$

The dual problem of linear regression:

Let $W = \sum_i^n \alpha_i \hat{X}_i$, $\hat{X}_i = i\text{th row of } X$. Then

$$\alpha = \frac{1}{\lambda}(Y - X \underbrace{W}_{X^T \alpha}) = (XX^T + \lambda I)^{-1} Y$$

What is XX^T?

$$(XX^T)_{ij} = \underbrace{\langle \hat{X}_i, \hat{X}_j \rangle}_{K(\hat{X}_i, \hat{X}_j)}$$

By the dual solution

Projection of test data x

$$\hat{x} = \langle x, W \rangle = \sum_{i=1}^{n} \alpha_i \underbrace{\langle x, \hat{X}_i \rangle}_{K(x, \hat{X}_i)}$$

Thus the computation of projection involves only inner products (kernel) of training data.

Kernel Trick

If one wants to do quadratic regression, x_1, x_2 are old features,

$$y \sim x_1^2 + x_2^2 + \sqrt{2} x_1 x_2$$

One doesn't have to transform $\Phi(x_1, x_2) \to (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$ then apply ordinal linear regression.

Instead use the dual solution, since the dimension of α is independent of feature space.

Replace $K(u, v) = \langle u, v \rangle$ by

$$K(u, v) = \langle u, v \rangle^2$$

because

$$\langle \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} {x'}_1 \\ {x'}_2 \end{pmatrix} \rangle^2 = \langle \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}, \begin{pmatrix} {x'}_1^2 \\ {x'}_2^2 \\ \sqrt{2}{x'}_1{x'}_2 \end{pmatrix} \rangle$$

Other common kernels include

$$K(u,v) = (\langle u,v \rangle + 1)^d$$

$$K(u,v) = e^{-\frac{\|u-v\|^2}{2\sigma^2}}$$

$$K(u,v) = \tanh(\mu\langle u,v \rangle + \nu)$$

All of them allow to solve non-linear regression in linear complexity.

Logistic Regression

In linear regression y continuous, here $y \in \{0,1\}$. $P(Y = 1|X) = \phi$

$$p(y;\phi) = \phi^y(1-\phi)^{1-y} = \exp\left(\underbrace{\log\left(\frac{\phi}{1-\phi}\right)}_{\eta} y + \log(1-\phi) \right)$$, hence

$$h = E[y|x] = 1 * \phi + 0(1-\phi) = \phi = \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}$$

our goal is to maximize

$$l(\theta) = \log \prod p(y|x;\theta) \sim \sum y \log h + (1-y)\log(1-h)$$

It turns out to be the same algorithm, because steepest decent

$$\frac{\partial l}{\partial \theta} = (y-h)x = (y-\phi)x$$

Repeat until convergence, for every j

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

We now show the boundary is linear. Suppose X is 2-dim, we show in the (x_1, x_2) space the boundary surface is a line.

$$p(Y = 1|x_1, x_2) = \phi = \frac{1}{1+e^{w_0+w_1x_1+w_2x_2}}$$

so

$$\frac{p(Y=1|x_1,x_2)}{p(Y=0|x_1,x_2)} = e^{w_0+w_1x_1+w_2x_2} \qquad \text{(LG-B)}$$

which can be > or < 1. At the boundary, it is 1, that is

$$w_0 + w_1x_1 + w_2x_2 = 0$$

is the boundary.

To avoid overfitting, use regulation, assume $p(w){\sim}N(0,\kappa)$, then use MAP

$$p(w|Y,X){\sim}p(Y|X,w)p(w) => l(w) = \log\prod p(y|x;w) - \sum\frac{w_i^2}{2\kappa^2}$$

which has unfavorable view toward larger w_i's.

One can change to non-linear boundary by using kernel, $w = \sum_i \alpha_i\phi(x_i)$

$$P(Y=1|x,w) = \frac{1}{1+e^{-(w^T\phi(x)+b)}} = \frac{1}{1+e^{-(\sum\alpha_iK(x_i,x)+b)}}$$

- Poisson Distribution

$p(y;\phi) = \frac{e^{-\lambda}\lambda^y}{y!} = \exp(y\log\lambda - \lambda - \log y!)$, hence

$$h = E[y|x] = \lambda = e^\eta = e^{\theta^Tx}$$

It turns out to be the same algorithm, because $l{\sim}y\log h - h$ and steepest decent

$$\frac{\partial l}{\partial\theta} = (y-h)x$$

Repeat until convergence for every *j*

$$\theta_j = \theta_j + \alpha\sum_{i=1}^m\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

- Softmax Regression

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

$$p(y;\phi) = \prod_i^k \phi_i^{I_{y=i}} = \exp\left(\log\left(\frac{\phi_1}{\phi_k}\right)T(y)_1 + \cdots + \log\left(\frac{\phi_{k-1}}{\phi_k}\right)T(y)_{k-1} + \log(\phi_k)\right)$$

$T(y)_i = \delta_{ji}$, hence

$$\eta_i = \log\frac{\phi_i}{\phi_k} = \theta_i^T x \to \phi_i = \frac{e^{\theta_i^T x}}{\sum_j e^{\theta_j^T x}}$$

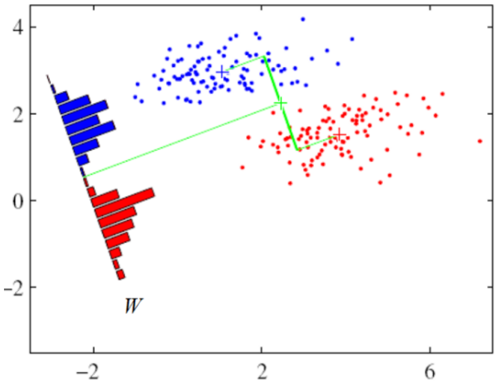
Algorithm: maximize

$$l(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)},\theta) = \sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1_{\{y^{(i)}=l\}}}$$

Example digit recognition using TensorFlow with 92.44% accuracy

Courtesy Yijun Xiao https://nbviewer.ipython.org/github/yjxiao/tensorflow-basics/blob/master/tf_tutorial.ipynb

- Fisher Linear Discriminant Analysis (LDA)



Picture from http://web.media.mit.edu/~javierhr/files/slidesPCA.pdf page 21

The goal is to project data to lower dimension while preserves their classifications.

Let $C_{1,2} = class$, $x =$ data with some dimensional, so m_i is a vector,

$$m_i = \frac{1}{N_i}\sum_{x\in C_i} x\,, i = 1,2$$

$$S_B = (m_1 - m_2)(m_1 - m_2)^T, S_W = \sum_{j=1}^{2}\sum_{x\in C_j}(x - m_j)(x - m_j)^T$$

Then our task is to find the projection direction W that

$$\arg\max_{W} \underbrace{\frac{W^T S_B W}{W^T S_W W}}_{J} = \frac{\text{maximize the distance between two projected class means}}{\text{minimize sum of distances of each date to its class mean (variance)}}$$

Derivative wrt W, get

$$S_W^{-1} S_B W = JW$$

Then

$$W = S_W^{-1}(m_1 - m_2)$$

Predict test data

$$\arg\max_{j} W^T(x - m_j)$$

For more than 2-class, see http://www.facweb.iitkgp.ernet.in/~sudeshna/courses/ML06/lda.pdf page 8

5. Generative: Gaussian discriminant Analysis, Naïve Bayes

• Bayes Optimal Classifier

X: data; Y: label

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

discriminative: map inputs directly to decisions, i.e. learn $p(y|x)$

Generative: learn $p(x|y)$ and $p(y)$ then use Bayes. Hence we want to maximize

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

Key: find decision boundary.

• Naïve Bayes Assumption

X is conditional independent of Y given Z iff

$$P(X = x|Y = y, Z = z) = P(X = x|Z = z), \forall x, y, z$$

Or equivalently

$$P(X, Y|Z) = P(X|Z)P(Y|Z)$$

Let $\{x_1, \dots, x_d\}$ be vocabulary, y ={spam, not-spam}, Naïve Bayes says

$$p(x_1, \dots, x_d|y) = \prod p(x_i|y)$$

- ### Naive Bayes Algorithm

Assume the likelihood is Bernoulli, use train data to get $p(x_i|y) =$ freq occured for each word for both spam and nonspam (why? See MLE Bernoulli distribution (BD-1)).

Then find y for test data to get

$$\max_y \prod p(x_i|y)$$

Remember to apply Laplace smoothing because for small train set there are certain words, either $p(x|spam) = 0$ or $p(x|nonspam) = 0$, then maximizing $\prod p(x_i|y)$ becomes absurd. Add Laplace smoothing is equivalent to apply conjugate prior and doing MAP.

Now show Naïve Bayes boundary is linear. $X =< X_1, \dots, X_n >$.

$$\frac{p(y=1|X)}{p(y=0|X)} = \frac{p(X|y=1)p(y=1)}{p(X|y=0)p(y=0)} = 1 => \log\frac{p(y=1)}{p(y=0)} + \log\frac{p(X|y=1)}{p(X|y=0)} = 0$$

Let $P(X_i|Y=1) = \theta_{i1}, X_i = \{0,1\}$

$$\log\frac{p(X|y=1)}{p(X|y=0)} = \sum \log\frac{p(X_i|y=1)}{p(X_i|y=0)} = \sum_i X_i \log\frac{\theta_{i1}}{\theta_{i0}} + (1-X_i)\log\frac{1-\theta_{i1}}{1-\theta_{i0}}$$

- ### Gaussian Naïve Bayes (Gaussian discriminant Analysis)

If the feature is continuous, say digit detection. x_i are pixels, $y = \{1, \dots, 9\}$. Assume Gaussian likelihood and assume Naïve Bayes assumption, then compute

1) $\mu_i =$ average pixels over training data

2) $\sigma_i^2 =$ variation pixels value over training data

for each position i, and each label y. Then determine

$$\max_y \prod p(x_i|y)$$

for the testing data.

Now show Gaussian Naïve Bayes boundary.

$$\frac{p(y=1|X)}{p(y=0|X)} = \frac{p(X|y=1)p(y=1)}{p(X|y=0)p(y=0)} = 1 => \log\frac{p(y=1)}{p(y=0)} + \log\frac{p(X|y=1)}{p(X|y=0)} = 0$$

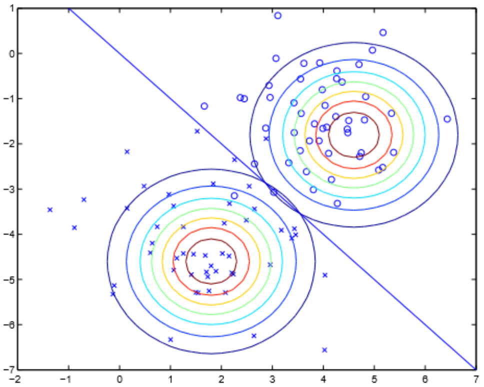
If we want to make the boundary linear, we have to have

$$y \sim Bernoulli(\phi)$$

$$x|y=0 \sim N(\mu_0, \Sigma), \qquad x|y=1 \sim N(\mu_1, \Sigma)$$

share the same covariance matrix (class-independent of y) to kill 2nd order x_i, then

$$\log\frac{p(y=1)}{p(y=0)} + \log\frac{p(X|y=1)}{p(X|y=0)} = \underbrace{\log\frac{1-\phi}{\phi} + \sum\frac{\mu_{i,y=1}^2-\mu_{i,y=0}^2}{2\sigma_i^2}}_{const} - \sum\underbrace{\frac{\mu_{i,y=1}-\mu_{i,y=0}}{2\sigma_i^2}}_{const} x_i = 0 \qquad \text{(NB-G)}$$



Ng, note: Generative Algorithms

In comparison to logistic regression: (NB-G) remains us of the similar formula for logistic regression

$$P(Y=0|X) = \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)} = \frac{1}{1+\frac{P(Y=1)P(X|Y=1)}{P(Y=0)P(X|Y=0)}}$$

$$= \frac{1}{1+e^{\log\frac{p(y=1)}{p(y=0)}+\log\frac{p(X|y=1)}{p(X|y=0)}}}$$

(NB-G) shows the last line above is

$$P(Y=0|X) = \frac{1}{1+e^{w_0+\sum w_i x_i}} \qquad \text{(LR-NB)}$$

Hence if we have Naive Bayes assumption, Gaussian distribution, and class-independent σ_i, Gaussian Naïve Bayes is reduced to logistic regression provided we have infinite training data. If NB assumption doesn't hold, logistic regression is better. If the linear decision surface assumption doesn't hold, then Gaussian NB with class-dependent σ_{iy} is better.

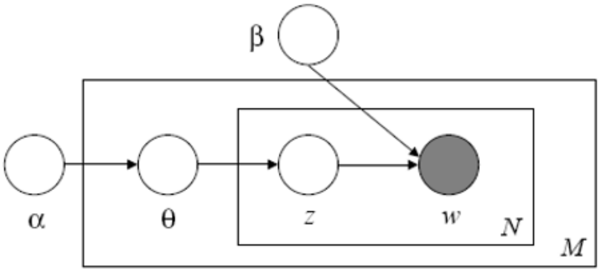
Asymptotic behavior, given finite training data, n, d features $X =< X_1, \dots, X_d >$

$$\epsilon_{LR,n} \le \epsilon_{LR,\infty} + \sqrt{\frac{d}{n}}$$

$$\epsilon_{GNB,n} \le \epsilon_{GNB,\infty} + \sqrt{\frac{\log d}{n}}$$

- Latent Dirichlet Allocation

Recall Dirichlet distribution θ is conjugate prior of multinomial. LDA groups words w into set of N topics z. z are the latent features. Hence this borrows idea from NB but it is an unsupervised learning.



David Blei, Andrew Ng, Michael Jordan, http://jmlr.org/papers/volume3/blei03a/blei03a.pdf page 5

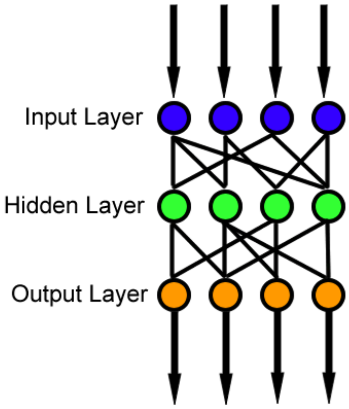
Following the diagram, the probability of the corpus given models α, β of M documents is

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N} \sum_{z_{d_n}} p(z_{d_n}|\theta_d) p(w_{d_n}|z_{d_n},\beta) \right) d\theta_d$$

where $\theta \sim$ Dirichlet, $z, w \sim$ Multinomial. If N is unknown a priori and is to be inferred from the data, see extension of LDA, http://people.eecs.berkeley.edu/~jordan/papers/hdp.pdf

6. Discriminative: Artificial Neural Network

- Forward Propagation for Prediction



Picture https://en.wikipedia.org/wiki/Feedforward_neural_network

Each input and output node represents one feature of input or output data. Following the picture suppose we have 1-hidden layer with 4 nodes (logistic neuron), and recall sigmoid function

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

For each one of the 4 outputs, $j = 1, \dots, 4$

$$O_j(X_1, \dots, X_4) = \sigma(w_{0j} + \sum_h w_{hj}\sigma(w_{0j}^h + \sum_i w_{ij}^h X_i))$$

Compare to (LG-B), the boundary here is definitively not linear. Forward propagation means follow the arrow to make prediction.

- Backpropagation for Training

Find the values of $W = \{w_{ij}^h, w_{hj}, \dots\}$ of each edge. Use MLE

$$\arg\min_W \sum_{train} (Y - O)^2$$

Or MAP

$$\arg\min_W \left[\sum_{train} (Y - O)^2 + c \sum_{w \in W} |w|^2\right]$$

Backpropagation Algorithm (MLE)

- initialize all weights to small random numbers, use forward propagation compute hidden layer O_h and output layer values, then use gradient decent to update weights

Use either Batch gradient decent (also called Vanilla Gradient Descent): Compute $\nabla E_D[W]$ for all data D,

$$E[W] = \frac{1}{2}\sum_{train}\left(Y^l - O^l\right)^2$$

$$\frac{\partial E}{\partial w_h} = \sum_{train} - \underbrace{(Y^l - O^l)O^l(1 - O^l)}_{\delta^l} O_h^l$$

$$\frac{\partial E}{\partial w_i^h} = \sum_{train} - \underbrace{\underbrace{(Y^l - O^l)O^l(1 - O^l)}_{\delta^l} O_h^l(1 - O_h^l)w_h}_{\delta_h^l} X_i^l$$

update $W \leftarrow W - \eta \nabla E_D[W]$

$$w_h \leftarrow w_h + \eta \delta^l O_h^l$$

$$w_i^h \leftarrow w_h + \eta \delta_h^l X_i^l$$

Or stochastic gradient descent:

loop through each data, compute $\nabla_i E_l[W]$, update $W \leftarrow W - \eta \nabla E_l[W]$, loop again

-repeat

Such algorithm doesn't guarantee stop at global minimum.

- ### Online Perceptron Theorem

Basis for Natural Language Processing. See Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms by Michael Collins
http://www.cs.columbia.edu/~mcollins/papers/tagperc.pdf

or lecture *Machine Learning for Natural Language Processing*,
http://www.cs.columbia.edu/~mcollins/courses/6998-2012/lectures/lec1.3.pdf

Consider the simplest perceptron net

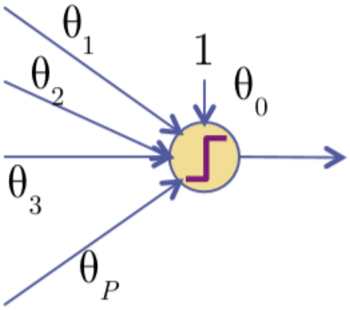
Instead of logistic neuron

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

use

$$\sigma(z) = step(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

and



Tony Jebara http://www.cs.columbia.edu/~jebara/4771/notes/class4x.pdf page 4

We are to minimize

$$\sum_{train}(Y-O)^2 = \sum_{train}\begin{cases} step(-W^TX_i) & Y_i = 1 \\ step(W^TX_i) & Y_i = 0 \end{cases}$$

If we use $Y = \{\pm 1\}$, above becomes

$$\sum_{train} step(Y_iW^TX_i)$$

Since 0-1 loss is not good for differentiation, we decide to use perceptron loss

$$E = \sum_{i \in missclassified} -Y_i(W^TX_i)$$

Apply gradient decent,

Online Perceptron Algorithm

- pick initial small W, loop training data

- if i misclassified

$$w_i = w_i - \eta\nabla_W E = w_i + \eta Y_i X_i$$

- repeat

Under certain conditions the online perceptron algorithm converges to zero error in finite time.

Online Perceptron Theorem:

Let a sequence of training samples $(X_i, Y_i)_{i=1}^m$ be given. Suppose that $\|X_i\| < r$ for all i, and the data with $Y = \pm 1$ are separable with margin γ. total number of mistakes that the perceptron algorithm makes on this sequence is at most $(r/\gamma)^2$

Supervised Learning—Non-parametric

7. KNN

In the next 3 sections we will focus on non-parametric models: the raw data may not exhibit any functional forms.

- ### Closest Mean & Kernels

Classify test data base on the distance to the closest class mean,

$$\arg\min_{i\ni C_i}\left\|x - x_{C_i}\right\|^2 \quad , \quad x_{C_i} = \frac{1}{N_{C_i}}\sum_{j\ni x_j\in C_i}^{N_{C_i}} x_j$$

Since $\left\|x-x_{C_i}\right\|^2 = \|x\|^2 - 2\langle x, x_{C_i}\rangle + \langle x_{C_i}, x_{C_i}\rangle$, hence

$$\arg\min_{i\ni C_i}\left(-\frac{2}{N_{C_i}}\sum \underbrace{\langle x, x_{C_i}\rangle}_{K(x,x_{C_i})} + \frac{1}{N_{C_i}^2}\sum_{ij}\underbrace{\langle x_i,x_j\rangle}_{K(x_i,x_j)}\right)$$

This reminds us the same kernel trick for linear regression can be applied here.

- KNN

Majority vote, Larger K => predicted label is more stable; Smaller K => predicted label is more accurate

- KNN Density Estimate & Kernel(Local) Regression

Probability density estimate: counting discrete points within the region

$$\hat{p}(x) = \frac{\sum_j^n \frac{1_{\left\|x_j-x\right\|\le\Delta}}{\Delta}}{n}$$

n total points, Δ size of the local region. Or replace

$$\frac{1_{\left\|x_j-x\right\|\le\Delta}}{\Delta} \tag{k-1}$$

by

$$\frac{1}{\sqrt{2\pi}\Delta}\, e^{-\frac{\left(x_j-x\right)^2}{2\Delta^2}} \tag{k-2}$$

So both $\hat{p}(x)$ act as smoother.

Kernel Regression: instead of counting, we average local data points to make prediction.

Let X_i be the position of ith data point, Y_i be its value.

Nadaraya–Watson kernel regression says

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^n K_h(x-x_i)y_i}{\sum_{i=1}^n K_h(x-x_i)}$$

where K any kernel e.g. above (k-1), (k-2).

Locally Weighted Linear Regression:

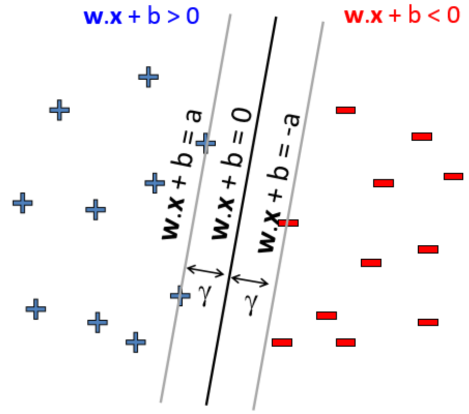
$$y = \theta^T x \;\; subject\ to \min_{\theta} \sum_i w_i (y_i - \theta^T x_i)^2, \;\; w_i = e^{-\frac{(x_i - x)^2}{2\tau^2}}$$

y_i, x_i are ith data point (vector-value), x in w_i is the point to evaluate. Thus this is a continue learning model.

8. Support Vector Machine

- ### Optimal Margin Classifier

Decision boundary $w^T x + b = 0$. Data feature x, label $y = \{\pm 1\}$.



Page 11 http://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture11.pdf

In graph above, $y_i = \pm 1$ points are separated by a hyperplane, so all points satisfy

$$y_i(w^T x + b) \geq a$$

Points on lines $w^T x + b = \pm a$ are called supported points. The margin (cushion of error)

$$2\gamma = \frac{2a}{\|w\|}$$

Proof: consider one of the supported points, x_p, on $w^T x + b = a$. Denote its root to $w^T x + b = 0$ to be p. Then the vector

$$\overline{p x_p} = \gamma \frac{w}{\|w\|}$$

Since p in on $w^T x + b = 0$,

$$w^T \left(x_p - \gamma \frac{w}{\|w\|} \right) + b = 0$$

and x_p is on $w^T x + b = a$,

$$w^T x_p + b = a$$

Therefore

$$\gamma \frac{w^T w}{\|w\|} = a => \quad \gamma = \frac{a}{\|w\|}$$

We have 3 unknowns: w, b, a. But clearly one can multiply them by a constant and the boundary line won't change. So let's fix $a = 1$.

Therefore our goal to maximize cushion of error becomes, i.e.

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad s.t. \ y_i(w^T x_i + b) \geq 1 \ \ \forall i \tag{SVM-1.1}$$

Then to predict testing data x, compute

$$sign(w^T x + b) \tag{SVM-1.2}$$

If the set is not linearly separable (above has no solution), soft margin approach, allow misclassified, say data i on the wrong side of its corrected supported line, and let ξ_i be slack variables.

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum \xi_i \quad s.t. \ y_i(w^T x_i + b) \geq 1 - \xi_i \ \ and \ \xi_i \geq 0 \tag{SVM-1.3}$$

and clearly

$$\xi_i = \max\{0, 1 - y_i \underbrace{(w^T x_i + b)}_{predict}\} = \begin{cases} 0 & y_i(w^T x_i + b) \geq 1 \\ 1 - y_i(w^T x_i + b) & y_i(w^T x_i + b) < 1 \end{cases}$$

called Hinge loss

Consider the case $y_i(w^T x_i + b) < 1$, ξ_i ∝ distance from the its correct side supported vector line.

If $0 < y_i(w^T x_i + b) < 1$, x_i is on the correct side but it stays in the error margin between its correct side supported vector line and the middle line.

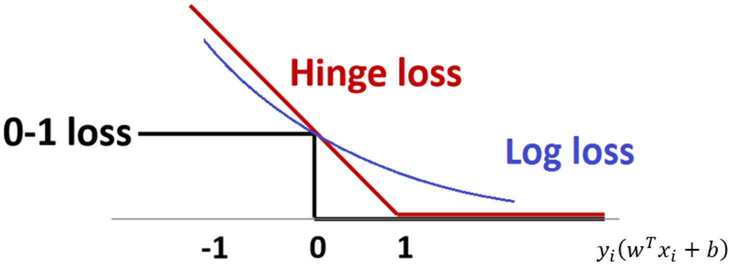
If $-1 < y_i(w^T x_i + b) < 0$, x_i is on the wrong side and in between the other side supported vector line and the middle line.

If $y_i(w^T x_i + b) < -1$, x_i is on the wrong side and further away from the other side supported vector line.

There is another way to count for non-linear separable case.

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\ \#mistake \quad s.t.\ y_i(w^T x_i + b) \geq 1$$

This turns out to be non-QP. And the $\#mistake$ term makes this to be 0-1 loss.



For comparison we add in logistic regression classification, recall for logistic regression $y_i = \{0,1\}$

$$f(x_i) = \frac{1}{1 + e^{-(w^T x_i + b)}}$$

and the loss

$$loss(f(x_i), y_i)) = (y_i - 1)\log(1 - f) - y_i \log(f - 0) = \begin{cases} \log\left(1 + e^{(w^T x_i + b)}\right) & y_i = 0 \\ \log\left(1 + e^{-(w^T x_i + b)}\right) & y_i = 1 \end{cases}$$

If we make it compatible with current discussion $y_i = \pm 1$, we see that loss function is also a function of $y_i(w^T x_i + b)$

$$loss(f(x_i), y_i) = -\log P(y_i | x_i, w, b) = \log\left(1 + e^{-y_i(w^T x_i + b)}\right)$$

- Multi-class SVM

Simultaneously learn multi-class say y has 3 values.

$$\min_{W,b} \sum_y w^{(y)} w^{(y)} + C \sum_j \sum_{y \neq y_j} \xi_j^{(y)}$$

$$w^{(y_j)} x_j + b^{(y_j)} \geq w^{(y)} x_j + b^{(y)} + 1 - \xi_j^{(y)},$$

$$\xi_j^{(y)} \geq 0, \qquad \forall y \neq y_j, \forall j$$

This produces 3 sets of w^k, b^k, ξ^k k=1,2,3. Each set provides a boundary to separate its kind to the rest of other kinds. If the predicted class fells into multiple classes, use the one has longest distance to the supported line, which is

$$\frac{w^T x + b - 1}{||w||} \tag{SVM-2}$$

One can also choose to do this not simultaneously, i.e. use usual SVM for each kind against the rest, then apply (SVM-2). But if we solve multi-class simultaneously then the 3 sets of w, b will have same scale. So predicting y(x), (SVM-2) can be simplify to

$$y = \arg\max_k \left(w^k\right)^T x + b^k$$

- Lagrange Duality

To solve (SVM-1.1), we construct Lagrange

$$L(w,b) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i [y_i(w^T x_i + b) - 1]\,, \alpha_i \geq 0$$

Then

$$\max_{\alpha \ni \alpha \geq 0}(L) = \begin{cases} \frac{1}{2}\|w\|^2 & if\ w, b\ satisfy\ constraint\ in\ (SVM\ 1.1) \\ \infty & otherwise \end{cases}$$

because if w, b satisfy the inequality constraint in SVM-1.1, put $\alpha_i = 0$, which tells us for non-supported points $\alpha_i = 0$. This is also part of KKT condition. Then SVM-1.1 is equivalent to

$$\min_{w,b} \max_{\alpha \ni \alpha \geq 0}(L)$$

We know under KKT conditions,

$$\max_{\alpha \ni \alpha \geq 0} \min_{w,b}(L) = \min_{w,b} \max_{\alpha \ni \alpha \geq 0}(L)$$

Let us see what the dual problem is. Find $\min_{w,b}(L)$,

$$\nabla_{\mathrm{w}} L = 0\ (This\ also\ met\ KKT\ ) => w = \sum_i \alpha_i \underbrace{y_i x_i}_{\substack{vector \\ comp. \\ prod}}$$

$$and\ \ \frac{\partial L}{\partial b} = 0\ (This\ met\ KKT\ ) => \sum \alpha_i y_i = 0$$

Plugging them in $\min_{w,b}(L)$, we get dual problem:

$$\max_{\alpha \ni \alpha \geq 0} \min_{w,b}(L) = \max_{\alpha}\left(\sum_i \alpha_i - \frac{1}{2}\sum_{ij} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle\right) s.t.\ \alpha_i \geq 0\ \ and\ \sum \alpha_i y_i = 0$$

This is a quadratic programming problem with one local optimum.

After we find α_i^* (dependent on $\langle x_i, x_j\rangle$) solves the dual problem, then we get

$$w^* = \sum_i \alpha_i y_i x_i$$

$$b^* = -\frac{\max_{i \ni y_i = -1} w^{*T} x_i + \min_{i \ni y_i = 1} w^{*T} x_i}{2}$$

which is the midpoint of the y-intersection of two supported points. Or

$$b^* = y_k - w^T x_k \ \ for\ any\ k \ni \alpha_k > 0$$

since non-supported point whose $a_i = 0$.

And SVM-1.2, i.e. to predict test data x, becomes

$$w^T x + b = \sum_{\substack{i \in Supported \\ vectors}} \alpha_i y_i \underbrace{\langle x_i, x \rangle}_{K(x_i, x)} + b$$

Since decision surface depends on kernel, kernel trick can be applied.

Recall in the preliminary we showed that Vapnik–Chervonenkis dimension VC(linear separating hyperplanes in n dim) = n + 1. Hence regardless how initial data seem to be non-linear separable, one can lift up the dimension (using kernel) so that it can be separated, but this is likely to result overfitting.

So we resolve it to the second method: SVM-1.3. The correct Lagrange is

$$L(w, b, \xi, \alpha, r) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i [y_i(w^T x + b) - 1 + \xi_i] - \sum_i r_i \xi_i$$

and the dual problem

$$\max_{\alpha} \left(\sum_i \alpha_i - \frac{1}{2} \sum_{ij} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \right) s.t.\ \ C \geq \alpha_i \geq 0 \ \ and \ \sum \alpha_i y_i = 0$$

This is quadratic programming problem with a single local optimum and C is found by cross-validation. Again kernel trick can be applied.

After solving α_i,

$$w^* = \sum \alpha_i y_i x_i, \ \ b^* = y_k - w^T x_k \ \ for\ any\ k \ni C > \alpha_k > 0$$

- SVM Kernel

For non-linear separable set, try features of features

$$\phi = \{x_1, x_1 x_2, x_1^2, \dots\}$$

to increase dimension to make it separable. After changing feature space, we solve

$$\max_{\alpha}\left(\sum_i \alpha_i-\frac{1}{2}\sum_{ij} y_i y_j \alpha_i \alpha_j \underbrace{\phi^T(x_i)\phi(x_j)}_{K(x_i,x_j)}\right)\ s.t. C \geq \alpha_i \geq 0\ \ and\ \sum \alpha_i y_i=0 \qquad \text{(SVM-3)}$$

After solving α_i,

$$w^*=\sum \alpha_i y_i \phi(x_i),\ \ b^*=y_k-w^T\phi(x_k)\ \ for\ any\ k \ni C>\alpha_k>0$$

The boundary is non-linear

$$w^T\phi(x)+b=\sum \alpha_i y_i K(x_i,x)+b=0$$

Predict test data

$$sign(w^T\phi(x)+b)$$

- Sequential Minimal Optimization Algorithm (SMO)

invent by John Platt https://www.microsoft.com/en-us/research/publication/fast-training-of-support-vector-machines-using-sequential-minimal-optimization/

or simpler version http://cs229.stanford.edu/materials/smo.pdf

We are out to solve (SVM-3).

-Iterate over all pairs α_i, α_j. Keep other constant

-Set bound $L \leq \alpha_j \leq H$

If $y^{(i)} \neq y^{(j)}, L=\max\left(0, \alpha_j-\alpha_i\right), H=\min(C, C+\alpha_j-\alpha_i)$

If $y^{(i)} = y^{(j)}, L=\max\left(0, \alpha_j+\alpha_i-C\right), H=\min(C, \alpha_j+\alpha_i)$

-Eliminate α_i, we get a quadric function in α_j, which has minimal value at

$$a_j^*=a_j-\frac{y_i\left[\sum_l \alpha_l y_l\left(K(x_l,x_i)-K\left(x_l,x_j\right)\right)-(y_i-y_j)\right]}{2K\left(x_i,x_j\right)-K(x_i,x_i)-K(x_j,x_j)}$$

Then set

$$\alpha_j:=\begin{cases} H & a_j^*>H \\ L & a_j^*<L \\ a_j^* & otherwise \end{cases}$$

And

$$\alpha_i := \alpha_i + y_i y_j \left(\alpha_j^{previous} - \alpha_j\right)$$

-repeat till converge

9. Tree

• Information Gain

Let X_i be features, Y label, find the first feature to split based on what criteria? After the first step the algorithm will repeat itself

Entropy of Y

$$H(Y) = -\sum_y P(Y = y) \log P(Y = y)$$

After splitting to $\{x\}$ branch (typically 2 branches per root), conditional entropy

$$H(Y|X_i) = -\sum_x P(X_i = x) \sum_y P(Y = y|X_i = x) \log P(Y = y|X_i = x)$$

The choice of the splitting feature X_i and criteria branch $\{x\}$ is to maximize information gain

$$\arg\max_{i,x} [H(Y) - H(Y|X_i)]$$

Or minimize

$$\arg\max_{i,x} H(Y|X_i)$$

To find $\{x\}$, for classification Y boundary given by majority vote, for continuous Y uses regression, constant, lin fit and detect jumps.

Avoid overfitting: fix maximum number of depth and leaves. Use chi-square test for independence to reduce number of features. Apply pruning so that pruning a sub edges from the bottom will improve results on a validation set.

10. Feature Selection, Model Selection and Ensemble methods: Boosting, Bagging

• Feature Selection

Approach 1: Score each feature and extract a subset

Common scoring bases

- training single feature classifier and cross-validation compare accuracy

$$f_i : X_i \to Y$$

- use mutual information

$$I(X_i, Y) = \sum_{x,y} P(X_i = x, Y = y) \log \frac{P(X_i = x, Y = y)}{P(X_i = x)P(Y = y)}$$

- Use χ^2 statistics to measure dependence between Y and X_i.

Common selecting steps

- choose the highest scoring feature, X_h

- score the rest of feature conditional on X_h, e.g.

$$score(X_i) = Accuracy(predictng\ Y\ from\ X_i\ and\ X_h)$$

$$score(X_i) = I(X_i, Y|X_h)$$

Approach 2: Regulation (MAP)

$$W = \arg\min_W \sum -\log P(Y|X, W) + \lambda\|W\|$$

Try L0 $\|W\|_0 = \#\{i|W_i \neq 0\}$ or $\|W\|_1$.

- Adaptive Boosting (AdaBoost)

m data points. Having a bunch of simple classifier (unlike to be overfitting, but hard to increase accuracy on its own), each

$$h_t : X \to Y = \{-1,1\}, t = 1, \dots, T$$

Boosting together

$$H = sign(\sum a_t h_t)$$

How to pick a_t?

AdaBoost inventor Robert Schapire, http://rob.schapire.net/papers/explaining-adaboost.pdf

- Initial data distribution $D_1 = 1/m$,

- Train weak learners $h_{1,\ldots,T}$ using D_t (what does it mean? Say $m = 2$, D_1 uniform gives the original data. Say in the next step $D_2(1) = \frac{1}{3}, D_2(2) = \frac{2}{3}$. We could pretend there were 2 data points at $i = 2$.)
- Compute error

$$e_t = \sum P_{D_t}(h_t(x_i) \neq y_i) = \sum_i^m D_t(i)\delta(h_t(x_i) \neq y_i)\,, \quad 0 < e < 1$$

- Let

$$a_t = \frac{1}{2}\ln\frac{1-e_t}{e_t} \rightarrow \begin{cases} a_t > 0 \;\; e < \dfrac{1}{2} \\ a_t < 0 \;\; e > \dfrac{1}{2} \end{cases}$$

- Move to D_{t+1} for each $i = 1, \ldots, m$

$$D_{t+1}(i) = \frac{D_t(i)\exp\left[-a_t \underbrace{y_i h_i(x_i)}_{\pm 1}\right]}{\sum_i^m D_t(i)\exp[-a_t y_i h_i(x_i)]}$$

Assume error < ½ (random guess), $D_{t+1}(i)$ increases from $D_t(i)$ for ith point if $h_t(x_i) \neq y_i$. Otherwise it decreases.

- Repeat till $t = T$.

Proof of Adaboost: Since

$$1 < e^{positive\ number}\ and\ \ 0 < e^{negative\ number}\ ,$$

$$total\ Adaboost\ error = \frac{1}{m}\sum_i^m \delta(H(x_i) \neq y_i) \leq \frac{1}{m}\sum_i^m e^{-y_i f(x_i)} := \prod Z_t$$

where $f(x) = \sum_t a_t h_t(x), H(x) = sign(f(x))$, and

$$Z_t = \sum_i^m D_t(i)e^{-a_t y_i h_t(x_i)}$$

because $D_{T+1}(i) = \frac{1}{m}\frac{e^{-y_i f(x_i)}}{\prod_t Z_t}$ and $\sum_{i=1}^m D_{T+1}(i) = 1$.

Thus Adaboost is to minimize the upper bound, via minimize Z_t in each iteration

$$Z_t = \sum_{i \ni y_i \neq h_t(x_i)} D_t e^{a_t} + \sum_{i \ni y_i = h_t(x_i)} D_t e^{-a_t} = e_t e^{a_t} + (1 - e_t) e^{-a_t}$$

Then $\frac{\partial Z_t}{\partial \alpha_t} = \epsilon_t e^{\alpha_t} - (1 - \epsilon_t) e^{-\alpha_t} = 0 \Rightarrow e^{2\alpha_t} = \frac{1-\epsilon_t}{\epsilon_t}$). This proves the choice of a_t. Plugging a_t in Z_t,

$$\prod Z_t = \prod \sqrt{1 - (1 - 2e_t)^2} \leq \exp\left(-2 \sum_t \left(\frac{1}{2} - e_t\right)^2\right)$$

Showing even for weak learners that are slightly better than random guess, Adaboost converges exponentially fast, but could result overfit.

- Bagging-bootstrap aggregating

1.Run independent weak learners on bootstrap replicates (sample with replacement, cross validation) of the training set

2.Average/vote over weak hypotheses

No iteration update weight, only variance reduction

Unsupervised Learning

11.Clustering

Similarity is in form of distance metric: Euclidean, Manhattan, Minkowski, Canberra, … and correlation: Pearson,…

- Hierarchical Clustering (<100 dataset)

Hierarchical algorithms: single-linkage, average-linkage, complete-linkage, centroid-based, minmax… They are Bottom-up algorithm. They are simple to implement but less accurate and computational expensive (O($n^2 \log n$). $O(n^2)$ for pair of distance, and $O(n \log n)$ for sorting).

Single Linkage Algo

Distance of an element, u, to the set V = $\min_{v \in V} D(u, v)$

Distance between two sets = $\min_{u \in U, v \in V} D(U, V)$

- link pairs that have the shortest distance

- Treat the linked pairs as if they are elements, repeat above, continue till there is only one cluster.

Complete Linkage Metric

Distance of an element, u, to the set V = $\max_{v \in V} D(u, V)$

Distance between two sets = $\max_{u \in U, v \in V} D(U, V)$

Average Linkage Metric

Distance of element, u, to the set V = $\operatorname{ave}_{v \in V} D(u, V)$

Distance between two sets = $\operatorname{ave}_{u \in U, v \in V} D(U, V)$

Centroid Linkage Metric

Distance of element, u, to the set V = Distance from u to centroid of V

Distance between two sets U, V = Distance between centroids of U, V

Minimax linkage Metric

Define

Radius of a set: $R(U) = \min_{x \in U} \max_{y \in U} D(x, y)$

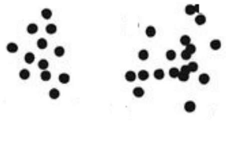
which is the smallest radius of the ball that encloses U and whose center is one of the elements of U.

Distance of element, u, to the set V = $R(u \cup V)$

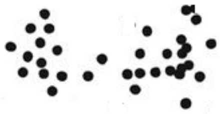
Distance between two sets U, V = $R(U \cup V)$

Single vs. Complete Linkage

Complete Linkage assumes isotopic, convex shapes but robust to outliers; while Single-linkage allows anti-isotopic, non-convex shapes but sensitive to outliers.



Both single and complete linkages will put above graph into two clusters at some level. However, if there are outliers in between the two clusters, single-linkage may pull the two clusters together, at no level recognizes they're two separated clusters. That is because single-linkage only consider min distance. Things can merge into one even if there are elements are very further apart in the cluster.



- ### Partition Clustering (large dataset)

  Partition algorithms: K-mean, Gaussian Mixture-Model (EM) based clustering, …

<u>K-Mean Algo</u>

-pick k initial centroids.

-assign each element to one of the closest centroids.

-update centroid location, reassign each element

-stop if none element change membership.

Computation: $O(nkl)$, n=elements, k=centroids, l=iterations.

Shortcomings: sensitive to outliers, highly depend on initial centroids.

<u>K-medoids Algo</u>

Replace centroid by medoid

robust to outliers.

K-Mean is to minimize potential, μ_i centroids, C_i clusters.

$$\min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j \ni C(j)=j} \left\| \mu_i - x_j \right\|^2$$

12. Expectation Maximization (EM) & Factor Analysis

This is a generative approach. We mix uncertainty in classification. $< X_1, \dots, X_n > n$ mixture data points. They are classifications $< Z_1, \dots, Z_n >$ are unobserved.

- Gaussian Mixture Model

$k = 1, . . , K$, best K can be later found by cross-validation. Assume

$$P(X_i|Z_i = k) \sim N(\mu_k, \Sigma_k), i = 1, \dots , n$$

Recall in Gaussian Bayes Classifier we show when σ is independent of k, boundary is linear. Here we don't assume that. Parameters $\theta = \{\mu_k, \Sigma_k, P(Z = k)\}$

We are not going to maximize total probability

$$\arg\max_\theta \prod_i \log P(X_i, Z_i)$$

The value of Z_i is soft classification, so instead we maximize the expectation of the total probability

$$\arg\max_{\theta'} \prod_i E_{P(Z_i|X_i,\theta)}[\log P(X_i, Z_i, \theta')]$$

Thus

$$\arg\max_{\theta'} \prod_i \sum_k P(Z_i = k|X_i, \theta) \log P(Z_i = k) \underbrace{\frac{1}{\sqrt{\det \Sigma'_k}} \exp\left(-\frac{1}{2}(X_i - \mu'_k)^T \Sigma'_k (X_i - \mu'_k)\right)}_{P(X_i|Z_i=k)} \qquad \text{(GMM)}$$

EM algorithm

-E step:

Randomly pick some initial model parameters $\theta^{(t)} = \{P^{(t)}(z = k), \mu_k^{(t)}, \Sigma_k^{(t)}\}$, and compute

$$P\left(Z_i = k \middle| X_i, \theta^{(t)}\right) = \frac{P(X_i|Z_i = k, \theta^{(t)})P(Z_i = k)}{\sum_k P(X_i|Z_i = k, \theta^{(t)})P(Z_i = k)}$$

Then formulate expectation value (GMM)

-M step:

Maximize above expectation, $\arg\max_{\theta^{(t+1)}} \prod_i E_{P(Z_i|X_i,\theta^{(t)})}\left[\log P\left(X_i, Z_i, \theta^{(t+1)}\right)\right]$, update

$$\mu_k^{(t+1)} = \frac{\sum_i X_i P(Z_i = k|X_i, \theta^{(t)})}{\sum_i P(Z_i = k|X_i, \theta^{(t)})}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_i \left(X_i - \mu_k^{(t+1)}\right)\left(X_i - \mu_k^{(t+1)}\right)^T P(Z_i = k|X_i, \theta^{(t)})}{\sum_i P(Z_i = k|X_i, \theta^{(t)})}$$

$$P^{(t+1)}(z=k)=\frac{\sum_i P(Z_i=k|X_i,\theta^{(t)})}{\#data\ points}$$

Hence used to be δ counts, now all become soft probabilities.

-repeat

- ## EM for Factor Analysis

Above computation of $\Sigma_k^{(t+1)}$ could run into trouble when $dimension\ of\ data \gg number\ of\ data$.

The jj diagonal element of $\Sigma_k^{(t+1)}\ \sim \sum_i [jth\ component\ of\ (x_i-\mu_k)]^2$

When $dimension\ of\ data\ (dimension\ of\ vector\ x_i) \gg number\ of\ data$, it is highly likely there will be some jj diagonal element of $\Sigma_k^{(t)}$ is 0. Hence self-adjoint $\Sigma_k^{(t)}$ becomes singular.

One fix is that when $dimension\ of\ data \gg number\ of\ data$, we force

$$\Sigma_k^{(t)} := \sigma_k^{(t)} I$$

where $\sigma_k^{(t)} = ave\left(\Sigma_k^{(t)}\right)_{jj}$.

A better fix is Factor Analysis.

The cause of singularity is that on some dimension j all data points have same value. To fix that, we add noise to data.

Formally, let (x,z) be data. $x \in R^n, z \in R^k$, z latent variables. From preliminary: Conditional Gaussian & Marginal Gaussian, we know if

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_x \\ \mu_z \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{zz} \end{bmatrix}\right)$$

then the condition probability is still Gaussian

$$P(x|z) = N\left(\mu_{x|z} = \mu_x + \Sigma_{xz}\Sigma_{zz}^{-1}(z-\mu_z), \Sigma_{x|z} = \Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx}\right) \qquad \text{(CG-1)}$$

In our case, we assume

$$z \sim N(\mu_z = 0, \Sigma_{zz} = I)$$

Thus

$$P(x|z) = N(\mu_x + \Sigma_{xz} z, \Sigma_{xx} - \Sigma_{xz}\Sigma_{zx})$$

This shows what kind of white noise to be added to x.

$$\epsilon \sim N(0, \Sigma_{xx} - \Sigma_{xz}\Sigma_{zx})$$

$$x^{new} \coloneqq x + \Sigma_{xz} z + \epsilon$$

Proof:

$$\begin{aligned} Var(x^{new}) &= E[(x^{new} - E[x^{new}])(x^{new} - E[x^{new}])^T] = E[(\Sigma_{xz} z + \epsilon)(\Sigma_{xz} z + \epsilon)^T] \\ &= \Sigma_{xz} E[zz] \Sigma_{zx} + E[\epsilon\epsilon^T] = \Sigma_{xx} \end{aligned}$$

$$\begin{aligned} Cor(x^{new}, z) &= E[(x^{new} - E[x^{new}])(z - E[z])^T] = E[(\Sigma_{xz} z + \epsilon) z^T] = \Sigma_{xz} + E[\epsilon z^T] \\ &= \Sigma_{xz} + E[\epsilon] + E[z] + Cov(\epsilon, z) = \Sigma_{xz} \end{aligned}$$

Hence nothing changes. QED

So that we can make new computation Σ_{xx} non-singluar. Then by inverse conditional Gaussian. Inverse (CG-1), we have the solution in the form

$$P(z|x, \mu_x, \Sigma_{xx}, \Sigma_{xz}) = N\left(\mu_{z|x} = \Sigma_{zx}\Sigma_{xx}^{-1}(x - \mu_x), \Sigma_{z|x} = \underbrace{\Sigma_{zz}}_{I} - \Sigma_{zx}\Sigma_{xx}^{-1}\Sigma_{xz}\right) \qquad \text{(CG-1)}$$

Now use EM to find $\mu_x, \Sigma_{xx}, \Sigma_{xz}$. If necessary, use coordinate transformation

$$\Lambda = \Sigma_{xx}, \Psi = \Sigma_{xx} - \Sigma_{xz}\Sigma_{zx}$$

General Proof of EM Convergence

x observed, z missing label (hidden/latent data), θ model

$$\log \prod P(x_i|\theta) = \sum_i \log P(x_i|\theta) = \sum_i \log \sum_k P(x_i, z = k|\theta)$$

Above formula has “log of sum” is not useful. Luckily we have Jansen’s.

Let $Q_i(z)$ be some distribution over z.

f concave function

$$f(ax_1 + (1 - a)x_2) \geq af(x_1) + (1 - a)f(x_2)$$

Here $\sum_z Q(z|x_i) = 1$, and log is concave.

$$\begin{aligned} \sum_i \log \sum_k P(x_i, z = k|\theta) \quad &= \quad \sum_i \log \sum_k Q_i(z|x_i) \frac{P(x_i, z = k|\theta)}{Q_i(z|x_i)} \\ &\geq \quad \sum_i \sum_k \underbrace{Q_i(z = k|x_i) \log \frac{P(x_i, z = k|\theta)}{Q_i(z=k|x_i)}}_{\propto P(z=k|x_i,\theta)} \end{aligned} \qquad \text{(EM-1)}$$

The Jensen's inequality becomes equality when

$$\frac{P(x_i, z = k|\theta)}{Q_i(z = k|x_i)} = constant\ independ\ of\ k \rightarrow independ\ of\ z$$

That is

$$f(ax_1 + (1-a)x_1) = af(x_1) + (1-a)f(x_1)$$

So to maximize the lower bound (EM-1), we let

$$Q^{(t)}(z = k|x_i) = P^{(t)}(z = k|x_i, \theta^{(t)})$$

i.e. use the best guess (posterior) for the latent variable. Proof

$$\frac{P(x_i, z = k|\theta)}{Q_i(z = k|x_i)} = \frac{P(z|x,\theta)P(x,\theta)}{P(z|x,\theta)} = P(x,\theta)$$

This is also known as turning Kullback–Leibler divergence into equality.

-E step

With some initial $\theta^{(t)}, Q^{(t)}$

formulate expectation with distribution $P(z = k|x_i, \theta) \propto Q_i(z = k|x_i) \log \frac{P(x_i, z = k|\theta)}{Q_i(z=k|x_i)}$

-M step

Maximize above expectation wrt $\theta^{(t)}$, update $Q^{(t)} = P^{(t)}(z = k|x_i, \theta^{(t)})$

-repeat

Above algorithm will monotonically increase expectation after each iteration.

13. Dimension Reduction—Linear: PCA, ICA, CCA

These lower Dimensional Representations are invoking latent features. They are very different from feature selection, which work by ignoring some features.

- ### Principal Component Analysis

Let $X = (X_1, X_2, \ldots, X_n)$ be n piece of training data, each X_i is vector of D dimensional features. Hence X is $D \times n$ matrix with one column per data point. To make later computation simple, we shift the entire data to be centered at the origin

$$X_i = X_i - \underbrace{ave(X)}_{\bar{X}}$$

Let $V_1, \ldots, V_d$ be the principal components, vectors of D dimension, i.e. they are orthonormal to each other. We want to represent

$$\hat{X}_i = \sum_{l=1}^{d} a_{il} V_l = \sum_{l=1}^{d} \langle X_i, V_l \rangle V_l$$

(here is why we need to shift X to the origin, because V are the new axes) and make the reconstruction error

$$\sum_i \left\| X_i - \hat{X}_i \right\|^2 = \sum_i \|X_i\|^2 + \underbrace{\left\|\hat{X}_i\right\|^2}_{1} \quad - 2X_i \cdot \hat{X}_i$$

as small as possible. Hence maximize

$$\textstyle\sum_i X_i \cdot \hat{X}_i = \sum_{i=1}^{n} X_i \cdot \sum_{l=1}^{d} \langle X_i, V_l \rangle V_l = \sum_{i,l} \langle X_i, V_l \rangle^2 = \sum_l \underbrace{V_l^T X X^T V_l}_{:=\lambda_l} \qquad \text{(PCA-1)}$$

The goal is

$$\max_{V_l} V_l^T X X^T V_l \quad subject\ to\ V_l^T V_l = 1$$

The interpretation of this is to find V_l such that projections on to it capture maximum variance in the data

Lagrange multiplier

$$\max(V_l^T X X^T V_l - \lambda V_l^T V_l)$$

Take derivative wrt V_l and set it to 0

$$(XX^T)V_l = \lambda_l V_l$$

Therefore, the problem is reduced to finding the eigenvector with largest eigenvalue (why largest? See PCA-1).

What is XX^T? It is the covariance matrix multiplied by n. By definition, the covariance matrix of a $D \times 1$ dimensional random vector (i.e. multivariate) R is

$$Var[R] = \mathbb{E}[(R - \mathbb{E}[R])(R - \mathbb{E}[R])^T]$$

Agree perfectly to our centered X, thus XX^T

$$(XX^T)_{\alpha\beta} = \sum_i (X_i)_\alpha (X_i)_\beta$$

After we find $V = (V_1, \dots, V_d)$, what is the reconstruction error?

$$\frac{1}{n}\sum_i \left\|X_i - \hat{X}_i\right\|^2 = \sum_{l=d+1}^{D} \lambda_l$$

- Singular Value Decomposition

Any matrix X, $X = VSU^T$, then

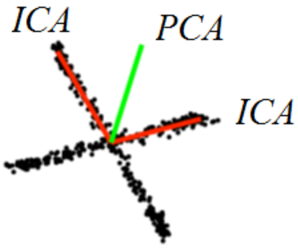
$$XX^T = VS^2V^T \rightarrow XX^TV = VS^2$$

where $V = (V_1, \dots, V_d, \dots)$ matrix whose columns are orthonormal eigenvectors of XX^T and S^2 is a diagonal matrix whose diagonal elements λ_l are the corresponding eigenvalues of XX^T. Each column of V is the principal axis, each column of (SU^T) are the coordinates, because

$$X_i = V(SU^T)_{i\text{th column}}$$

- Independent Components Analysis

Recall in PCA we first find the direction that has the largest variance in the random vectors.



Picture from Nathan Kutz http://courses.washington.edu/amath582/582.pdf page 359

If the data has two independent variance directions, PCA won't be good. Or if the second moment of data is

$$XX^T \sim I$$

white noise. ICA seeks $V_1, \dots, V_d$ that are most statistically independent to each other and they are non-Gaussian, the sum of them minimizes

$$I(X, V_1, \dots, V_d) = \sum_i^d H(V_i) - H(X)$$

More on ICA algorithm derived from maximization of non-Gaussianity, or maximum likelihood estimation, see Aapo Hyvärinen http://www.cs.helsinki.fi/u/ahyvarin/teaching/uml2015/uml2015_lecturenotes.pdf pages 49-64

- ### Canonical Correlation Analysis

Find common low dimension basis for two datasets A, B. PCA will be to minimize

$$\sum_{X_i \in A} \left\| X_i - \hat{X}_i \right\|^2 + \sum_{Y_i \in B} \left\| Y_i - \hat{Y}_i \right\|^2$$

CCA will be to maximize (already centered)

$$Cov(A, B) = \frac{1}{N} \sum_{i}^{N} \frac{(\hat{X}_i)(\hat{Y}_i)}{\sigma_X \sigma\_Y}$$

14. Non-Linear Dimensionality Reduction

Manifold Preserving

Laplacian Eigenmaps, ISOMAP, Local Linear Embedding,…

Tony Jebara http://www.cs.columbia.edu/~jebara/6772/notes/notes2.pdf

https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

Structured Models

15. Hidden Markov Models

Study sequential data: time series, speech recognition, gene sequence…

Joint Distribution $P(X_1, X_2, \dots, X_t) = P(X_1)P(X_2|X_1)P(X_3|X_2, X_1) \dots = \prod_i^t P(X_i|X_{i-1}, \dots, X_1)$

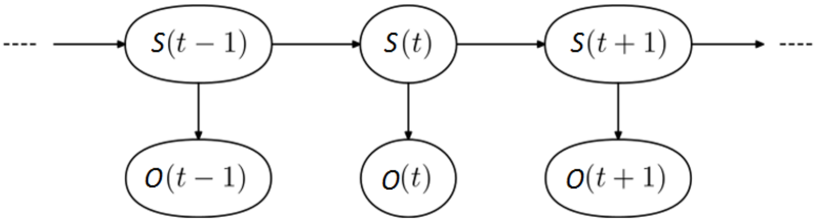
Markov Assumption mth order: states depend on limited past history.

$$P(X_i|X_{i-1}, \dots, X_1) = P(X_i|X_{i-1}, \dots, P_{i-m})$$

E.g. Linear mth order autoregressive model $AR(m)$

$$P(X_i|X_{i-1}, \dots, X_{i-m}) \sim N(LN(X_{i-1}, \dots, X_{i-m}), \sigma)$$

some linear function $LN(X_{i-1}, \dots, X_{i-m}) = c_i + \sum_j^m a_j X_{i-j}$, where a_j's are independent of i, which is called homogeneous/stationary Markov model



O observation states, S hidden states.

1st order HMM, total probability

$$P(S_1, \dots, S_T; O_1, \dots, O_T) = \prod_{t=1}^{T} P(O_t|S_t)\, P(S_1) \prod_{t=2}^{T} P(S_t|S_{t-1})$$

The parameters θ involved are the initial probability $P(S_1 = s) = \pi_s$, transition probability $P(S_t = j|S_{t-1} = i) = p_{ij}$, and emission probability $P(O_t = y|S_t = i) = q_i^y$.

16. Three Main Questions for HMM

• Forward Algorithm

- EVALUTION: given HMM parameters, find how likely an observed sequence $\{O_i\}$ can appear.

$$P(O_1, \dots, O_T)$$

Use forward algorithm.

Brutal force says

$$P(O_1, \dots, O_T) = \sum_{\{S_t\}} P(S_1, \dots, S_T; O_1, \dots, O_T) \qquad \text{(HMM-BF)}$$

Forward algorithm uses dynamic programming so that the computation is linear in number of parameters. Initialize

$$\alpha_1^k = P(O_1|S_1 = k)P(S_1 = k)$$

Forward stepping (filtering), iterate

$$\alpha_t^k = P(O_1, \dots, O_t, S_t = k) = P(O_t|S_t = k) \sum_i a_{t-1}^i P(S_t = k|S_{t-1} = i)$$

Terminate $t = T$,

$$P(O_1, \dots, O_T) = \sum_k \alpha_T^k$$

- DECODING: given HMM parameters and an observed sequence, find the hidden states.

• Forward-Backward Algorithm

Find a particular hidden state $P(S_t = k|\{O_t\}_{t=1}^{T})$, use Forward-Backward algorithm.

Initialize α_1^k and

$$\beta_T^k = 1$$

Forward stepping (filtering), Iterate α_t^k and backward stepping (smoothing), iterate

$$\beta_t^k = P(O_{t+1}, \dots, O_T | S_t = k) = \sum_i P(S_{t+1} = i | S_t = k) P(O_{t+1} | S_{t+1} = i) \beta_{t+1}^i$$

Terminate at $t = t$. Since $P(S_t = k; \{O_t\}_{t=1}^T) = \alpha_t^k \beta_t^k$,

$$P(S_t = k \middle| \{O_t\}_{t=1}^T) = \frac{P(S_t = k; \{O_t\}_{t=1}^T)}{P(\{O_t\}_{t=1}^T)} = \frac{\alpha_t^k \beta_t^k}{\sum_i \alpha_t^i \beta_t^i}$$

Hence the most likely state at t

$$\arg\max_k \alpha_t^k \beta_t^k$$

- ## Viterbi Algorithm

Find the entire hidden states sequence $\{S_t\}$, use Viterbi algorithm.

$$\arg \max_{\{S_t\}_{t=1}^T} P(S_1, \dots, S_T | O_1, \dots, O_T)$$

Initialize

$$V_1^k = P(O_1 | S_1 = k) P(S_1 = k\ )$$

Iterate

$$V_t^k \coloneqq \max_{\{S_t\}_{t=1}^{t-1}} P(S_1, \dots, S_{t-1}, S_t = k; O_1, \dots, O_T)$$

$$= P(O_t | S_t = k) \max_i P(S_t = k | S_{t-1} = i) V_{t-1}^i$$

Terminate at $t = T$,

$$S_T = S_T^* = \arg\max_k V_T^k$$

Trace back,

$$S_{t-1}^* = \arg\max_i P(S_t^* | S_{t-1} = i) V_{t-1}^i$$

- ## Baum-Welch Algorithm (EM)

- LEARNING: assume HMM and given an observed sequence, find the parameters for the HMM

$$\arg\max_\theta P(O_1, \dots, O_T | \theta)$$

Use Baum-Welch Algorithm (EM).

Initialize with random parameters

E-step

$$\gamma_i(t) = P(S_t = i \middle| \{O_t\}_{t=1}^{T}, \theta) = \frac{\alpha_t^i \beta_t^i}{\sum_j \alpha_t^j \beta_t^j}$$

$$\xi_{ij}(t) = P(S_{t-1} = i, S_t = j \middle| \{O_t\}_{t=1}^{T}, \theta) = \frac{\gamma_i(t-1) p_{ij} q_j^{O_t} \beta_t^j}{\beta_{t-1}^i}$$

M-Step, maximize expectation $\sum \gamma_i, \sum \xi_{ij}$, solve

$$\pi_i = \gamma_i(1), \qquad q_i^k = \frac{\sum_{t=1}^{T} \delta_{O_t=k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}, \quad p_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

17. Kalman Filter (Linear Dynamical Systems)

“Filter” means forward stepping--current knowledge continuously shaping the understanding of future observations.

Used in GPS navigation, computer vision devices. Same as HMM with continuously changing S. Observations and States are multivariate Gaussians whose means are linear functions of their parent states.

$$S_1 \sim N(0, \Sigma_{initial})$$

$$S_t = AS_{t-1} + \sim N(0, R)$$

$$Q_t = BS_t + \sim N(0, R')$$

A transition matrix, B observation matrix. Use Conditional Gaussian and Marginal Gaussian equation (CG-MG) from preliminary section,

$$\mathbb{E}\left[S_{t|t-1}\right] = A \cdot \mathbb{E}[S_{t-1}]$$

$$var\left[S_{t|t-1}\right] = A \cdot var[S_{t-1}] \cdot A^T + R$$

$$\Sigma = \begin{bmatrix} var[S_{t|t-1}] & var[S_{t|t-1}] \cdot B^T \\ B \cdot var[S_{t|t-1}] & B \cdot var\left[S_{t|t-1}\right] \cdot B^T + R \end{bmatrix}$$

Update

$$\mathbb{E}[S_t] = \mathbb{E}\left[S_{t|t-1}\right] + \Sigma_{12}\Sigma_{22}^{-1}(O_t - \mathbb{E}(O_t))$$

$$var[S_t] = var\left[S_{t|t-1}\right] - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

See Bishop 13.3.33

Roger Labbe, *Kalman and Bayesian Filters in Python* (Py notebook, PDF) and FilterPy

18. Graphical Models: Representation

Move from HMM (sequential dependence) to lose dependence.

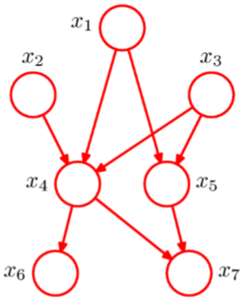
The graph represents sets of conditional independence assumptions.

- ### Direct Graph: Bayesian Networks (BN)

Useful for designing models

BN is a directed acyclic graph that has no cyclic dependence, Directed Acyclic Graph (DAG) and it satisfies Markov assumption, Conditional Probability Tables (CPT): X is are conditional independent to its non-decedents given its immediate parents. Hence we separate grandparents from grandchildren.

$$p(x_1, x_2, \ldots, x_6) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$



http://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture18.pdf page 11

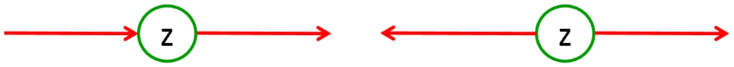
Representation Theorem: the following two sets are equivalent

- Set of distributions that factorize according to the graph

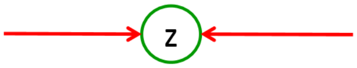
- Set of distributions that respect conditional independencies implied by d-separation properties of graph

Representation theorem says A is D-separated from B by C iff $A \perp B|C$. It links conditional independence with graphical structure.

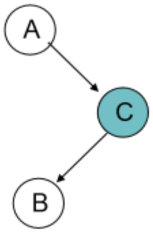
Let A, B, C disjoint sets of nodes. C can be empty. We say A, B are D-separated by C if every path between A and B contains a node z such that (1) either z is a non-collider in C



Or (2) z is a collider and neither z nor its decedents in C

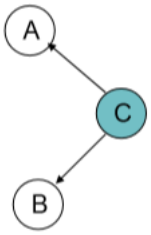


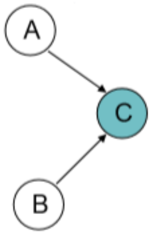
Proof:



$$\longleftarrow P(a,b|c) = \frac{P(a,b,c)}{P(c)} = \frac{P(a)P(c|a)P(b|c)}{P(c)} = P(a|c)P(b|c)$$

$$P(a,b|c) = \frac{P(a,b,c)}{P(c)} = \frac{P(a|c)P(b|c)P(c)}{P(c)} = P(a|c)P(b|c) \rightarrow$$



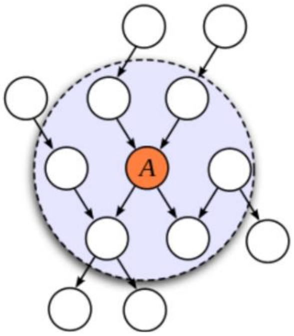


$$P(a,b) = \sum_c P(a,b,c) = P(a)P(b)\sum_c P(c|a,b) = P(a)P(b)$$

but it does not imply $P(a|c) = P(a|b,c)$

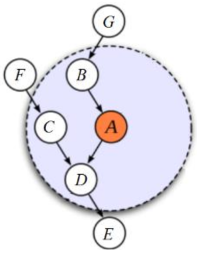
Markov Blanket

The set of nodes that relate to A. They are A 's parents, its children, and its children's other parents.



Picture from wiki http://en.wikipedia.org/wiki/Markov_blanket

Proof



$$P(A) = \sum_{\{B,C,\ldots,G\}} P(A,\ldots,G)$$

$$= \sum_{\{B,C,\ldots,G\}} \underbrace{P(G)P(B|G)}_{P(G|B)P(B)} P(A|B) \underbrace{P(F)P(C|F)}_{P(F|C)P(C)} P(C,A|D)P(E|D)$$

Since $\sum_E P(E|D) = \sum_F P(F|C) = \sum_G P(G|B) = 1$, we get

$$P(A) = \sum_{\{B,C,D\}} P(B)P(A|B)P(C)P(C,A|D)$$

QED.

Approximation Methods: Monte Carlo, Variational Method, Expectation Propagation

Similar to the proof above, say one wants to infer,

$$P(A = a|G = g, D = d) = \frac{P(A, G, D)}{P(G, D)} = \frac{\sum_{\{A,\ldots,G\}/\{A,G,D\}} P(A, \ldots, G)}{\sum_{\{A,\ldots,G\}/\{G,D\}} P(A, \ldots, G)}$$

One'll have to do the giant summation which has

$$2^{\#\ elements\ in\ the\ Markov\ blankets\ of\ A,G,D\ excluding\ themselves}$$

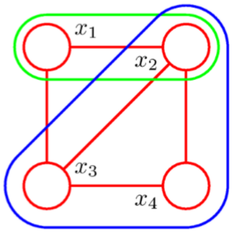
if all RV's are binary.

Hence it is NP-hard. We have seen this in HMM brutal force (HMM-BF). Luckily there we have elegant algorithms to make it linear in number of parameter. Here we can use Monte Carlo to make it linear.

- Indirect Graph: Markov Random Fields (MRF)

Applications in computer vision, statistical physics…

A complete graph is a simple (no loop to its own vertex, single edge) undirected graph in which every pair of distinct vertices is connected by a unique edge. A clique is a subset of vertices of an undirected graph such that its induced subgraph is complete.



Clique, x_C = {x_1,x_2}

Maximal clique
x_C = {x_2,x_3,x_4}

Representation Theorem: Hammersley-Clifford Theorem

The undirected graph is factorized according to

$$P(X_1, \ldots X_4) = \frac{\prod_C \Psi_C(X_C)}{Z} = \frac{\psi(X_1, X_2, X_3)\psi'(X_2, X_3, X_4)}{Z}$$

C is the set of maximal cliques in the graph, Z is the normalization.

$$Z = \sum_X \prod_C \Psi_C(X_C)$$

In the above formula, we see $\{X_1, X_2, X_3\}, \{X_2, X_3, X_4\}$ are factors. Recall in directed graph, factors are related to D-separation (conditional independent). For undirected graph, we say A, B, C non-intersecting set of nodes, that

$$A \perp B \mid C$$

if all paths between nodes in A & B are "blocked" by C. In example above

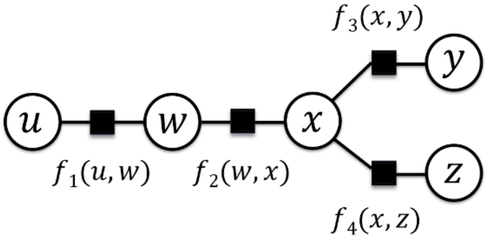
$$\{X_1\} \perp \{X_4\} \mid \{X_2, X_3\}$$

Lattice Theory

Graphical Models image denoising and its relation to Ising model (spin glasses, ferromagnetism) - Christopher Bishop http://youtu.be/c0AWH5UFyOk?t=2064

- ## Factor Graphs

Useful for inference and learning. Factor graph provides explicit factorization. (un)directed graphs can be written in factor graphs. Below is a undirected graph written in factor form.



From Chris Bishop http://mlss.tuebingen.mpg.de/2013/bishop_slides.pdf page 78

$$P(u,w,x,y,z) = f_1(u,w) f_2(w,x) f_3(x,y) f_4(x,z)$$

Then to compute

$$P(w) = \sum_{u,x,y,z} P(u,w,x,y,z)$$

Hence brutal force says it is $O(K^4)$, K is the number of discrete values that u, w, x, y, z take on.

As before we can interchange summations, and this will allow us to use dynamic programming: inductive step

$$P(w) = \underbrace{\sum_{u} f_1(u,w)}_{m_{f_1\to w}(w)} \underbrace{\sum_{x,y,z} f_2(w,x) f_3(x,y) f_4(x,z)}_{m_{f_2\to w}(w)}$$

$m_{f_2\to w}(w)$ is called the message from f_2 to w, and it's a function of w.

$$m_{f_2\to w}(w) = \sum_{x} f_2(w,x) \underbrace{\underbrace{\sum_{y} f_3(x,y)}_{m_{f_3\to x}(x)} \underbrace{\sum_{z} f_4(x,z)}_{m_{f_4\to x}(x)}}_{m_{x\to f_2}(x)}$$

We summarize it for tree structure factor graph

$$P(x) = \prod_{f\in\{factors\ to\ x\}} m_{f\to x}(x)$$

$$m_{f\to x}(x) = \sum_{x_i\in\{arguments\ of\ f\ expect\ x\}} f(x_i) \prod_{x_i\in\{arguments\ of\ f\ expect\ x\}} m_{x_i\to f}(x_i)$$

$$m_{x\to f}(x) = \prod_{f_i\in\{factors\ to\ x\}/\{f_i\}} m_{f_i\to x}(x)$$

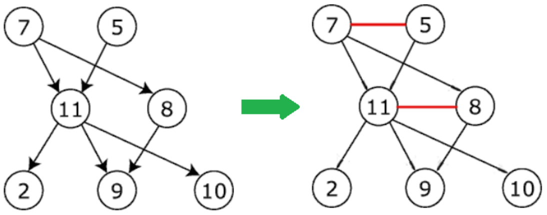
The last two equations are recursive definitions. Each links two opposite direction messages. After all messages being computed, computing marginal probability $P(each\ node)$ becomes $O(K)$.

19. Graphical Models: Inference

ask questions arose from the joint probability distribution. Above we showed variable eliminations for tree graph.

- ### Moral Graphs & Junction Tree Algorithm

For complex directed graph or for increasing inference speed, we convert it to moral graph by marrying nodes that share child.



Picture from https://en.wikipedia.org/wiki/Moral_graph

Then to find marginal probability, we first replace nodes by factors

$$P(X_7) = \sum_{X_5,X_{11},X_8} \psi_1(X_7,X_5,X_{11})\psi_2(X_7,X_8,X_{11}) = \underbrace{\sum_{X_{11}} \underbrace{\sum_{X_5}\psi_1(X_7,X_5,X_{11})}_{f_1(X_{11},X_7)} \underbrace{\sum_{X_8}\psi_2(X_7,X_5,X_8)}_{f_2(X_8,X_7)}}_{f_3(X_7)}$$

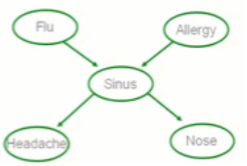
This way the problem becomes linear in # variables but becomes exponential in size of largest factor generated, which is approximately equal to tree-width (max clique size-1) in moral graph.

20.Graphical Models: Learning

learn the parameters and structure of a graphical model

- ### Variables Elimination-Dynamic Programming

learning CPT from fully observed data and known graph structure.



Following http://www.cs.cmu.edu/~tom/10701_sp11/slides/GrMod3_2_17_2011-ann.pdf page 4

Same as Naïve Bayes, compute all parameters,

$$\theta_{S=1|F=1,A=0} = \frac{\sum_{train} \delta(S_k=1,F_k=1,A_k=0)}{\sum_{train} \delta(F_k=1,A_k=0)} \qquad \text{(GML-1)}$$

It is NP-hard, use dynamic programming.

- ### Partly Observed Data (EM)

Let $X =$ observed variables, $Z =$ unobserved variables. We would not be able to compute

$$\theta = \arg\max_{\theta} \log P(X, Z|\theta)$$

Instead we use EM

$$\theta = \arg\max_{\theta} \mathrm{E}_{P(Z|X,\theta)}[\log P(X, Z|\theta)]$$

E-step: use X and current θ to calculate $P(Z|X,\theta)$

M-step: replace current θ by

$$\theta := \arg\max_{\theta\prime} \mathrm{E}_{P(Z|X,\theta)}[\log P(X, Z|\theta')]$$

Suppose Sinus is unobserved.

Randomly pick initial parameters of the missing ones $\theta_{S|F,A}, \theta_{H|S}, \theta_{N|S}$) then we can compute

$$P(F, A, S, H, N)$$

Then compute, E-step

$$P(S = s|F, A, H, N, \theta) = \frac{P(F, A, S = s, H, N)}{\sum_s P(F, A, S, H, N)}$$

Then M-step

$$\mathrm{E}_{P(Z|X,\theta)}[\log P(X, Z|\theta')] = \sum_{train}^{m} \sum_{s=\{0,1\}} P(S = s|F, A, H, N, \theta) \log P(F, A, S = s, H, N|\theta')$$

Compare this MLE with previous (GML-1), we get

$$\theta_{S=1|F=1,A=0} = \frac{\sum_{train} \delta(F_k = 1, A_k = 0) P(S = 1|F, A, H, N, \theta)}{\sum_{train} \delta(F_k = 1, A_k = 0)}$$

Repeat till converges.

• Learn Directed Tree Graph Structure (Chow-Liu Algorithm)

Tree: every node has only one immediate parent.

How to compare which tree is better? The one minimizes Kullback-Leibler divergence from $T(X)$, tree-structured network, to $P(X)$, total distribution,

$$KL(P(X)||T(X)) = \sum_k P(X = k) \log \frac{P(X = k)}{T(X = k)}$$

Proof: Let g be graph structure

$$\arg\max_{g} \log P(X|\theta, g)$$

We have

$$\log P(X|\theta, g) = \sum_{X_i \in \{nodes\}} \sum_{x, x' \in \{possible\ values\}} \log P(X_i = x|parents\ of\ X_i = x')$$

$$\underbrace{=> \sim}_{\substack{use \\ expectation}} \sum_{X_i \in \{nodes\}} \underbrace{\sum_{x, x' \in \{possible\ values\}} P(x, x') \log P(X_i = x|parents\ of\ X_i = x')}_{-H(X_i = x|parents\ of\ X_i = x')}$$

Recall conditional entropy

$$H(Y|X) = \sum_x P(x) H(Y|X = x) = -\sum_x P(x) \sum_y P(y|x) \log P(y|x) = -\sum_{x,y} P(x, y) \log P(y|x)$$

and mutual information

$$\begin{aligned} I(X,Y) &= H(Y) - H(Y|X) \\ &= -\sum_{x,y} P(x,y)\log P(y) + \sum_{x,y} P(x,y)\log P(y|x) = \sum_{x,y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \end{aligned}$$

Thus $\arg\max_{\mathscr{g}} \log P(X|\theta,\mathscr{g})$ becomes

$$\arg\max_{\mathscr{g}} \log P(X|\theta,\mathscr{g}) = \arg\max_{\mathscr{g}} \sum_{X_i\in\{nodes\}} \underbrace{-H(X_i|parents\ of\ X_i)}_{I(X_i,parents\ of\ X_i)-H(X_i)}$$

Since $H(X_i)$ is the entropy of all nodes independent of graphical structure, we arrived KL convergence. QED

Chow-Liu Algorithm

- Compute mutual information for all nodes (as if it is a complete graph)

$$I(X_i,X_j) = \sum_{x_i,x_j} P(x_i,x_j)\log\frac{P(x_i,x_j)}{P(x_i)P(x_j)}$$

- then use greedy algorithms (Kruskal’s, prim's, …) to find the maximal spanning undirected tree weighted by $I(X_i,X_j)$

- pick any node as the root, breath first search gives directions.

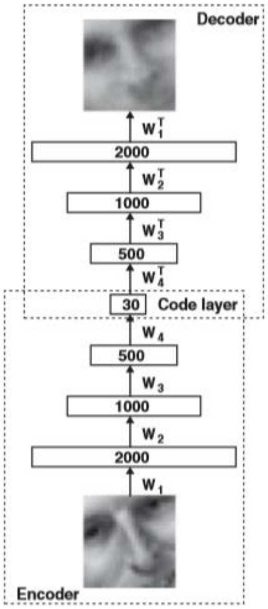
- to avoid overfitting use BIC to delete unnecessary edges.

Deep Learning and Reinforced Learning

Deep learning overview Yann LeCun, Yoshua Bengio & Geoffrey Hinton, http://nature.com/nature/journal/v521/n7553/full/nature14539.html

Joan Bruna Estrach, Stat212b: Topics Course on Deep Learning, Berkeley http://joanbruna.github.io/stat212b/

21. Deep Belief Network



Hinton & Salakhutdinov, 2006 Science http://www.cs.toronto.edu/~hinton/science.pdf

Algorithm

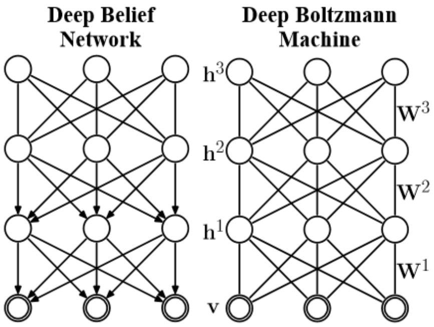
- train W_1 until 2000-layer resemble good image.

- train W_2 until 1000-layer resemble good 2000-layer data.

- train W_3 until 500-layer resemble good 1000-layer data.

- train W_4 until 30-layer resemble good 500-layer data.

- Deep Boltzmann Machines (undirected)



Hinton & Salakhutdinov http://www.cs.toronto.edu/~fritz/absps/dbm.pdf

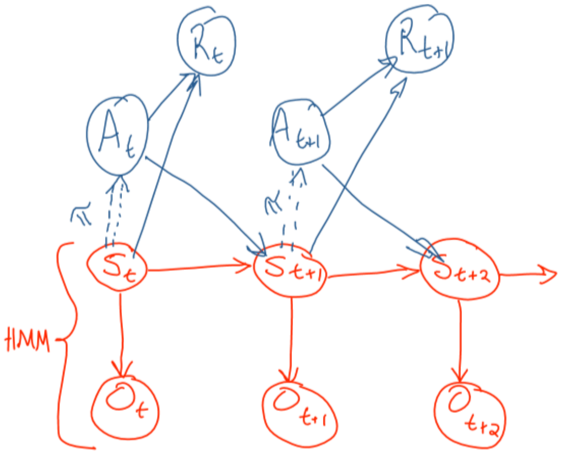
- Regulation

- Use the logistic regulation $||W||^2$

- Dropout method

For each training section, randomly drop half of the hidden connections. In this way every random set of half of the activated neutrons are able to produce correct outputs. So when the full neural network in dealing with test data, it is prone to greater level of accuracy and less overfitage.

22. (Partially Observed) Markov Decision Process (POMDP/MDP)



Picture from http://www.cs.cmu.edu/~tom/10701_sp11/slides/MDPs_RL_04_26_2011-ann.pdf page 4

If states S_t are fully observed, $O_t = S_t$. That differentiates POMDP and MDP

We suppose there is a mapping from possible states to possible actions

$$\pi: S \to A$$

i.e. for each $s \in S$, what action it will take is deterministic. But what state it will end up is not deterministic. It is given by

$$P(S_{t+1} = s' | S_t = s, A_t = a)$$

There is a rewards function $R: S \times A \to \mathbb{R}$.

For now let's assume MDPs.

Our goal is to maximize

$$\mathbb{E}[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots]$$

Discount constant γ helps to make the series converges and it means to reach high rewards in the small number of steps.

- ### Bellman Equations

Define value function

$$V^{\pi}(s) = \mathbb{E}[R(s,a_0) + \gamma R(s_1,a_1) + \gamma^2 R(s_2,a_2) + \cdots |S_0 = s, \pi]$$

Then we have Bellman equations

$$V^{\pi}(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} P\big(s' \big| s, \pi(s)\big) V^{\pi}(s')$$

which is a system of linear equations if $|S| < \infty$, can be solved by matrix inversion, but what is the optimal π?

- ### Value Iteration

Continue our goal to find best value

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

and optimal policy π^*

$$\pi^*(s) = \arg\max_{a \in A} \underbrace{\sum_{s' \in S} P(s'|s,a) V^*(s')}_{\mathbb{E}_{s'|s,a}[V^*(s\prime)]}$$

Above looks a lot like EM. So does the algorithm.

- Initialize $V(s)$

- loop through S, A, update

$$V(s) = \max_a \left(\underbrace{r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s')}_{\hat{Q}(s,a)} \right)$$

- repeat til converges and we get $V^*(s)$, and

$$V^*(s) = \max_a Q(s,a)$$

- ### Q-Learning-Model Free

If we don't know $P(S_{t+1}|S_t, A_t)$, we formally define Q-function

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}(V^*(s'))$$

Thus

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a}\left(\max_{a'} Q(s',a')\right)$$

Q-learning for deterministic world

We have $P(s'|s,a) = \delta(s')$

$$Q(S_t = s, A_t = a) = r(S_t = s, A_t = a) + \gamma \max_{a'} Q(S_{t+1} = s', A_{t+1} = a')$$

- initial table $Q(s,a)$ for each s, a

- loop over all actions available from state s, follow the action to state s', then update $Q(s,a)$.

- repeat

Q-learning for non-deterministic world

It is not as easy as it sounds to compute

$$P(s'|s,a) = \frac{\#times\ we\ took\ action\ a\ in\ state\ s\ and\ got\ to\ s\prime}{\#times\ we\ took\ action\ a\ in\ state\ s} \qquad \text{(QPP)}$$

Instead we bind them together

$$Q(s,a) \leftarrow (1-\alpha_n)Q(s,a) + \alpha_n\left[r(s,a) + \gamma \max_{a'} Q(s',a')\right]$$

Where learning rate $\alpha_n = \frac{1}{1+n(\#visit\ s,a)}$. Hence first time update $Q(s,a)$, $\alpha_1 = 1/2$, second time $\alpha_2 = 1/3$, and etc.

Since Q-learning converges slower than value iteration does, Dyna-Q algorithm

Richard Sutton https://webdocs.cs.ualberta.ca/~sutton/papers/sutton-90.pdf

- apply Q-learning to the real world

- use (QPP) to find the approximate world from the previous Q-learning experiences, apply value iteration, update Q

- ### Example: Q-learning Stock Automatic Trading Machine

- initial random Q table

- update Q table till converge

$$Q(s,a) \leftarrow (1-\alpha_n)Q(s,a) + \alpha_n \left[r(s,a) + \gamma \max_{a'} Q(s',a')\right]$$

$s =$ relative price, relative volume, normalized deviation from mean, … other indicators. Discretize them

$a =$ buy, sell, do nothing

$r =$ return.

Short term return result faster convergence than long term (cumulative) return. At the beginning $\max_{a'}$. is not strictly enforced.

- Temporal Difference Learning

Back to Q-learning for deterministic world

Of course the following Bellman equations are also true if we go 2-step ahead

$$V^{\pi}(s) = R(s,\pi(s)) + \gamma R(s_{t+1},\pi(s_{t+1})) + \gamma^2 \sum_{s'\in S} P\big(s'\big|s,\pi(s)\big)V^{\pi}(s')$$

Hence we change

$$Q^{(1)}(s,a) = r(s,a) + \gamma \max_{a'} Q(s'_{t+1},a'_{t+1})$$

to

$$Q^{(2)}(s,a) = r(s,a) + \gamma r(s_{t+1},a_{t+1}) + \gamma^2 \max_{a'} Q(s'_{t+2},a'_{t+2})$$

and $Q^{(3)}$ etc.

We can blend all of these, using $(1-\lambda)(1+\lambda+\lambda^2+\cdots) = 1, 0 < \lambda < 1,$

update

$$Q^{\lambda}(s,a) = (1-\lambda)\big(Q^{(1)}(s,a) + \lambda Q^{(2)}(s,a) + \lambda^2 Q^{(3)}(s,a) + \cdots\big)$$

$$= r(s,a) + \gamma\left[(1-\lambda)\max_{a'} Q(s,a') + \lambda Q^{\lambda}(s_{t+1},a_{t+1})\right]$$

This turns out to converge faster than $Q^{(1)}(s,a)$.

Machine Learning in Action: Artificial Intelligence

23.Computer Can Read: Natural Language Processing

- Regular Expression

^ starting of line, $ end of line, [Ll][Oo][Vv][Ee], ^[^a] not start with ‘a’, ‘.’ any character except \n (\. for true .), | or, ( )? optional, * repeat (it’s greedy), + at least one, {m,n} match >=m <n times, \n match text, \\d digit, \\D not digit, \\s space, \\S not space, \\w word, \\W not word, \\t tab, \\n new line, i+ at least one time, i* zero or more times, i? zere or 1 time, [:alnum:] alphanumeric, [:alpha:] and [:digit:], [:blank:], [:cntrl:], [:graph:], [:lower:], [:upper:], [:print:], [:punct:], [:space:], [:xdigit:]

```
library(stringer)
exContract <- "Monday 08/01/16 Economist View"
str_extract(exContract, "[\\d][\\d][/][\\d][\\d][/][\\d][\\d]")
#return "08/01/16"
```

- Twitter Stocks Sentiments

http://thinktostart.com/sentiment-analysis-on-twitter/

24.Computer Can Think: Neural Network

- Neural Net Package

https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/

- Deep Generative Models

http://www.cs.princeton.edu/~blei/papers/HoffmanBleiWangPaisley2013.pdf

- Convolution Neural Network (CNN)

Divide image into region(pitch) to recognize items and apply statistics invariance

LeNet http://yann.lecun.com/exdb/lenet

- Recurrent Neural Network(RNN)

Remember the past

25.Computer Can Talk: Voice Recognition

- Speech API

Google http://cloud.google.com/speech/

Amazon Alexa http://developer.amazon.com/alexa-voice-service

Apple Siri http://developer.apple.com/sirikit

26.Computer Can See: Computer Vision

- ### Understand Picture API

Microsoft http://www.captionbot.ai

- ### Virtual Reality

Facebook Oculus http://oculus.com

Microsoft Hololens https://www.microsoft.com/microsoft-hololens

Google Cardboard http://vr.google.com/cardboard

Reference

27.Books

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *An Introduction to Statistical Learning with applications in R* (with R code), Springer-Verlag, 2013
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, *Elements of Statistical Learning: data mining, inference and prediction* 2nd ed , Springer-Verlag, 2009
- Christopher Bishop, *Pattern Recognition and Machine Learning* (with Matlab code), Microsoft 2006
- Kevin Murphy, *Machine Learning, A Probabilistic Perspective* (with Code), MIT 2012
- Martin Wainwright, Michael Jordan, *Graphical Models, Exponential Families, and Variational Inference*, Unpublished, Berkeley
- Daniel Jurafsky, James Martin, *Speech and Language Processing* (draft 3rd ed), Prentice Hall; 2nd edition 2008
- Christopher Manning, Hinrich Schuetze, *Foundations of Statistical Natural Language Processing* (free online), MIT press 1999

28.Packages

- http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
  Andreas Muller, NYU, Scikit-Learn Maintaining manager, Scikit Tutorial, Open Source
- http://torch.ch/ CPU friendly Deep Learning, NYU
- http://www.tensorflow.org Google deep learning
  Udacity course https://www.udacity.com/course/deep-learning--ud730
- http://deepmind.com Google deep reinforcement learning

- http://www.cntk.ai Microsoft deep learning
- http://www.mathworks.com/help/stats/index.html
- http://cran.r-project.org/web/views/MachineLearning.html

29. Courses

- Aarti Singh, Tom Mitchell, *Machine Learning 701* (with Note & Video), Carnegie Mellon University
- David Sontag, *Introduction To Machine Learning*, New York University
- Mehryar Mohri, *Foundations of Machine Learning*, New York University
- Mehryar Mohri, *Advanced Machine Learning*, New York University, Google Research
- Terence Tao, *Topics in Random Matrix Theory* (with Note), UCLA
- Yann LeCun, *Deep Learning* (with video, note), New York University, Facebook AI
- John Langford, Yann LeCun, *Big Data Learning* (with video, note), New York University
- Joan Bruna Estrach, *Topics Course on Deep Learning*, Berkeley
- Fei-Fei Li, Andrej Karpathy, Justin Johnson, *Convolutional Neural Networks for Visual Recognition*, Stanford University
- Fei-Fei Li (Stanford), Rob Fergus (NYU), Antonio Torralba (MIT), *Recognizing and Learning Object Categories* (computer vision)
- Eric Xing, *Probabilistic Graphical Models* (Note&Video), Carnegie Mellon University

30. On-line Course

- Tucker Balch, *Machine Learning for Trading*, Georgia Tech, Udacity
- Andrew Ng, *Machine Learning* (with notes), Stanford, Coursera
- Daphne Koller, Kevin Murphy, *Probabilistic Graphical Models* (with notes, video), Stanford, Coursera
- Dan Jurafsky, Christopher Manning, *Natural Language Processing* (with slide), Stanford, Coursera
- Michael Collins, *Natural Language Processing* (with notes), Columbia University, Coursera

31. Data

- https://www.google.com/publicdata/directory
- https://aws.amazon.com/datasets/
- https://www.data.gov/
- http://factfinder.census.gov/
- https://nycopendata.socrata.com/
- http://www.sec.gov/edgar.shtml
- http://bea.gov
- https://www.irs.gov/uac/tax-stats
- http://data.worldbank.org/
- http://developer.trulia.com/

- https://developer.foursquare.com/
- http://www.image-net.org/

32. Competition & Conference

- International Conference on Machine Learning
- http://deeplearning.net
- https://www.kaggle.com
- Machine Learning Summer Schools
- NYU and Columbia Hackathon