Statistical learning Theory & (Rakhlin - UPenn)

☑ Handwritten Digit Recognition

mnist dataset → great variability in data → current accuracy ≃ 100%

Solutions → hard code → hard to capture variability

→ ML → feature maps → intensity!
curvatures
...

→ labels → 

☑ Predict Topic of a News Article

filter news by topics

Solutions → hard code

→ ML → feature maps → bag of words

* Why ML? → impossible to hard-code
→ adaptive to changes in environment

- Learning: process of acquiring knowledge or skill

↳ better: design & analysis of algorithms that improve performance after observing data

↳ conclude something general from specific example → ill-posed problem

↓

many alternatives that consistent with observations

↓

need prior knowledge

☑ به رستوران می‌ریم و غذا خوب نیست ← غذای خارجی نیست
رستوران بد است.
...

↳ induction (≠ deduction)

↳ Memorization is not learning nor reduction

- Data

↳ Needs domain knowledge ⟹ feature map Space of inputs

↳ Space of Outputs or responses

- Frameworks

- Supervised l.
- Unsupervised l.
- Semi-supervised l. → mix of above

↳ labeling is expensive process

- Labels :
  - Binary
  - Multiclass
  - Regression
  - Structure prediction ⟶ ☑ graph, sentence
- Protocol :
  - Active
  - Passive
- Data Assumption
  - Batch ⟶ typically iid
  - Online ⟶ iid / worst case / stochastic

* Why Theory

- Theory ♡ Practice

* Statistical Learning Theory (STL)

- Statistical
  - x related to y ⟶ learn relationship ⟶ distribution $\mathbb{P}$ (population)
  - we have access to samples of dist ⟶ iid : all data from dist
    independant sampling

☑ Not good for spam filtering

☑ Fair for hand writing recognition

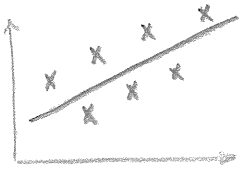
- $\mathbb{P} = \mathbb{P}_x \times \mathbb{P}_{y|x}$
  ↳ marginal ↳ conditional
- $\hat{f}_n : X \longmapsto Y$ mapping from dataset to function
  ↳ relation we wanted to learn
- ℓ : loss function $\ell : Y \times Y \longmapsto \mathbb{R}$ measure of performance
- We're gonna average on all data, what my performance is

$$\mathbb{E}_{(X,Y)}\left\{\ell\left(\hat{f}_n(X), Y\right)\right\}$$ ⟶ Expectation of loss

- Loss function
  - Indicator Loss ⟶ are they equal or not ⟶ $\{0,1\}$
  - Square loss ⟶ $(y-y')^2$ ⟶ regression cases
  - Absolute loss ⟶ $|y-y'|^2$

☑ linear least squares

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle \beta, x_i \rangle)^2 \longrightarrow \text{regression}$$



$$\hat{f}_n(x) = \langle \hat{\beta}, x \rangle$$

☑ regularized least squares

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - \langle \beta, x_i \rangle)^2 + \lambda \|\beta\|^2$$

- So far we have nothing to do with problem (distribution)

Methods vs. Problems Algorithms $\hat{f}_n$ ⟶ Dist P

- No best algorithms for all problems ⟶ No free lunch
- Expected Loss : for any function $f : X \longmapsto Y$

$$L(f) = \mathbb{E}\, \ell(f(X), Y)$$

P unknown ⟶ $L(f) = ?$

P known ⟶ problem solved!

- Empirical loss : for any function

$$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

$\hat{L}$ ⟶ related to data

- what is Random here?

data are random i.i.d draw from P ⟹ Any function of data is random variable too →

- $\hat{L}(f)$ ⟶ r.v.
- $\hat{f}_n$ ⟶ r.v. output of learning after seeing dat
- $L(\hat{f}_n)$ ⟶ r.v.
- for a given $f : X \longmapsto Y$ $L(f)$ is not not random!

- Gold Standard
  - smallest expected loss
    ↳ not unique

$$f^* = \arg\min_{f} L(f) \longrightarrow \text{Bayes Optimal}$$

- Bayes Error $L(f^*) = \inf_f L(f)$

↳ P unknown → we cannot calculate

- We usually don't know the form of Bayes optimal function f^*

But two cases:

↳ Binary Classification $f^*(x) = \mathbb{I}_{\{\eta(x) \geq \frac{1}{2}\}}$ $\eta(x) = \mathbb{E}[Y|X=x]$

↳ Regression $f^*(x) = \eta(x)$ $\eta(x) = \mathbb{E}[Y|Y=x]$

P: unknow → value of f^* unknow but we know the form

- Start of STh → Can we garantee $L(\hat{f}_n) - L(f^*)$ to be small if n is large?

* Consistency

- Consistent : an algorithm that ensure

$$\lim_{n \to \infty} L(\hat{f}_n) = L(f^*)$$

- Good news: possible to achieve → easy if X is finite / countable set

↘ not too hard if X is infinite and relation of X & Y is continue

- There are always Bad news! → we can't prove any thing about $L(\hat{f}_n) - L(f^*)$

↳ need prior knowledge

↘ No free Lunch Theorem

* No free Lunch Theorem : if we don't have further assumptions

- for any alg $\hat{f}_n$ → P exist with $L(f^*) = 0$

$$\mathbb{E}\, L(\hat{f}_n) \geq \frac{1}{2} - \varepsilon$$

if we don't give any assumption, we can't perfor better than a coin tossing!

- for any alg $\hat{f}_n$ → P exist with $L(f^*) = 0$

$$\lim_{n \to \infty} a_n = 0$$

$$\mathbb{E}\, L(\hat{f}_n) \geq a_n$$

Means ⟹

we can't say how long we should wait for alg to perform good enough.

- Is it really bad news?
  - we always have domain knowledge
  - Two ways of incorporating knowledge
    - ↳ Direct knowledge → Modeling or Generative approach → we guess the distribution
    - ↳ Indirect knowledge → Redefine the goal (Statistics)

$$L(\hat{f}_n) - \inf_{f \in F} L(f)$$

  * Started by Vapnik → discriminitive approach
  * F encapsulates our inductive bias

↳ Cons / Pros → Slide 33

↘ we should know both → sometimes one of them is better

☑ Linear Discriminant Analysis

$\Sigma = I$ →

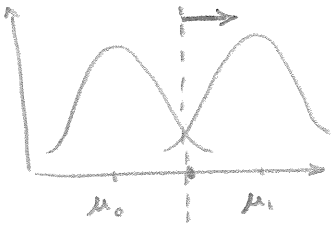
$$p(x|y=0) = (2\pi)^{-k/2} \exp\left\{-\frac{1}{2}\|x-\mu_0\|^2\right\}$$

$$p(x|y=1) = (2\pi)^{-k/2} \exp\left\{-\frac{1}{2}\|x-\mu_1\|^2\right\}$$

⇒ μ_0, μ_1 : unknown

Best Classifier $f^* = I_{p(y=1|x) \geq 1/2}$ → linear : $p(x|y=1) \geq p(x|y=0)$

↳ Bayes



$\Sigma_1 \neq \Sigma_2$ → Quadratic Discr. Anal.

$I\{q(x) \geq 0\}$

$q(x)$ ~ quadratic function

* Bias - Variance Trade off
  - Choosing a model (assumption) → Bias
  - How do we choose F?

$$L(\hat{f}_n) - L(f^*) = \underbrace{L(\hat{f}_n) - \inf_{f \in F} L(f)}_{\text{Est. Err}} + \underbrace{\inf_{f \in F} L(f) - L(f^*)}_{\text{Approx. Err}}$$

  - F larger → smaller approx err
  - n ↑ → smaller estimation err. & no effect on approximation err.

- Trade of F and n ⟶ Structural Risk Minimization or Model Selection

• if we guessed f^* so $f^* \in F$ ⟶ $L(\hat{f}_n) - L(f^*) = L(\hat{f}_n) - \inf_{f \in F} L(f)$

* Occum's Razor

• Principle for choosing the simplest theory or explanation

• Help to try understanding the complexity

• we look for alg $\hat{f}_n$ that with n samples have small $L(\hat{f}_n) - \inf_{f \in F} L(f)$

↳ smaller than desired accuracy

↳ with fixed set of F ⟶ rich enough

• Bounds

- In Expectation $\quad \mathbb{E}\left\{ L(\hat{f}_n) - \inf_{f \in F} L(f) \right\}$

- In Probability $\quad \mathbb{P}\left\{ L(\hat{f}_n) - \inf_{f \in F} L(f) \geq \varepsilon \right) < \psi(n, \varepsilon)$

↳ control the tail

or $\quad \mathbb{P}\left(L(\hat{f}_n) - \inf_{f \in F} L(f) \geq \Phi(\delta, n) \right) < \delta$

$\delta := \psi(n, \varepsilon)$ ⟶ solving for ε

$\Phi(\delta, n)$ ⟶ we want in this lecture ⟶ "the rate"

- Sample Complexity ⟶ what is the # of examples algorithm need to reach ε accuracy with "$1-\delta$" probability

$$\mathbb{P}\left(L(\hat{f}_n) - \inf_{f \in F} L(f) \geq \Phi(\delta, n) \right) < \delta$$

$\varepsilon := \Phi(\delta, n)$ ⟶ solve for n

$$\mathbb{P}\left(L(\hat{f}_n) - \inf_{f \in F} L(f) \geq \varepsilon \right) \leq \delta \qquad n \geq n(\varepsilon, \delta)$$

$n \downarrow \Rightarrow$ faster rate to reach ε-guarantee

$O\left(\frac{1}{\sqrt{n}}\right) \Rightarrow O\left(\frac{1}{\varepsilon^2}\right)$, $O\left(\frac{1}{n}\right) \Rightarrow O\left(\frac{1}{\varepsilon}\right)$

* Clarify

Slide 44

(A)

(B) To guarantee the performance of $\hat{f}$

(C)

(D) Look at function and difference to empiric data ⟶ which one is better

(E) Upper bound for (B) Model Selection ⟶ grow F

* Bounds

- $L(\hat{f}_n) - \hat{L}(\hat{f}_n) \leq \sup_{f \in F} \{ L(f) - \hat{L}(f) \}$ $\Rightarrow$ (D) implies a bound on (C)

- Bound on (B)? $\rightarrow$ (C) or (D)? depends on what the alg does

- ERM: Empirical Risk Minimization $\quad \hat{f}_n = \arg\min_{f \in F} \hat{L}(f)$

- Regularized ERM $\quad \hat{f}_n = \arg\min_{f \in F} \hat{L}(f) + pen_n(f)$

- $f_F = \inf_{f \in F} L(f) \rightarrow$ one of the best choices for f

$$L(\hat{f}_n) - L(f_F) \leq \underbrace{\{ L(\hat{f}_n) - \hat{L}(\hat{f}_n) \}}_{(C)} + \underbrace{\{ \hat{L}(\hat{f}_n) - \hat{L}(f_F) \}}_{\leq 0} + \underbrace{\{ \hat{L}(f_F) - L(f_F) \}}_{\circledast}$$

$$\leq (C) + \circledast$$

- Hoeffding Inequality

$\omega_i \rightarrow$ i.i.d

$P(a \leq \omega \leq b) = 1$

$$\Rightarrow \mathbb{P}\left(\mathbb{E}\omega - \frac{1}{n} \sum_{i=1}^{n} \omega_i > \varepsilon \right) \leq \exp\left(-\frac{2n\varepsilon^2}{(b-a)^2} \right)$$

gaussian tails are enforced by $\varepsilon \Rightarrow \varepsilon \uparrow \rightsquigarrow$ more gaussian tails

using centeral limit theorem

$$\mathbb{P}\left(\frac{1}{n} \sum_{i=1}^{n} \omega_i - \mathbb{E}\omega > \varepsilon \right) \leq \exp\left(-\frac{2n\varepsilon^2}{(b-a)^2} \right)$$

$$\Rightarrow \mathbb{P}\left(|L(f_F) - \hat{L}(f_F)| > \varepsilon \right) \leq 2\exp\left(-\frac{2n\varepsilon^2}{(b-a)^2} \right)$$

$$a \leq \ell(f_F(x), y) \leq b$$

- $\mathbb{P}\left(\underbrace{\mathbb{E}L(\hat{f}_n(x,y))}_{} - \underbrace{\frac{1}{n} \sum \ell(\hat{f}_n(x_i), y_i)}_{\text{empirical}} > \varepsilon \right) \leq 2\exp(\sim)$

$\ell(\hat{f}_n(x_i, y_i)) \rightarrow$ memory, test on same training set $\rightarrow$ cannot generalize

$\rightarrow$ biased estimate of $\mathbb{E}\,\ell(\hat{f}_n(x), y) \rightarrow$ overfitting

out of sample performance: $\mathbb{E}\omega = \mathbb{E}\,\ell(\hat{f}_n(x), y)$

in sample perf.: $\mathbb{E}\omega_i = \mathbb{E}\,\ell(\hat{f}_n(x_i), y_i)$

$\neq \Rightarrow$ gap $\rightsquigarrow$ overfitting

- ERM f_n → memory, perfect recall of data → cannot generalize

$$\sup_{f \in F} \{ L(f) - \hat{L}(f) \} = 1 \longrightarrow (D) : \text{called uniform deviation}$$

1 → cannot generalize ; 0 = perfect fit

* Uniform Deviation

- $$\sup_{f \in F} \left\{ \mathbb{E}_{x,y}\, \ell(f(x), y) - \frac{1}{n} \sum \ell(f(x_i), y_i) \right\}$$

$F = \{f_0\}$ → have good bound, always answer the same

F = rich → maybe has bad bound

- $z_i = (x_i, y_i)$ → training pair

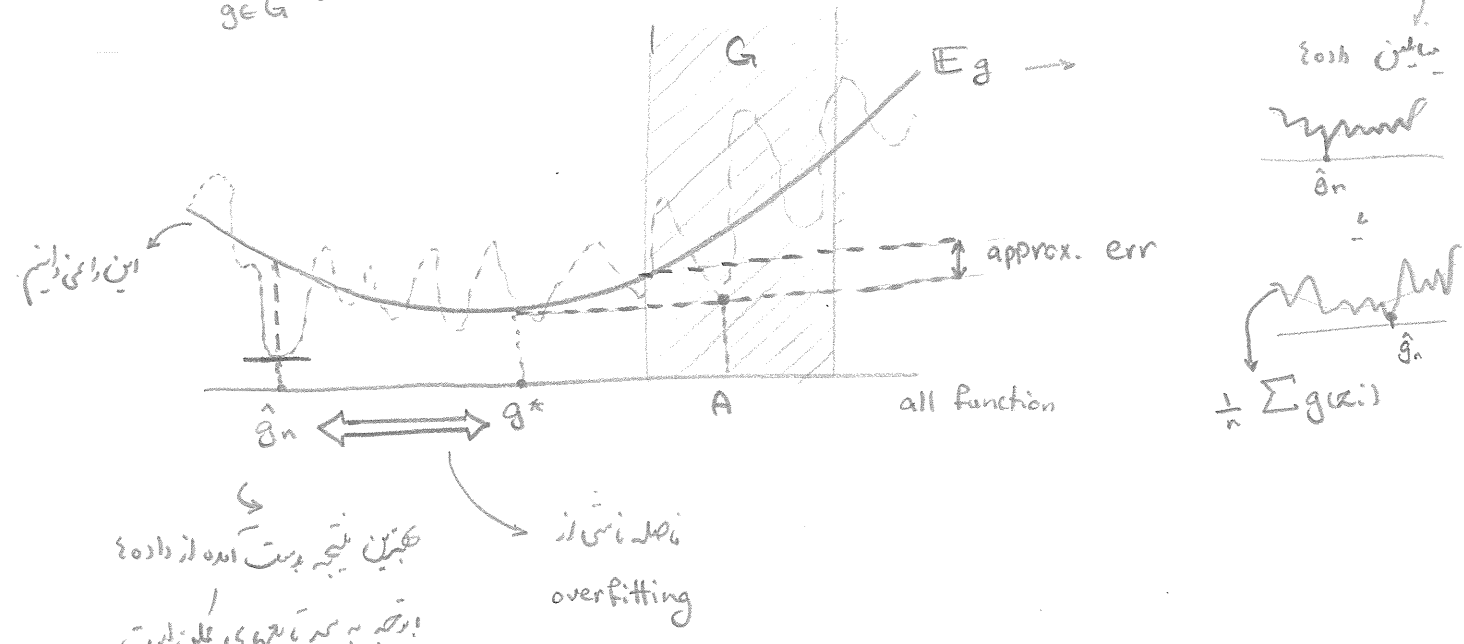
$g(z) = \ell(f(x), y)$

$\hat{g}_n = \ell(\hat{f}_n(\cdot), \cdot)$

$g_G = \ell(f_F(\cdot), \cdot)$

$g^* = \underset{g}{\operatorname{argmin}}\ \mathbb{E}\, g(z) = \ell(f^*(\cdot), \cdot)$ → Bayes optimal loss function → best overall function

$$\circledast = \sup_{g \in G} \left\{ \mathbb{E}\, g(z) - \frac{1}{n} \sum g(z_i) \right\}$$



بهترین نتیجه بدست آمده از داده‌ها

برتری به همه تابعهای ممکن است

فاصله ناشی از overfitting

$\hat{g}_n$ ← نقطه‌ای «A» متعلق می‌شود (در نمونه‌های دیده نشده) ← می‌خواهیم با اندازه

اگر پنجره G را کوچک کنیم

کافی G را کوچک کنیم.

- Stochastic Process : collection of random var. index by some set.

  - Empirical Procces ~ stochastic process by function G

  - Uniform Law of large Number

$$\sup_{g \in G} \left| \mathbb{E} g - \frac{1}{n} \sum g(z_i) \right| \xrightarrow{n \to \infty} \infty$$

  - question : How big can "G" be ?

  - reminder : Boole's Inequality $P(\bigcup A_j) \leq \sum P(A_j)$

$$P(\exists g \in G : \mathbb{E} g - \frac{1}{n} \sum g(z_i) > \varepsilon) \leq \sum_j P(\mathbb{E} g_j - \frac{1}{n} \sum_i g_j(z_i) > \varepsilon)$$

$$\begin{cases} G = \{g_1, g_2, \cdots, g_N\} \rightarrow \text{\# of function} \\ P(a \leq g(z_i) \leq b) = 1 \end{cases}$$

$$P($$

  - How many example do we need?

$$P\left(\sup_{g \in G} \left\{ \mathbb{E} g - \frac{1}{n} \sum g(z_i) \right\} > (b-a) \sqrt{\frac{\log N + \log(1/\delta)}{2n}} \right) \leq \delta$$

--- slide 56

- Message
  - with prob at least $1-\delta$ for any $g \in G \Rightarrow \mathbb{E} g - \frac{1}{n} \sum g(z_i) \leq \varepsilon$
  - for any $g \in G$ with prob at least $1-\delta \Rightarrow \mathbb{E} g - \frac{1}{n} \sum g(z_i) \leq \varepsilon$

- weighted Union Bound : Countable Class

$$\sum_{g \in G} \omega(g) \leq 1$$

$$L(f) - \hat{L}(f) \leq \underbrace{(b-a) \sqrt{\frac{\log \frac{1}{\omega(f)} + \log \frac{1}{\delta}}{2n}}}_{pen_n(f)}$$

$$L(\hat{f}_n) - L(f_F) \leq \{L(\hat{f}_n) - \hat{L}(f_n) - pen_n(f_n)\} + \{\hat{L}(\hat{f}_n) + pen_n(\hat{f}_n) - \hat{L}(f_F) - pen_n(f_F)\} \hookrightarrow \leq 0 + \{\hat{L}(f_F) - L(f_F)\} + pen_n(f_F)$$

$$\leq \sup_{f \in F} \{L(f) - \hat{L}(f) - pen_n(f)\} + \{\hat{L}(f_F) - L(f_F)\} + pen_n(f_F)$$

$$L(\hat{f}_n) - L(f_F) \leq \{\hat{L}(f_F) - L(f_F)\} + pen_n(f_F)$$

$$\leq 2(b-a)\sqrt{\frac{\log \frac{1}{\omega(f_F)} + \log \frac{1}{\delta}}{2n}}$$

- Uncountable Class: Compression Bound

algorithm $\hat{f}_n$ depend on dataset $S \longrightarrow \hat{f}_n = \hat{f}_n[S]$

C_k select a subset of useful examples $\longrightarrow \hat{f}_n[S]$, $\hat{f}_k[C_k(S)]$ $\longrightarrow$ not general

$$L(\hat{f}_n) - \hat{L}(\hat{f}_n) = \mathbb{E}\, \ell(\hat{f}_k[C_k(S)](x), y) - \frac{1}{n}\sum \hat{f}_n[C_k(S)](x_i), y_i$$

$\leq$ max slide 61

most of data are now out of samples

$$L(\hat{f}_n) - \inf_{f \in F} L(f) \leq 2(b-a)\sqrt{\frac{k \log(\frac{en}{k}) + \log \frac{1}{\delta}}{2n}} + 2\frac{(b-a)k}{n}$$

$F_{k,n}$ has cardinality at most $\binom{n}{k} \leq (\frac{en}{k})^k$

* Stability

- change of a single example don't change the algorithm too much

$$\hat{f}_n = \arg\min \frac{1}{n}\sum (f(x_i) - y_i)^2 + \lambda \|f\|_k^2$$

↳ stable $\longrightarrow$ Reproducing kernel Hilbert Space

* Uniform Convergance

$z'_1, \dots z'_n \longrightarrow$ iid from P

$\epsilon_1, \dots \epsilon_n \longrightarrow$ iid with $P(\epsilon_i = -1) = P(\epsilon_i = +1) = 1/2$ $\longrightarrow$ coin flip

Rademacher r.v.

$$\mathbb{E} \sup_{g \in G} \{\mathbb{E}g(z) - \frac{1}{n}\sum g(z_i)\} = \mathbb{E}_{z_{1:n}} \sup \{\frac{1}{n}\sum g(z'_i) - \frac{1}{n}\sum g(z_i)\}$$

comparison of empirical with model $\Longrightarrow$ comparison of empirical 1 to empirical 2

* Symmetrization

$$\mathbb{E}\, L(f_n) - \inf_{f\in F} L(f) \lesssim \mathbb{E} \sup_{f\in F} L(f) - \hat{L}(f) \leq 2 \underbrace{\mathop{\mathbb{E}}_{\text{data}} \mathop{\mathbb{E}}_{\epsilon_{1:n}} \sup_{g} \frac{1}{n} \sum \epsilon_i g(z_i)}_{\text{Rademacher Complexity Bound}}$$

data points z_i $\longrightarrow$ { function 1 : $f(z_1), \dots f(z_n)$; function 2 : $f'(z_1), \dots, f'(z_n)$; ... } $\Big\}$ $F|_{x_{1:n}} \subseteq \{0,1\}^n$

↳ subset of nD space

$$v: \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \longrightarrow v^T \epsilon = \sum_i \epsilon_i v_i$$

☑ VC Classes

Half spaces $\longrightarrow$ half predict 0, half predict 1

* Large Margin Theory for Classification

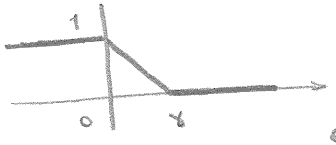
- $I(F) = \{ I_{\{f \neq 0\}} : f \in F \}$
- $f(x) = f_w(x) = \langle w, \psi(x) \rangle$ $\longrightarrow$ $\psi(x)$: kernel $\longrightarrow$ VC dimension is high
  - ↘ but SVM working on this basis work well

- Hard margin $\longrightarrow$ $\exists f \in F : \forall i,\ y_i f(x_i) \geq \gamma$

  $\exists f \in F : \frac{\text{misclassification}}{n}$ $\longrightarrow$ hope to be small

$\phi(y f(x))$

- Soft margin



$\longrightarrow$ penalize correct examples but near margin

$\Phi(y f(x))$

Surrogate loss

$$\mathbb{E} \sup_{f\in F} \{ L_\phi(f) - \hat{L}_\phi(f) \}$$

if ϕ is L-lipschitz $\longrightarrow$ $R_n(\phi(F))$ is bounded

* Properties of Rademacher Complexity

1.
2. Convex hull of function $\longrightarrow$ dont increase R
3. $\hat{R}_n(cF) = |c| \hat{R}(F)$
4.

- Easier to obtain sample complexity for Learning Tree, NN, ...
  ↳ using Rademacher complexity

* Sequential Prediction

- Motivation
  - Future look like the past
  - Get rid of iid

☑ www.seed.ucsd.edu/~mindreader → human cannot make a random sequence
we are finite state machines

↳ input : $y_1, y_2, \ldots \in \{0,1\}$ → one by one

↳ output : $z_1, z_2, \ldots$ → prediction

↳ goal → make average of correct predictions large

→ proportion of correction prediction is larger than proportion of 1's or 0's

→ make less mistakes than two predictors → { one always says 1 / one always says 0 }
experts
strategists

☑ Spam Detection

$x_1, \ldots x_n$ → emails, revealed one-by-one

→ not iid

→ spammers adapt to what you do → adversarial

- Supervised Learning
  - X input , Y output
  - Regret = you are close to some hypothesis (# of incorrect predictions)
    - if you use that f you suffer that much
  - Goal : keep regret small for any sequence of (x_i, y_i)

- Online Convex and Linear Optimization
  - loss function is convex
  - ☑ squared loss $\ell(f,(x,y)) = (\langle f,x\rangle - y)^2$
    hinge loss $\ell(f,(x,y)) = \max\{0, 1 - y\langle f,x\rangle\}$
  - $\text{Reg}_n = \frac{1}{n}\sum_t \ell(f_t, z_t) - \inf_{f\in F} \frac{1}{n}\sum_t \ell(f, z_t)$
  - Gradient Descent : $f'_{t+1} = f_t - \eta \nabla \ell(f_t, z_t)$
  - Alg. guarantee $\text{Reg}_n \leq O\left(\frac{1}{\sqrt{n}}\right)$
  - Function strongly convex → grow faster than quadratic function

- $\forall f^* \in F : \; Reg_n \leq O\left(\frac{\log n}{n}\right)$

- Use for i.i.d :

$Reg_n \leq R_n$ $\qquad$ $\mathbb{E}\, L(\bar{f}) - \inf_{f \in F} L(f) \leq R_n$

Distribution P ~ unknown

↳ Realize in « Pegasos » alg.

- Pegasos (= SVM in linear case)

$$\hat{f}_n = \arg\min_{f \in R^d} \frac{1}{m} \sum \max\{0, 1 - y_i \langle f, x_i \rangle\} + \frac{\lambda}{2} \|f\|^2$$

$\ell(f,z) = \max\{0, 1 - y\langle f, x_i \rangle\} + \frac{\lambda}{2}\|f\|^2$ ⟶ hinge loss func.

goal : $\min_f \mathbb{E}\, \ell(f,z)$

guarantee : $\mathbb{E}\, \ell(\bar{f},z) - \inf \mathbb{E}\, \ell(f,z) \leq O\left(\frac{\log n}{n}\right)$

alg : for $t = 1, \dots, n$

Choose a random example (x_{i_t}, y_{i_t})

set $\eta_t = \frac{1}{\lambda t}$

if $y_{i_t} \langle f_t, x_{i_t} \rangle < 1$

$$f_{t+1} = (1 - \eta_t \lambda) f_t + \eta_t x_{i_t} y_{i_t}$$

else

$$f_{t+1} = (1 - \eta_t \lambda) f_t$$

Statistical Relational learning : (Domingos – U of Washington)

* Motivation

- iid assumption → one type of object, no relation
- Real application → Multiple objects, heavily related

☑ web search, information extraction, NLP, perception, Medical Diagnosis, Social Net.

↓	↓	↓	↓	↓	↓
linked pages	partial overlap of info	Sinthetic parse	legs of chairs	epidemy	opinions
...	...	Symmantic	...	...	habits
		...			fashion

* Costs & Benefits

- Benefits → Better predictive Accuracy
  - Better understing of Domains
  - Growth path of ML
- Costs → Learning is much harder
  - ☑ iid → 100 features → vector of 100
  - SRL → 100 features → things has interaction
  - Inference becomes a Crucial issue
    - ↳ even harder problem than learning
  - Greater complexity for user

* Goal

- goal : learn from non-iid data as easy as iid ones
- progress → "close enough" to goal
  - burgeoning research area
  - Open Source, easy to use
- old & new questions

* Plan

- elements : probability, logic, learning and inference
- put them together
- lots of application

* Probablistic Inference : Markov Networks

- Undirected graphical model ⟶ structure of dependencies
- Potential functions defined over cliques ⟶ Markov Blancket

$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_c \prod_c \Phi_c(x_c)$$

$\Phi_c \geq 0$

↳ no restriction $\sum \Phi \neq 1$

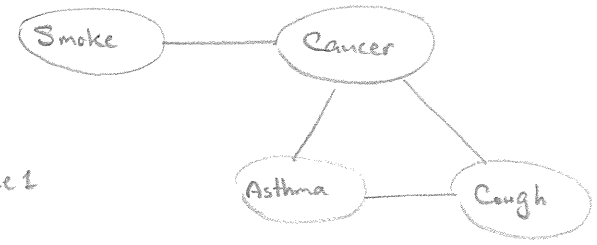
⟶ partition function : the most important part of Markov Net.

- Conditional independance advantage
- Size of Tables are exponential to size of cliques
- Log linear model

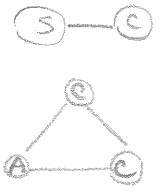
$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x)\right)$$

☑ $f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 0 \\ 1 \end{cases}$

$w_1 = 1.5$



Sm	Can	Φ
0	0	4.5
0	1	4.5
1	0	2.7
1	1	4.5



clique 1

clique 2

- Hammersley - Clifford Theorem
  - Distribution is strictly positive $P(x) > 0$ ⟹ Dist = $\prod$ potentials over cliques of graph
    Graph Encodes conditional potentials
  - Inverse ⟶ Markov Networks = Gibbs distribution is True
- MN vs. Bayesian Networks ⟶ Slide 12
  - ↳ Real world have cycles ⟶ MN ✓
  - ↳ Z is unknown ⟶ MN ☹
  - ↳ Indep check ⟶ MN is easier ✓ ⟶ easier to interpret
- MAP / MPE Inference ↳ Most Probable Explanation
  - Goal : find most likely state of world given evidence

$$\max_y P(y \mid x)$$

query, evidence

  - Iterated conditional modes ⟶ ☺ fast
    ↘ ☹ sensitive to local optima

- Simulate annealing → 😐 very slow
  → 🙂 for large iteration, it find the global min
- Graph Cuts
- Belief Propagation (max-product)

* Statistical Learning

- Learning MN → learning params → generatively
  → discriminatively
  → learning structure (features)
- Generative weight learning
  - Convex problem → no local maxima
  - numerical optimization → gradient or 2nd order

$$\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)]$$

$n_i(x)$: # of times feature i is true in data

$E_w[n_i(x)]$: expected # of times feature i is true according to model

underestimate: model < real → $\frac{\partial}{\partial w_i} > 0$ → $f_i \downarrow$

  - requires inference at each step → too slow!
- Psudo likelihood → if the problem you have is too hard, solve an easy one!
  - Fake likelihood → easy to compute

$$PL(x) = \prod_i P(x_i \mid \text{neighbours}(x_i))$$

(neighbours: Markov Blanket)

  - no need to inference each step
  - ☑ hold true for many cases: Machine Vision
  - may not worke well for long inference chains
- Discriminative Weight Learning
  - Conditional random fields
  - Maximize conditional likelihood of query (y) given evidence (x)

$$\frac{\partial}{\partial w_i} \log P_w(y \mid x) = n_i(x, y) - E_w[n_i(x, y)]$$

$E_w[n_i(x, y)]$ can be approximated by counts in MAP state of y given x

- Other weight learning
  - Generative: Iterative scaling
  - Discrim...: Max margin

- Structure learning
  - Start with atomic features
  - Join features → Improved score? → keep it !
  - Problem : calculate all weights for every new feature
  - Approx : keep the weights of previous features constant

* Logical Inference

- First-Order Logic
  - Constants, variables, functions, predicates
  - Literal → predicate
  - Clause → Disjunction of Literals
  - Grounding
  - World
- Inference in First-order Logic
  - Propositionalization
  - Model checking → Backtracking → ☑ DPLL
    → Stochastic Local Search → ☑ Walk SAT
- Satisfiability
  - Inputs → sets of clauses → CNF
  - Output → Truth assignment
    → NP complete
  - Solution → Search
    → Most SAT problems are easy
    → Hard region : Narrow range of $\frac{\#\text{ clauses}}{\#\text{ variable}}$
- Backtracking
  - Assign truth val. by DFS → Dead end? → change last assignment
  - Empty set of clauses → Success
  - Empty clause → Failure
  - Assigning a variable → delete false literals
    → satisfies clauses
  - Improvements → Unit propagation → fast
    → Pure literals

- Stochastic local Search
  - Completely assign → random initial → flip vars in unsatisfied clauses
  → minimize # unsatisfied (Hill climbing) → random flip (avoid local optima)
  → multiple restarts
  ☑ Walk SAT → faster DPLL
  ↘ if cannot find result it doesn't mean there's not one!

* Inductive Logic Programming

- Rule Induction
  - Examples → x: attributes
  ↘ y: concept
  - Goal → coverage of + samples
  ↘ not covering of any − sample
  - Eval → Accuracy
  ↘ Coveragee
  ↘ ...

— Algorithm

☑ Spam filtering with bag-of-words model

- First-Order Rule Induction
  - literals to add → predicates or their negations
  ↘ must include at least one var.
  ↘ changing the # of grounding of rule
  - Eval

* Putting things together

☑ KB model construction
Stochastic logic programs
Probablistic Relational Models

- keys
  - Logical Language → First order, Horn Clauses
  - Probablistic Language → Bayes Net, Markov Net, PCFG
  - Types of Learning → Generative / Discr., Structure / Params, knowledge rich / poor
  - Type of inference

- KB Model Construction
  - Logical → Horn
  - Prob → Bayes Net
    - → Ground Atom : Node
    - → Head of Clause : Child Node
    - → Body of Node : Parent Node
    - → Combination function : Noisy OR
      - >1 clause with same head
  - Learning → ILP + EM
  - Inference → Partial Grounding
    - → Belief Prop

- Stochastic Logic Programs
  - Logical → Horn
  - Prob → Probabilistic CFG
    - → Attach probs. to clauses
    - → $\sum$(clauses with same head) = 1
  - Learning → ILP
    - → "Failure Adjusted" EM
  - Inference → Do all proof
    - → Add probabilities

- Probabilistic Relational Model
  - Logical → Frame
  - Prob → Bayes Net
    - → BN temple for each class of object
    - → Obj1 attr. can be dependant on Obj & attr.
    - → Binary relations
    - → relations are independent
  - Learning → Params: Closed form (EM for missing data)
    - → Structure : "Tiered" BN
  - Inference → Full Grounding
    - → BP

- Relational Markov Nets
  - Logical → SQL query
  - Prob → Markov Net
    - → SQL query define clique
    - → Potential for each query
    - → No uncertainity over relations
  - Learning → Discreminative weight Learning
    - → No structure learning
  - Inference → Full Grounding
    - → Belief Propagation

- Baysian Logic
  - Logic → First order
  - Prob → BN → BLOG specifies how to generate relational world
  - ...

allows objects to be uncertain

- Markov Logic
  - Logic → First order
  - Prob → MN → Syntax : First order formula + weights
    - Semantics : Template for Markov net features
  - Learning → Params : Gen / Discr
    - Structure : ILP with arbitary clauses and MAP scores
  - Inference → MAP : weighted satisfiability
    - Marginal : MCMC → moves proposed by SAT solver
    - Partial grounding + Lazy inference

- Logical KB : a simple false clause in world → make it impossible (hard)

  Markov Logic : a false clause make world less probable (soft)

- in Markov Logic each formula has a weight of truth

  ☑ Smoking cause cancer → Logic : $\forall x \; Smoke(x) \Rightarrow Cancer(x)$

  → Markov : 1.5 $\forall x$ ...

- $R(x,y)$ results to $\begin{cases} R(x,y) \\ R(y,x) \\ R(x,x) \\ R(y,y) \end{cases}$

  ☑ Friend(A, B) → Friendship is assymetric

- Cliques are shown by relations and atoms

- Statistical Model : Special case of Markov Logic with zero-arity

- First order Logic : " " " " " " infinite weight

- MAP Inference

$$\underset{y}{\operatorname{argmax}} \frac{1}{Z_x} \exp\left(\sum w_i n_i(x,y)\right) \Rightarrow \underset{y}{\operatorname{argmax}} \sum_i w_i n_i(x,y) \Rightarrow \text{weighted max SAT problem}$$

☑ MaxWalk SAT

↳ Problem : Memory explosion → Solution : ground clauses lazily → sparseness

☑ Lazy SAT alg.

- Computing prob

↳ P(formula | MLN, C) = ? → MCMC sample & check formula

↳ P(f1 | f2, MLN, C) = ? → make world → satisfy f2? → satisfy f1? yes ✓

→ use lifted inference → active field

→ construct min subset of net to satisfy f2 → MCMC → ...

↳ Problem → MCMC fails most of the time → deterministic dependency

→ MCMC slow → near deterministic dependancy

→ Solution → Combine MCMC + walkSAT

☑ MC-SAT algorithm

* Applications of SRL

• Running Alchemy

- Programs
- Options
- MLN File
- Database File

⇒ Robust Programs
Smaller Codes

☑ Uniform Dist : Flipping Coin

empty MLN → Type : flip = {1, ..., 20}

Predicate : Heads (flip)

$$P(\text{Heads}(f)) = \frac{\frac{1}{Z}e^{0}}{\frac{1}{Z}e^{0}+\frac{1}{Z}e^{0}} = \frac{1}{2}$$

→ number of features = 0

☑ Binomial Dist : Unfair coin

Type : flip = {1, ..., 20}

Pred. : Heads (flip)

Formula : Heads (f)

Weight : log odds of head → $w = \log\left(\frac{p}{1-p}\right)$

using unit clause

$$P(\text{Heads}(f)) = \frac{\frac{1}{Z}e^{w}}{\frac{1}{Z}e^{w}+\frac{1}{Z}e^{0}} = \frac{1}{1+e^{-w}} = p$$

☑ Multi Nominal Dist : Dice

Type : throw = {1, ..., 20}
face = {1, ..., 6}

Predicate : outcome (throw, face)

Formula : Outcome(t, f) ∧ f != f' ⇒ !outcome(t, f'), ...

Predicate : outcome {throw, face!}

hard, need shortcut, use blocking

Formula : outcome (t1, ...)
outcome (t2, ...)
outcome (t3, ...)
...
⟹ outcome (t, +f)

☑ Logistic Regression

$$\log\left(\frac{P(C=1 \mid F=f)}{P(C=0 \mid F=f)}\right) = a + \sum_i b_i f_i$$

Type : obj : {1, ..., n}

Query Pred : C (obj)

Evidence Pred : F_i (obj)

Formulas : a C(x) → a is weight
b_i $F_i(x) \wedge C(x)$ → b is weight

Resulting Dist : $P(C=c \mid F=f) = \frac{1}{Z} \exp\left(ac + \sum_i b_i f_i c\right)$

$$\log\left(\frac{P(C=1 \mid F=f)}{P(C=0 \mid F=f)}\right) = \log\left(\frac{\exp\left(a + \sum_i b_i f_i\right)}{\exp(0)}\right) = a + \sum_i b_i f_i$$

اثبات اینکه این دو با هم برابرند

Alternative : $F_i(x) \Rightarrow C(x)$ → horn clause → معمولاً از ^ استفاده می‌شود که ساده‌تر است

☑ Text Classification

page = {1, ..., n}
word = {...}
topic = {...}
→ اختیاری → خودشان قابل استنتاج نیستند

Topic (page, topic!) → برچسب title دارد

Has Word (page, word)

! Topic (p, t) → اختیاری

Has word (p, +w) ⇒ Topic (p, +t) → حاصل یک ترکیب است

☑ Hypertext Classification

Text classification
Topic (p, t) ^ Link (p, p') ⇒ Topic (p', t)

☑ Information Retrieval

idea from Page Rank

InQuery (word)
Has Word (page, word)
Relevant (page)
InQuery (w+) ^ HasWord (p, +w) ⇒ Relevant (p)
Relevant (p) ^ Links (p, p') ⇒ Relevant (p')

☑ Entity Resolution → Given database, find duplicate records
↘ like google scholar want to merge duplicate citation

Need Transivity → hard problem in Learning
↘ easy for logic

We can resolve the fields as well as records. → two field values refer to same thing

Paper 1: Adam Smith "Pap name" conf1
paper 2: Adam Smith "pap name" conf2
→ we can guess that conf1 = conf2

☑ HMM

obs = {obs1, ..., obs N}
State = {st1, ..., st M}
time = {0, T}

State (state !, time) → exactly one state in each time
Obs (obs !, time)

State (+S, 0) → each state has different prob. at time 0
State (+S, t) ⇒ State (+S', t+1)
Obs (+O, t) ⇒ State (+S, t)

☑ Information Extraction

Auther Auther Auther | Paper Paper Paper | Venue Venue | Date → Tokening by HMM
Auther Auther Auther: Field
Paper Paper Paper: Field
Venue Venue: Field
Date: Field
record
→ Entity Resolution Needed

☑ Statistical Parsing

Sentence ⇒ Most probable parse

Symbolic Parser + Probability → Statistical Parser

CFG { A → BC, A → a }
WCFG
↳ weighted

☑ NP → N
N → Det N
⇒ Probabilistic CFG { 0.7 NP → N, 0.3 NP → Det N } ⇒ Weighted CFG

☑ Query Predicate → John ate the pizza. ⇒ NP (2, 4)
1 2 3 4
NP

• Tips
- Not make every predicate closed world → nothing to infer!
- Conjunctions are closer to probability
- Complexity → low close arity
↘ low # of constants
↘ Short inference chains
⇒ go for smallest MLN that work then expand

Design Machine Learning & (Lin - National Taiwan Univ.)

* Good Design
  - Automatic / Semi - Automatic
  - Give simple users reasonable results

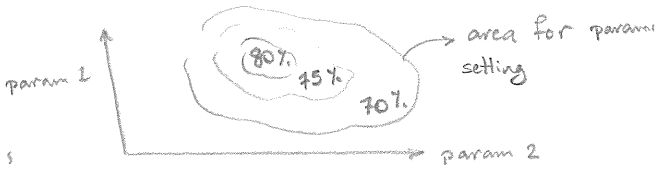
* Tests available
  - Training Accuracy
  - Magnitude of Data
  - Kernel Analysis → when kernel becomes like Identity Matrix?
    ↳ what kind of data?

$$\Phi(x_1) = [1, 0, \dots, 0]^T$$
$$\dots$$
$$\Phi(x_\ell) = [0, \dots, 0, 1]^T$$
$$\Rightarrow \Phi = e^{-\gamma\|x_i - x_j\|} = \begin{cases} i=j, 1 \\ i \neq j, 0 \end{cases}$$
$$\Phi = I$$

  - overfitting

* Improving Performance
  - Data Scaling → overfitting for some kernels can be avoided
    → prevent large value features dominance
    → Methods → linear
      → zero mean, $\Sigma = I$ (whitening)
    → Use same scaling methods and factors on training and testing
      ↳ training and testing are from the same distribution
      so max, min, var, ... are approx. similar
    → depend on kernel use range $[0, 1]$ or $[-1, +1]$ or ...

  - Parameter Selection → Regularization
    → kernel parameters → Polynomial degree
      → kernel width
    → Cross Validation Parameter → k-fold
      → Boost training (?)
    → Contouror of CV accuracy
      → only for training set
      → grid of parameters
        ↳ try proper ranges



- Selecting Kernels → RBF first → good for most data
  - → fewer params
  - → then Polynomial

- SVM Procedure
  1. Data Scaling → linear first
  2. Kernel selection → RBF first
  3. Use CV to find parameters → γ and C for SVM
  4. Use best parameters to train on training set
  5. Test

☑ in LIBSVM use easy.py

☑ With bad results when using data scaling → go to test set, print the min and max
→ maybe some min are far from 0 or max are far from 1 → that scaling socks!!
→ maybe log-scaling

* Machine Learning Software Engineering
- Some people have zero knowledge about ML → not target
- Few people are ML researcher → not target
- Target: general users
  ↳ automatic / semi-automatic setting

- Multi-class classification → one-against-rest → k binary classifier
  → one-vs-one → $\binom{k}{2}$ binary classifier
  → more suitable for kernel SVM
  [Hsu and Lin, 2002]
  ↳ user-driven demand! → new direction of research topic!

- User questions → add "one-vs-one" to library
  → add χ^2 kernel → For machine vision

– Extremes → one option for lib → simple code → easy to maintain
→ many options for best results → powerful ✓
→ complicated ✗
→ Choose Strategy

– People often try all options even if not needed.
↳ some of them has same result but different characteristics.
↳ user have different need ☑ precision / recall
↳ use sample codes to extend lib → ☑ ROC curves
F-score, AUC, ... for CV

– Avoid changes for marginal improvements ⇒ introduce unnessecary complexity
☑ Adding stratified CV → improve stability → worth adding.

- Shoot for industry → search for lessons learned

- Simplicity vs Performance → sometimes a little lass accuracy is meant really simpler algorithm

- Intuition vs. non stable formal methods → visualization and using user intuition
  - ↘ simple foolish ideas ☑ Brute force

- Numerical and Optimization Methods → pay attention to algorithm properties

  Scalability ≤ trade time for space
  - ☑ keep data in RAM? (Batch)
  - ☑ breakable in smaller problems?
  - ☑ stability for # of training samples
  - ☑ need accurate parameters?
  - ☑ simplification of optimization parts?
    $O(nl) \rightarrow O(n)$

- Numerical Stability → Catastrophic cancellation → solution: reformulation
  - ↘ numerical overflow (Floating point) → Cases of overflow → reformulation

- legacy → encapsulation
  - ↘ documentation
  - ↘ learn from questions → users assumption
    - ↘ ambiguities

Graph Mining : (Tsuda – AIST Japan)

* Structural Data

- Nature Tend to use few elements to make complex structures

☑ DNA → A, C, G, T

☑ Proteins → 20 Amino Acids

- Structure Hidden in Sequences
  - infer hidden structures → classification

☑ Biological Graph → RNA, Gene Expression Data,

☑ Molecular Graph → labled graph → nodes : elements
↘ edges : bonds

- Biological Network
  - Networks made of 1000's of nodes / 100,000's of edges

- Possible prediction

Given:		
	Sequence	Structure
	Structure	⇒ Function
	Expression	Interactions
	Pylogeny : history	Localizations

* Itemset Mining

- Frequent Item set mining → all set of elements appearing $> \alpha$
  ↳ minimum support parameter
  $0 \leq \alpha \leq T$
  ↳ usually constant

- Σ set of elements

- T → transactions → $t \subseteq \Sigma$
  ↘ $T = \{t_1, \ldots, t_m\}$

  - appearance : $X \subseteq t$
  - occurance : $occ(X,T) = \{t \in T : X \subseteq t\}$
  - frequency : $fr(X,T) = |occ(X,T)|$
  - frequent : $Fr(X,T) \geq \alpha \rightarrow X$ is frequent

  - Hasse Diagram
    - Edge : Adding item
    - Need a tree to avoid duplication → need a simple tree → BFS

  - Backtracking Algorithm → Find Frequent sets → FP growth algorith
    - Monotonicity → decrease only
    - DFS → Prune if support $< \alpha$

  - Confidence : from data provided to guess the prob. of the rest of the items

  ☑ (A, B, C) → A, B ⇒ C

- Problem: huge # of frequent itemsets
  - ↳ hard to analyze
  - ↳ many of them are similar
  - ↳ solution : Closed Pattern Mining

* Closed Pattern Mining

- The maximal set among all itemset with same occurances
- Brute Force Alg
  - ↳ generate all freq. sets → test maximality
  - exponential size
  - delay : time between two pattern outputs
  - Brute force is exponential delay w.r.t pattern size
- To get Linear delay
  - Must jump from closed set to closed set
  - How? → Reverse Search
- Reverse Search
  - Correctness proof mathematically
  - Duplication → Naïve BackTracking
  - Duplication Marking → Exponential Memory
  - Visit all nodes without duplication
  - Reduction Map
    - ↳ Mapping from children to parent
    - ↳ Given closed set X → Shrink till the occurance change → closure operation $\cap\{t \text{ in } T : X \subseteq T\}$

      (A, B, C, D) (A, B, ~~C~~, ~~D~~)
    - ↳ Arrows are in reverse direction
      - ↳ generate all children candidates → apply reduction map → Remove if not coming bac
  - Prove the correctness
    - ↳ Reduction map meet all nodes
    - ↳ find the root node from any node (by applying reduction map repeatedly)
    - ↳ children generation is inverse of reduction map
  - Algorithm : LCM
    - ↳ prefix preserving closure extension → repeat : add item and taking closure
      - ↳ ensure any closed set is generated from unique parents → choosing the smallest index of matching items → change DAG to Tree
    - ↳ linear delay in pattern size

* Ordered Tree Mining

- Finding all frequent substructures
- Labeled ordered trees
  - Rooted
  - ordered
  - Labled
- Frequency of a pattern tree(T) → # root occurances of T in data
- Given: collection of labled order tree, σ ⟹ Task: discover freq. order tree, frequency $\geq \sigma$, without duplicates
- Naive Alg
  - sorting from the smallest tree → grow pattern by adding a new node 1 by 1
  - exponential ways, need expilicit duplicatin test
  - solution: use DFS → Rightmost Expansion
    ↳ attaching a new node on the rightmost branch
- Alg.
  - Enumerate all freq. order tree by BackTracking
  - Tree extension → Rightmost Expansion → No duplication

* Unordered Tree Mining

- No order between children → exponentially many isomorphic trees
- Cannonical Representation → code(T) = $((depth_{v1}, label_{v1}) \dots (depth_{vk}, label_{vk}))$
  - special case: ordered tree → sort → lexicographically maximum code
  - Left heavy condition → ordertree is canonical ⟺ code(Tv1) ≥ code(Tv2)

* Graph Mining

- GSPan → Most popular
- Canonical Representation → No root → Choose one
  - → DFS order → number nodes → DFS code
  - → Some edges not traversed → backword edge (dest# < src#)
  - (src, dst, src_label, edge_label, dest_label)
  - Multiple DFS code → different { start point, child order } → Minimum DFS code (sort)
- Reduction Map
  - Removing the tail of min DFS code → preserve minimality

- Children Generation
  - candidate : add element to DFS code → candidate minimum? No → remove it
- Minimality Check
  - Reconstruct graph from DFS code
  - Derive min DFS code by trying all DFSs → speed up by traversing min label only
  - minimal code ≠ original ? → prune it
- gSpan → OK

  CloSpan → not ok → [Takigawa et. al, 2011]

* Dense Module Enumeration

- Challenges
  - False negative edges → go beyound clique search
  - False positive edge → assign confidence scores to edges
- Previous
  - Clique percolation
  - Partitioning
  - Heuristic local search
- Exhaustively enumerate all dense subgraph
  - Efficient
  - Detect overlaps
  - Min density for outcoming module
  - All modules output → satisfying threshold
- Problem
  - Interaction Net $G = (V, E)$
  - Edge weight $\omega \in [0,1]$
  - Density $d(U) := \dfrac{\sum_{\{u,v\} \in E} \omega(\{u,v\})}{|U|(|U|-1)/2}$

  ⇒ Find $U \subset V$

  $\begin{cases} d(U) \geq \theta \\ e_1(U) \geq n_1 \rightarrow \text{criteria 1} \\ e_0(U) \geq n_0 \rightarrow \text{criteria 0} \end{cases}$
- Enumeration
  - Set up a search tree → pruning by anti-monotonicity
    - ↳ freq (entity) < freq (sub entity)
  - density is not monotonic → pruning impossible for lexicographical ordering
  - need a search tree with monotonically decreasing density → Use reverse search
- Reverse Search
  - Specify search tree → use reduction map
    - ↳ generate a parent from a child → remove the node with smallest degree → always density ↑

- Alg for Enumeration
  1. set of children from a parent
  2. Choose children satisfying reduction mapping
  3. Prune if no children remain

* Graph ML Examples
  - Given set of graph ⟹ cluster them in groups
  - Graph Regression
  - Substructure Representation
    - 0/1 vector of pattern indicator
      - ↳ huge dimensionality
      - ↳ need graph mining for selecting features
  - Frequent Substructure Mining
    - Enumerate all patterns in at least m graph
    - Gspan
  - Discriminative patterns
    - $w_i > 0 \rightarrow +$ class
      $w_i < 0 \rightarrow -$ class
    - weighted Substructure Mining
    - Patterns with large freq difference
  - Multi class
    - $w_{\ell i} > 0 \rightarrow$ graph $i \in$ class ℓ
    - search patterns overrepresented in a class

* EM-Based graph clustering
  - learning feature & model at the same time
  - L1 regularization ⟶ make graph sparse
  - E-step → Mining step → M-step
  - Prob.
    - Binomial Mixture

$$p(x|\theta) = \sum_{\ell=1}^{C} \alpha_\ell \, p_\ell(x|\theta_\ell)$$

(α_ℓ: weight; θ_ℓ: parameter for cluster ℓ)

$$p_\ell(x|\theta_\ell) = \prod_k \frac{\exp(\theta_{\ell k} \cdot x_k)}{1 + \exp(\theta_{\ell k})}$$

  - EM

$$\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \sum_{\ell=1}^{C} \alpha_\ell \, p_\ell(x_i|\theta_\ell)$$

E-step: $r_{\ell i} = p(y_i = \ell \mid x_i)$ ⟹ computationally

M-step: estimate θ_ℓ

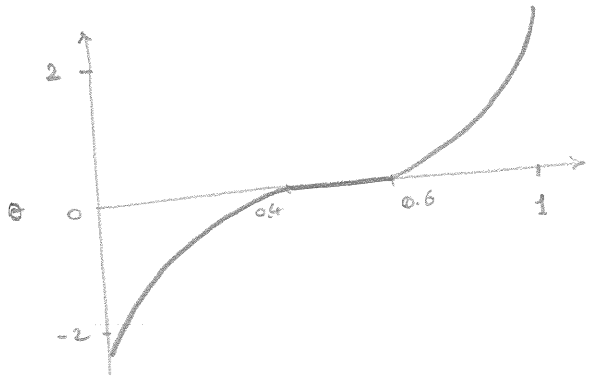
  - Use L1-regularized log likelihood

- E-step
  - Active pattern $F = \{k \mid \exists e : \theta_{ek} \neq \theta_{ok}\}$
  - E-step computation for active patterns ✓

$$p(y=e|x) = \frac{\alpha_e \prod_{k \in F} p_{ek}(x_k|\theta_{ek})}{\sum_e \text{صدت کسر}}$$

- M-step
  - use graph mining to find active patterns

$$\theta_{ek} = \begin{cases} \log \dfrac{\eta_{ek} - \lambda_e}{1-(\eta_{ek}-\lambda_e)} \\ \theta_{ok} \\ \log \dfrac{\eta_{ek} + \lambda_e}{1-(\eta_{ek}-\lambda_e)} \end{cases}$$



for active pattern, occurance prob is far from avg.

λ_e = weighted occurance

η_{ek} = occurance prob in cluster

η_{ok} = occurance prob overally

* Graph Boosting

- Given : graph → predict class label (+/−)

Typically features are given

- Motivation → Missing Descriptors
  → New features → informative patterns → greedy pattern discovery (Boosting + gSpan)
- Linear Programing Boosting
  → less calls to graph mining
  → faster than AdaBoost
- Formulation
  - $\min_{\alpha,\xi} \|\alpha\|_1 + C\sum_{n=1}^{\ell} \xi_n$
    s.t. $y_n \alpha^T x_n \geq 1 - \xi_n$, $\xi_n \geq 0$
  - sum of hinge loss and L1 regularizer
  - dual proble : ↑ # of weigh variables ⟹ ↑ # of constraints
  - Finding the most violated constraint → search by weighted substructure mining

* Entire Regularization Path

- LARS - LASSO , L1SVM
- Start from small set of features → trace solution trajectory of L-1 reg. learning → add more features

- LASSO Regression

$\beta(\lambda) = \arg\min_{\beta} L(y, X\beta) + \lambda \|\beta_1\|$ $\quad \lambda = \{0.001, 0.01, 0.1, 1, 10, \dots\}$

A : active feature set

- Piecewise Linear Path → At turning point → add new feature to A
  - ↘ delete feature from A
- Cross validation → Grid search → Solve QP many times x.
  - ↘ Path following → does not include QP → $\lambda^*_{cv} = ?$
- Feature Space of patterns
  - Representation $x_i = (x_{it})_{t \in T}$, $x_{it} = I(t \subseteq G_i)$
  - find pattern t that minimize d_t = step size
  - Tree shaped search space → node : pattern
    - ↘ edge : add edge in each level

* Kernels

- learning → general purpose learning → ☑ SVM, PCA
  - ↘ problem specific kernel
- kernel → mapping from a space to a linear (or simple) space
  - ↘ show similarity between two things
  - implicit mapping Φ → similarity measure k → $k(x,y) = \Phi(x) \cdot \Phi(y)$
  - similarity measure → invariance or other a priori knowledge
    - ☑ in vision → invariance to ratation
    - ↘ class of function
    - ↘ possibily infinite dimension → still computationally efficient
  - Kernel Trick → $x \cdot y \Rightarrow \Phi(x) \cdot \Phi(y)$
    - ↳ ☑ graph kernel
    - ↘ distance based : $d(x,y) \Rightarrow k(x,x) + k(y,y) - 2k(x,y)$
  - Valid kenels → Positive definite $\int K(x,y) \cdot dx \cdot dy \geq 0$
    - ↳ eigen values are non-negative
  - Design Kenels → Convolution
    - ↘ Generative Model-based

* Marginalized Kennel

- define kernel on the hidden variable
- Joint kernel : count kernel with context info → marginalize Joint k.
- Marginalized graph kernels
  - both vertices and edges are labled

- Label path $\{v_1, e_1, v_2, e_2, \dots v_n\}$ → by random walk

↘ probabilities are uniform → initial
↘ transition
↘ terminal

- kernel $k(h, h') = \begin{cases} 0 \\ k_v(h_1, h_1')\, k_e(h_2, h_2') \cdots k_v(h_p, h_p') \end{cases}$

↳ Marginalized $k(G, G') = \sum_h \sum_{h'} p(h|G)\, p(h'|G')\, k(h, h')$

Brain Computer Interfacing : (Muller - TU Berlin)

☑ ECoG

* Invasive vs. noninvasive BCI

• BCI: human intention ⟹ technical control signal

↳ without using activity of muscles / peripheral nerves

↳ Brain Signal ⟹ Decoding ⟹ Control Device

☑ BBCI Pong ⟶ Subject : assume two string to control the cursor

after 30 min ⟶ no need to string pulling

skill acquisition

• Challenges ⟶ cereberal cocktail party problem ⟶ different mental activity

↘ Brain Rythms ⟶ α rythm in visualcortex ⟹ overlapping things having same rythm

↘ μ rythm in sensory corter

↘ Variances ⟶ single-trial vs. averging

↘ season to season variability ⟶ day 2 day diff.

↘ inter subject variability

non stationarity ⟶ shifting distribution

• Data ⟶ channel + time ⟶ for imagining of brain state

* Common Spatial Pattern Analysis

• 2 class : Data ⟶ Whitening of joint matrix ⟶ Rotation (R matrix should be chosen so one matrix should be diagonal and the other should be 1-Diagonal).

* In Mlr non-stationarity is neglected.

* Bad Cross Validation

• Test set should not be used.

• Preprocessing of data regarding all data

• Features are selected on whole data ⟶ ☑ automatic feature selector

• Parameter selection from all data

• Insufficient validation for paradigms with block design

- Block design ⟶ blocks are dependant

↘ leave-one-block-out validation ⟶ no more optimistic statements

* Non Staticnarity

- Mathematical flvors
  - Bias adaptation
  - Covariance shift → we have model of world, then
  - Projection Method → project data to lower data that is stationary in that
  - Mixed effect model → subject dependence
  - Co-adaption → Machine and Subject adapt to each other
    ↘ Non stationarity in Rh
- Weighted Linear Regression for Covariance shift compensation
  - $P_{train}(x) \neq P_{test}(x)$
  - use weight in MSE → unbias it again → have proof
  - density estimation in high dimension is hard → weights make it nighmare
- Projection Technique : Splitting into stationary and substationary subspace (SSA)
  - PCA : X = orthogonal mixing x uncorrelated sources

$$X = A \begin{bmatrix} s_1 \\ \vdots \\ s_d \end{bmatrix} \begin{matrix} \rightarrow \text{max var} \\ \\ \rightarrow \text{min var} \end{matrix}$$

  - ICA : X = arbitary mixing x independant sources

$$X = A \begin{bmatrix} s_1 \\ \vdots \\ s_d \end{bmatrix}$$

JMLR website

  - SSA = Stationary Subspace Analysis

$$X_t = A \begin{bmatrix} s_d^0 \\ s_t^1 \end{bmatrix} \begin{matrix} \rightarrow \text{stationary sources} \\ \\ \rightarrow \text{non stationary sources} \end{matrix}$$

to get a projection that everything stationary

$$X(t) = A \cdot S(t) = \begin{bmatrix} A^s & A^n \end{bmatrix} \begin{bmatrix} S^s(t) \\ S^d(t) \end{bmatrix}$$

- SSA

objective $$\begin{bmatrix} \hat{S}^s(t) \\ \hat{s}^n(t) \end{bmatrix} = \begin{bmatrix} \hat{B}^s \\ \hat{B}^n \end{bmatrix} x(t)$$

$$\hat{B}_s = \underset{RR^T = I}{\operatorname{argmin}} \sum_{i=1}^{N} (-\log \det \hat{\Sigma}_i^s + \hat{\mu}_i^{sT} \hat{\mu}_i^s)$$

Computationally complex

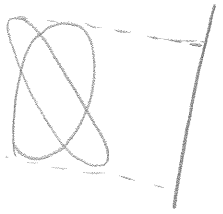
- Optimizing

$B^{new} \leftarrow R\,B^{old}$ $\Rightarrow$ find R to ease calculation of gradients

under theses rotation they look the same

we need a third one

how many rotation?



$N > \frac{D-d}{2} + 2.$ $\rightarrow$ non-stationarity needs both mean and covariance

$\Rightarrow$ Paper : Kiraly et. al. 2012 JMLR

$\Rightarrow$ BBCI.de

$\Rightarrow$ Paper : Samek et al. 2012 JNE

* Bias in computing Covariance

- sample size large

$\mu = \frac{1}{n}\sum x_k$

$\Sigma = \frac{1}{n-1}\sum (x_k-\mu)(x_k-\mu)^T$

- sample size small $\rightarrow$ biased! ☑ BCI, Financial Analysis

  - large eig values $\rightarrow$ too large
  - small eig values $\rightarrow$ too small
  - using shrinkage estimation

- Shrinkage

  - $\tilde{\Sigma}(\gamma) = (1-\gamma)\hat{\Sigma} + \gamma\nu I$

    ν = average eigen values $\rightarrow$ eigen values change toward it.

  - $\tilde{\Sigma} = (1-\gamma)\hat{\Sigma} + \gamma\nu I = (1-\gamma)VDV^T + \gamma\nu I = V((1-\gamma)D + \gamma\nu I)V^T$

    ↳ $\gamma = 0$ $\rightarrow$ no shrinkge

    ↳ $\gamma = 1$ $\rightarrow$ spherical covariance matrix

  - optimal selection of γ

$$\gamma^* = \frac{n}{(n-1)^2}\,\frac{\sum_{i,j=1}^{d} \mathrm{var}_k(z_{ij}(k))}{\sum_i (s_{ii}-\nu)^2 + \sum_{i\neq j} s_{ij}^2}$$

$\hat{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_d \end{bmatrix}$

$\hat{\Sigma} = \begin{bmatrix} & s_{ij} & \end{bmatrix}$

$z_{ij} = ((x_k)_i - (\hat{\mu})_i) \times ((x_k)_j - (\hat{\mu})_j)$

* ECoG

- Presurgical localization of brain part causing epilepsy ⟶ [Schalk]
  good to understand human

☑ good for experimenting ⟶ Index - vs - rest , Thumbs - vs - rest for finger movement

* Motory Decision

- 600 ms to action ⟶ start of process
- 300 ms to action ⟶ peak signal
- 150 ms to action ⟶ the point of no return from the action

* Brain & Reinforcement learning & (Doya - OIST)

* RL

- Value func: expected future reward
- learn action policy : $s \rightarrow a$ to maximize reward
- Temporal Difference : diff between what you expected & what you have

- Reward Field → 3D plot ⇒ Value func → 3D plot
  - $\gamma \downarrow$ ⇒ agent is lost → flat plot
  - $\gamma \uparrow$ ⇒ agent use TD → hill type plot

☑ Cart-pole swing up → r: Height of pole
  - punishment : collision

☑ learning to stand up → s: joint / head, angles
  a: torque to motors
  r: height of the head

- MDP
  - optimize policy : max cumulative reward
    - finite horizon
    - infinite horizon ($\gamma \leq 1$)
    - average reward
  - Solutions
    - Dynamic Programming → $P(s'|\cdots)$, $P(r|\cdots)$ → known
      - Solve Bellman eq.
        - Value iteration, Policy ite
      - Apply optimal policy
    - RL → $P(s'|\cdots)$, $P(r|\cdots)$ → unknown
      - learn from actual data
      - Solution → Monte Carlo, SARSA, QL, ACH, Policy gradi
      - Model-based

* Actor-Critic Learning

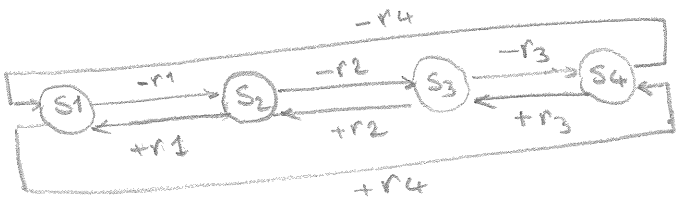
- Actor : parameterize policy
- Critic : learn value function
  - in table or neural network (function approximator)
- TD error $\xrightarrow{\delta}$: diff of actual value and predicted → supervised
- Update → critic : $\Delta V = \alpha \delta$
  → actor : $\Delta w = \delta \frac{\partial P(a|\dots)}{\partial w}$ → reinforce action by δ

* SARSA and QL

- Action-Value function: $Q(s,a) = \mathbb{E}[r(t) + \gamma r(t+1) + \dots]$
- Action selected
  - ϵ-greedy → $\arg\max_a Q(s,a)$ with prob $1-\epsilon$
    → all the sub optimal actions takes similar weights
  - Boltzman → action with higher $Q(s,a)$ is taken with higher prob.
    → β : parameter controlling greediness

- SARSA : on-policy update
  ↳ take your decision based on policy → then update $Q(s,a)$
- QL : off-policy update
  ↳ choose next action based on greedy decision → the update $Q(s,a)$

☑ "Loss to Gain" task



$\frac{r2}{r1}$ → ratio important

γ small → occillation

β large → occillation → low explore

* Issues of RL → Predict ⇒ Action Sel ⇒ Update

- Define State, action
- Design reward → balance between reward & punishment
- which algorithm
- How to set Meta parameters → learning rate, exploration
- Partial Observability → QL have problem, need hidden states
- Non Stationary → change of environment before convergance

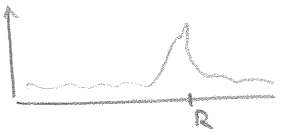
- Modules & Hierarchy

* The Brain

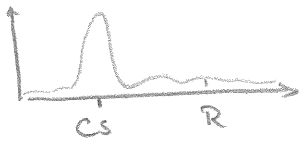
- Dopamine neurons code TD error [Scultz et al. 1997]

☑ dopamine neurons

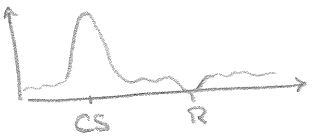
→ unpredicted reward



- اگر نداند چیزه به پرنت ترشح می کند.
- predicted reward
- اگر منتظر باشد، زمان انتظار ترشح می کند.



- اگر منتظر باشد ولی پاداش ندهیم، دیرتر پاداش بعدی ترشح می کند.
- ommited reward



- Basal Ganglia for RL? [Doya 2007] [Doya 2000]
  - state & action → cereberal cortex
  - reward pred → straitum $Q(s,a)$, $V(s)$
  - temporal diff → dopamine neurons δ
  - action selection → pallidum
  - memory (?) → thalamus

☑ addicts → problem with dopamine release

Hebbian Learning → dopamine dependant plasticity

☑ Monkey Free choice task [Samejima et al. 2005]

↳ dissociated action & reward

↳ neurons in straitum is recorded → action-value coding in striatum

☑ Forced and Free Choice Task → Iko, OIST

↳ Generalized Q-Learning Model [Ito & Doya, 2009]

↳ α_1: learning rate α_2: forgetting rate κ_1: reinforcement κ_2: aversio

- Model Fitting by particle filtering

- Neural activity in the Striatum
- Hierachy in Cortico-striatal Net [Voon et al 2004]
  - Dorsolateral → motor
    - ↳ early action → what action
  - Dorso medial → frontal
    - ↳ action value → what context?
  - Ventral → limbic
    - ↳ state value → whatever worth doing?

* Temporal Discount Factor

- γ large → go for far goals → impulsive
- γ small → go for only near goals → depression
- control the character of agent
- γ should be large enough → too large : slow learning
- Neuro modulators for Metalearning [Doya, 2002]
  - to tune parameter
  - Dopamine : TD error
  - Serotonine : higher firing during waiting for reward, stops before giving up
    ↳ regulation for patient
  -
  -
- ☑ Delag Tone-Food-Tone-Water Task

Convex Optimization : (Vanderberghe - UCLA)

* Intro

- minimize a function subject to some constraints
- global min is hard to find
- special case : Linear Programming → famous → polynomial time
- many problems can be solved → has a condition : convex
  ↘ provide heuristics for non-convex problem
- new application since 1990's
- advances → interior-point method
  ↘ first-order algorithm

* Convex Set

- $x_1, x_2 \in C$, $0 \leq \theta \leq 1 \Longrightarrow \theta x_1 + (1-\theta)x_2 \in C$ — هر خطی بین دو نقطه شکل در شکل است

- Basic sets
  - Affine set
  - Half space
  - Polyhedron → finite
  - Ellipsoid → quadratic , positive definite
  - Norm ball →
  - Positive Semidefinite Cone → S_+^n
- Intersection of convex sets is convex → using definition of convexity

* Convex Function

- $(x, f(x))$, $(y, f(y))$ $\Longrightarrow$ هر خطی بین دو نقطه تابع دیگر طرف تابع قرار می گیرد. → Jensen Inequality

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

- Basic Examples
  - linear combination
  - e^x , $-\log x$, $x \log x$
  - x^α , $\alpha \geq 1$, $x \in \mathbb{R}^+$
  - f convex $\Longrightarrow$ -f concave

- Epigraph ⟶ epi(f) : تمام نقاط بالای تابع
  ↘ f : convex func ⟺ epi(f) ⟹ convex set

- Differentiable Conv. func
  ↘ well defined min ⟶ convexity is good for minimization

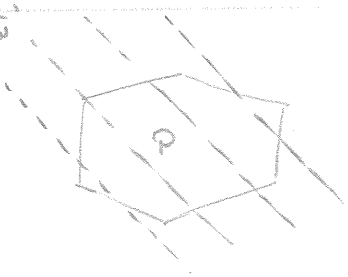
- Showing Convexity
  - 1. Def
  - 2. $\nabla^2 f(x) \geq 0$
  - 3. a function of convex function that preserve convexity

☑ point wise max , point wise supremum (infinite # for taking max)

- Conjugate function
  - $f^*(y) = \sup_{x \in dom\, f} (y^T x - f(x))$ ⟶ $f(x)$, $y^T x$ فاصله نقطه‌ای از
  - f^* ⟶ convex ⟶ not matter what is f
    ↘ sup (convex) = convex ⟶ fix x : linear : convex
  - good for duality

* linear Programming

- min $c^T x + d$
- s.t. $Gx \leq h$ (subject to)
  $Ax = b$
- look at the lines , move them , to find the result



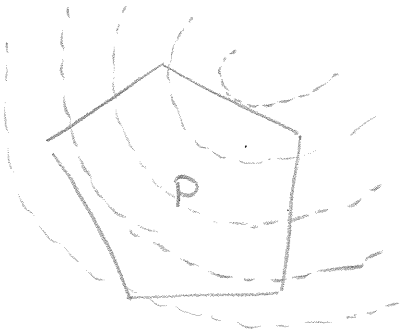
- usually found in piece-wise linear minimization
  - min $f(x) = \max_i (a_i^T x + b_i)$
  - transform to min t
    s.t. $a_i^T x + b_i \leq t$, $\forall i$

- ℓ_1-norm approximation
  - def $\|y\|_1 = \sum_k |y_k|$
  - min $\|Ax - b\|_1$
  - min $\sum y_i$
    s.t $-y \leq Ax - b \leq y$

- l_∞ - norm $(\|y\|_\infty = \max_k |y_k|$

  - min $\|Ax-b\|_\infty$

  - min y
    s.t $-y\vec{1} \leq Ax-b \leq y\vec{1}$



* Quadratic Programming

- min $(\frac{1}{2}) x^T P x + q^T x + r$
  s.t. $Gx \leq h$

- linear program with random cost
  min $c^T x$
  s.t $Gx \leq h$

  $c \sim N(\bar{c}, \Sigma)$

- Kernel Formulation
  - min $f(Xa) + \|a\|_2^2$ → Euclidean dist
  - dual → min $f(y) + y^T Q^{-1} y$

    $Q = X X^T$ → kernel matrix

* Geometric Programming

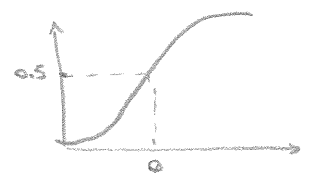
- $f(x) = \sum_{k=1}^{k} c_k x_1^{a_{1k}} x_2^{a_{2k}}$

  min $f_0(x)$
  s.t. $f_i(x) \leq 1 \quad \forall i$

- Second - order Cone Programming (SOCP)

  min $f^T x$
  s.t. $\|A_i x + b_i\|_2 \leq c_i^T x + d_i$

- Robust linear program (stochastic)
  - min $c^T x$
    s.t. $\text{prob}(a_i^T x \leq b_i) \geq \eta$ → ☑ $\eta = 90\%$
  - $\eta \uparrow$ ⇒ stay away from edges of polyhedron
    $\eta \downarrow$ ⇒ more toward non - convexity ⟹ sharp edge ⇝ $\eta = 50\%$
  - convert to LP (require $\eta \geq 50\%$)



$\Phi^{-1}(\eta) > 0 \iff \eta \geq 0.5$
convex

Φ : cumulative density function

* Semidefinite Program (SDP)

- min $c^T x$
  s.t. $x_1 A_1 + x_2 A_2 + \cdots + x_n A_n \preceq B$
- include many non-linear constraints as special cases
- semidefinite relaxation
  - to solve non-convex → find bounds
    ↘ heuristic
  - reformulation trick → change the variable → try to make constraints convex
    ↘ expand formula
  - replace constraints with weaker ones → relax!
    ↘ for example to obtain lower bound

* Conic Optimization

- Inequality → constraint $x \in K$ → K : convex cone
- Proper Cones → closed, pointed, nonempty interior
- Cone linear Program
  - min $c^T x$
    s.t $Ax \preceq_K b$
  - popular → modeling : small # of primitive cones is sufficient
    ↘ algorithm : extend LP
- Second order cone program
  - min $c^T x$
    s.t $\| B_{k0} x + d_{k0} \|_2 \leq B_{k1} x + d_{k1}$
- R^n_+, Q^p, S^p → enough cone set for many applications
  ↳ exponential & logarithm

* Duality

- Primal: min $c^T x$ s.t. $Ax \preceq b$ ↳ optimal p^* ⟷ Dual: min $-b^T z$ s.t. $A^T z + c = 0$, $z \succeq 0$ ↳ optimal d^*
- $p^* \geq d^*$ → weak duality
  $p^* = d^*$ → strong duality
- $x \succeq_* y \equiv x \succeq_{K^*} y$ → using proper cone K

* Boosting & (Schapire - Princeton U.)

- To finding single highly accurate prediction rule
  - ↳ from "rule of thumbs" that are often correct
  - ↳ how to choose examples in each round of finding rule of thumbs
    - ↳ hardest examples
  - ↳ how to combine those rules
    - ↳ weighted majority vote of them

- have weak learning algorithms with accuracy better than random → acc > 55 % (rule of thumbs)

⇒ make very high accuracy classifier (e.g. 99 %)

* History

- Arise from PAC model : From weak PAC to strong PAC

↓

If procedure found we can reach perfect accuracy

↓

learning is in two states → completely learn
→ random performance

- First boosting algorithms
- Ada Boost → ...

* Ada Boost

1. m training example $(x_1, y_1) \dots (x_m, y_m)$ → $y_i \in \{0,1\}$
2. T rounds
   - ↳ construct dist D_t on $\{1, \dots, m\}$ → weights → $\in \mathbb{R}^+$ / $\Sigma = 1$
   - ↳ Find weak classifier h_t
   - ↳ measure error ε_t on D_t → prob that an example missclassified
3. Combine classifiers H_{final}

- Construct D_t
  - $D_1(i) = 1/m$
  - $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha t} & y_i = h_t(x_i) \\ e^{\alpha t} & y_i \neq h_t(x_i) \end{cases}$ → Z: normalization factor
    → increase the weight on missclassified
    ↓
    finding hardest examples

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i))$$

- Construct Final Classifier

$$H_{final}(x) = \text{sign}\left(\sum \alpha_t h_t(x)\right)$$

- Calculate α

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$$

- The training error is always decreasing during the training

↳ edge $\gamma_t \longrightarrow \varepsilon_t = \frac{1}{2} - \gamma_t$

$$\text{training error}(H_{final}) \leq \prod_t \left[2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right] \leq \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

↳ even if γ is small $\quad \text{error}(H_{final}) \leq \exp(-2\gamma^2 T) \qquad \gamma_t \geq \gamma > 0$

↳ does not need to know γ or T apriori $\longrightarrow$ adaptive boosting

- Proof

1. $D_{t+1}(i) = D_t(i) \cdot \sim \quad \Longrightarrow D_{final}(i) = \frac{1}{m} \dfrac{\exp\left(-y_i \sum \alpha_t h_t(x_i)\right)}{\prod_t Z_t}$

2. $\text{training error} \leq \prod_t Z_t$

$$\text{training error} = \frac{1}{m}\sum I(y_i \neq H_{final}(x_i)) = \frac{1}{m}\sum I(y_i F(x_i) \leq 0)$$

$$\leq \frac{1}{m}\sum \exp(-y_i F(x_i)) = \sum \underbrace{D_{final}(i)}_{1} \prod_t Z_t = \prod_t Z_t$$

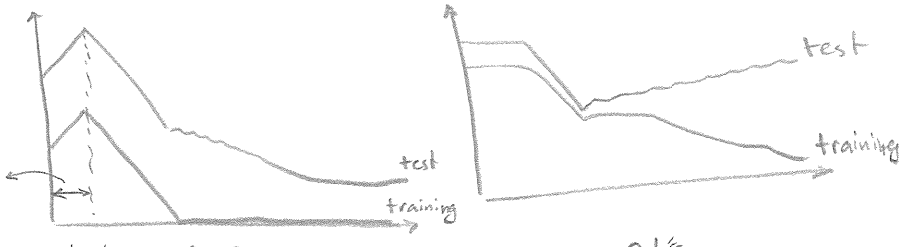
3. $Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$

$$Z_t = \sum D_t(i) \exp(-\alpha_t y_i h_t(x_i)) = \sum_{i: y_i \neq h_t} D_t(i) e^{\alpha_t} + \sum_{i: y_i = h_t} D_t(i) e^{-\alpha_t}$$

$$= \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \quad \curvearrowright \text{where } \alpha_t \text{ come from}$$

- The test error
  - increase in begining $\longrightarrow$ do better
  - decrease after round* $\longrightarrow$ adding boosting a new classifier each round
    - $\searrow$ overfitting $\longrightarrow$ according to Occam's razor

1 classifier ⇒ dictatorship
2 classifiers ⇒ one always wins



training err = 0
test error decreasing } ⇒ occum razor!

- **Margin explanation**

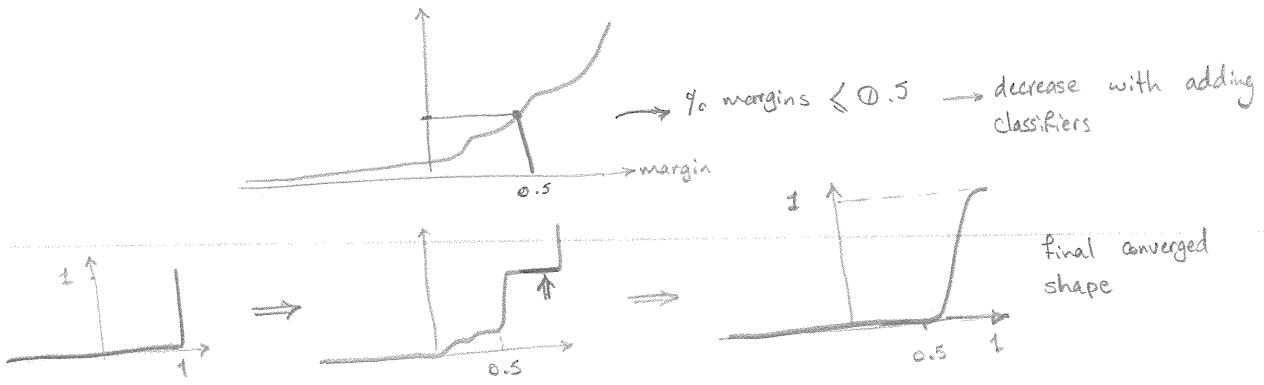
  - idea → training err = validating classification
    - ↘ we need away of measuring confidence

  - for voting → margins show the winner and confidence
    - ↘ AdaBoost is a weighted majority vote
    - ↘ margin = strenth of the vote

      = weighted fraction voted correctly − weighted fraction voted incorrectly

      - ↳ sign (margin) → voting correctly / incorrectly
      - ↳ |margin| → confidence

  - margin dist = cumulative dist of margins of training example



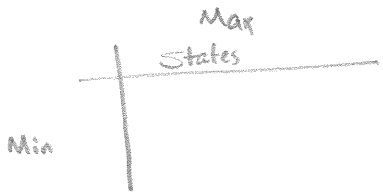
  - larger margins ⇒ better bounds on generalization err
    - ↳ independent of # of rounds

  - boosting tends to increase margins of training examples (given weak classifier assumption)

  - final classifier → getting bigger
    - ↘ getting more look like simpler classifiers
    - ↘ larger margins
    - ↘ resistent to overfitting → small edge : underfit
      - ↘ overly complex weak classifier : overfit

* Game Theory Perspective

- Boosting as a game
- Game



→ Min : dist P over rows

Max : dist Q over columns

$$\text{Min Loss} = \sum P(i) M(i,j) Q(j) = P^T M Q = M(P,Q)$$

- Sequential Game : Min play before Max

↳ $L(P) = \max_Q M(P,Q)$

↳ $\min_P L(P) = \min_P \max_Q M(P,Q)$

↳ playing better is usually looks better but in real it makes no difference → Von Neumann's theorem of Minmax

$$\min_P \max_Q M(P,Q) = \max_Q \min_P M(P,Q) = v$$

☑ in Rock-Paper-Scissors ⇒ play with uniform dist on action

- Classic theory weakness

↳ game unknown

↳ game might be exteremly large

↳ not adverserial opponent → room for better performance

- Repeated Play

- possible to learn to play well → learn game matrix / opponent

- M unknown

in a loop
- Min chooses P_t
- Max chooses Q_t
- Loss of Min = $M(P_t, Q_t)$
- Min observe loss $M(i, Q_t)$ of each pure strategy i

- Multiplicative-weight Algorithm

↳ punish weight of strategies suffering the most loss

$$P_{t+1}(i) = \frac{P(i) \exp(-\eta M(i, Q_t))}{\text{normalization}}$$

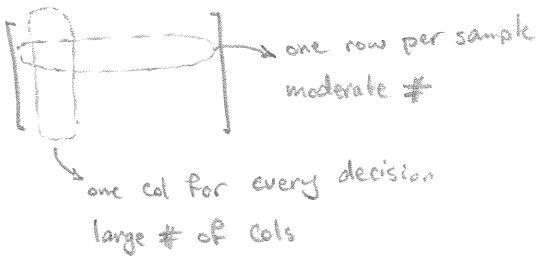
↳ goal : actual avg. loss $\leq$ best average loss + Δ_T

$$\underset{\text{regret}}{\Delta_T} = O\left(\sqrt{\frac{\ln m}{T}}\right) \to 0 \quad \begin{cases} \text{ind}(\#\text{ columns}) \to \text{opponent} \\ \text{dep}(\#\text{ rows}) \end{cases}$$

- Solving a game
  - Min play using MW
  - Max choose best response

- Boosting as a game
  - Min → booster (row)
  - Max → weak learner (col)
  - Matrix M
    - row : training exampler
    - col : weak classifier

M :



- M(i,j) = which classifier are correct / incorrect on each example
  - γ-weak learning assumption
    for each dist on example there's weak classifier with weighted err $\leq \frac{1}{2} - \gamma$
    - value (M) $\geq \frac{1}{2} - \gamma$
    - there's a weighted majority classifier that correctly classifies all training examples by margin $\geq 2\gamma$
      - Use MW to solve the game ⇒ variant of AdaBoost

* Loss Minimization

- Fitness measuring → ex) least Square, ...
- why? → convergance proof
  - decoupling alg from objective
- AdaBoost
  - training err $(H_t) \leq \prod_t Z_t$
  - $Z_t = \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$
    - α_t minimize Z_t
    - h_t minimize ε_t ⇒ minimize Z_t
  - greedy procedure for minimizing exponential loss
    - $\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i F(x_i))$
    - $F(x) = \sum_t \alpha_t h_t(x)$

- why exp. loss
  - ↳ encorage exponentially to $F(x_i)$ and y_i have same signs
  - ↳ upper bound on training err
    - ↳ non-convex

- Coordinate Descent
  - ↳ $F(x) = \sum_t \alpha_t h_t(x) = \sum_j \lambda_j g_j(x)$
    - ↳ find λ_i to minimize loss

$$L(\lambda_1, \dots \lambda_N) = \sum_i \exp\left(-y_i \sum_j \lambda_j g_j(x_i)\right)$$

- Ada Boost as coordinate descent → powerful over huge space of functions
  - ↳ choose coord. λ_i ⇒ update by corresponding h_t incrementing by α_t

- Functional Gradient Descent (function of function)
  - ↳ $\mathcal{L}(F) = \mathcal{L}(F(x_1), \dots, F(x_m)) = \sum_i \exp(-y_i F(x_i))$
  - ↳ gradient decent : $F \leftarrow F - \alpha \nabla_F \mathcal{L}(F)$
    restricted : $F \leftarrow F + \alpha h_t$
    ⇒ h_t close to $-\nabla_F \mathcal{L}(F)$
  - ↳ Equivalent to Ada Boost

- Estimate $\text{Prob}(y=+1|x)$
  - ↳ empirical version → x, y random from true dist

$$\mathbb{E}_{x,y}\left[e^{-yF(x)}\right] = \mathbb{E}_x\left[\Pr(y=+1|x)\, e^{-F(x)} + \Pr(y=-1|x)\, e^{F(x)}\right]$$

$$\min_F(x) = \frac{1}{2} \cdot \ln\left(\frac{\Pr(y=+1|x)}{\Pr(y=-1|x)}\right) \Rightarrow \Pr(y=+1|x) = \frac{1}{1+e^{-2F(x)}}$$

- Calibration Curve

- Regularization

$$L(\lambda) = \sum \exp\left(-y_i \sum_j \lambda_j g_j(x_i)\right)$$

$$\ell_1\text{-reg}: \min L(\lambda) \text{ s.t. } \|\lambda\|_1 \le B$$

$$\equiv \min L(\lambda) + \beta\|\lambda\|_1$$

↳ if $\alpha \to 0$: results looks like Ada Boost

* Information - Geometric View

- Loss Minimization ⟶ functions computated on ada boost ⟶ weights on weak classifie
  ⇕ dual
  Info - Geometrix ⟶ distributions ⟶ weights on examples

- Iterative - Projection Alg
  - find point closest to x_0 in set ⟶ $P = \{\text{intersection of hyperplane}\}$
  - pick x_0 ⟶ project on hyperplane ⟶ converge if $P \neq \emptyset$
  - Ada Boost is Iterative - Projection .
    - ↳ points = dists over training examples
    - ↳ distance = relative entropy ⟶ distance between two distributions
    - ↳ reference point x_0 = uniform dist
    - ↳ hyperplanes defined by all possible weak classifiers ⟶ looking for hardest dist.
  - Alg.
    start from uni. dist ⟶ pick weak classifier ⟶ entropy projection on hyperplane

* Multiclass Problem

- Direct Approch ⟶ $H_{final} = \underset{y \in \{1,\dots,k\}}{\operatorname{argmax}} \sum_{t:\, h_t(x)=y} \alpha_t$
  Ada Boost . M1
  ↘ bad result for "weak" weak classifire
- One - Against - All ⟶ k - Binary class
  Ada Boost . MH
- Output Coding

* Ranking Problem

- Learn to Rank ⟹ Rank Boost ⟹ bts of decision of A better than B
  from example

* Confidence Rated Prediction

- Ideal prediction ⟶ one side of prediction is always + example
  ↘ the other side is not important
- Hard prediction ⟶ bad predictions ⟶ need clean up
  ↘ speed ↓
- introduce Confidence ⟶ $|h_t(x)|$ = confidence
  ↘ $\operatorname{sgn}(h_t(x))$ = prediction

* Optimal Accuracy

  • AdaBoost can converge to Bayes optimal

    – enough data, run for many runs, good weak classifiers

    – Universally consistent

  • Noise

    – there are data sources that AdaBoost fails miserably

    – but anyway is better than random → err ≤ 50

    – works bad → not rich enough classifiers

    → prone to noise → Solution: branching program

    → regularization maynot help.

* Optimal Efficiency

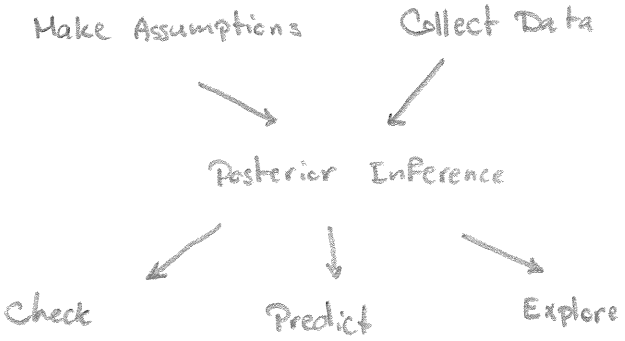
  • AdaBoost works no better then boost-by-majority

    – AdaBoost train error is like Chernoff approx of BBM

Topics Models

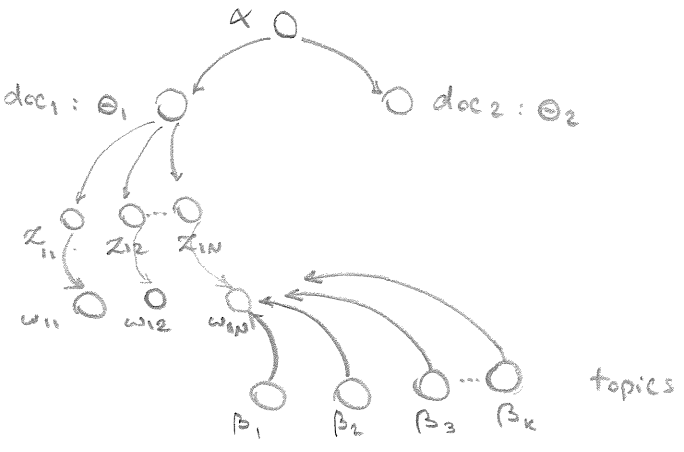
(Blei – Princeton)

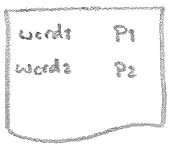
* Topic Models → Bayesian Model
  - Specifying models
  - Model Selection
  - Aproximate posterior inference
  - Using and Evaluating Models

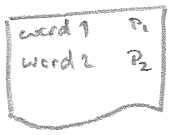


☑ Baysianly → word appear in [Robin, 1984]

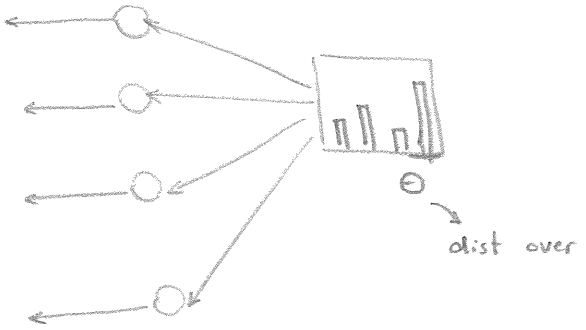
* Latent Dirichlet Allocation (LDA)
  - Intuition: Docs exhibit multiple topics
    - Topic: Dist. over words
    - Doc: New dist over topis for each doc. → Bag-of-word assumption
    - Topics are fixed over all docs. But each doc may have different topics
      → Mixture → Topics Fixed for all docs, docs have different dist. of topic
    - Simplex
    - Doc → Empty Topic → According to dist of topics → Pick a topic
      → According to topic dist → pick a word ($Z_{d,n}$) → Repeat it for all words in topic
  - A graphical models
    - edges → possible dependence
    - nodes → random variable (shaded = observed)
    - plane (▭) → hyper paramters
    - Encodes dependance assumption
    - $Z_{d,n}$ → depends on topics and → index of topic
      → drawn on θ







word in doc



$$p(\beta,\theta,z|w) = \frac{p(\beta,\theta,z,w)}{p(w)} = \frac{p(\beta,\theta,z,w)}{\int_{\beta,\theta,z} p(\beta,\theta,z,w)}$$

$$p(\beta,\theta,z,w) = \left(\prod p(\beta_i|\eta)\right)\left(\prod_{d=1}^{D} p(\theta_d|\alpha) \prod_{n=1}^{N} p(z_{d,n}|\theta_d)\, p(w_{d,n}|\beta_{1:k}, z_{d,n})\right)$$

☑ Jstor → scan docs → OCR them → index them

- conjugate → if prior comes from a family of dist. (Dirichlet for example) →
  → sampled
  posterior is from the same family

- Dirichlet Distribution

$$p(\theta|\alpha) = \frac{\Gamma(\sum \alpha_i)}{\prod \Gamma(\alpha_i)} \prod \theta_i^{\alpha_i - 1}$$

$$\mathbb{E}[\theta] = \frac{\alpha}{\sum \alpha_i} \longrightarrow \text{☑ } \mathrm{Dir}(\underbrace{\alpha,\alpha,\dots,\alpha}_{10}) \Big|_{\alpha=1} \Rightarrow \mathbb{E}[\theta] = \frac{1}{10}$$

$\sum \alpha_i$ → variance around the mean

$\alpha \uparrow \Rightarrow$ closer to the mean

$\alpha \downarrow \Rightarrow$ more sparsity $\Rightarrow \sigma^2 \uparrow$

$\alpha \to 0 \Rightarrow$ choose on prior and give it all the mass (= 1)

☑ Movie : powers of ten

- Why LDA works?
  - For each doc, allocate words to as few topics as possible.

$$p(\beta,\theta,z|w) \propto p(\beta,\theta,z,w)$$

$$= p(\beta) \prod_d p(\theta_d) \left(\prod_n p(z_{dn}|\theta)\, p(w_{dn}|z_{dn},\beta)\right)$$

($\beta \to \eta$, $\theta_d \to \alpha$, $p(z_{dn}|\theta)$)

$p(z_{dn}|\theta) \downarrow \Rightarrow p(\beta,\theta,z|w) \downarrow$

more topics $\Rightarrow p(z_{dn}|\theta) \downarrow$ ← its sits on simplex

- For each topic, assign high probability to as few terms as possible

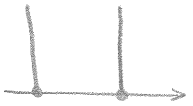
  ↳ same reason ...

- if we have only 1 topic ⇒ high prob → more topic ⇒ posterior ↓

  if we have only 5 topic ⇒ high word prob → more words ⇒ posterior ↓

* Extending LDA : Correlated and Dynamic Topic Model

- LDA → hidden assumption → component are nearly independent

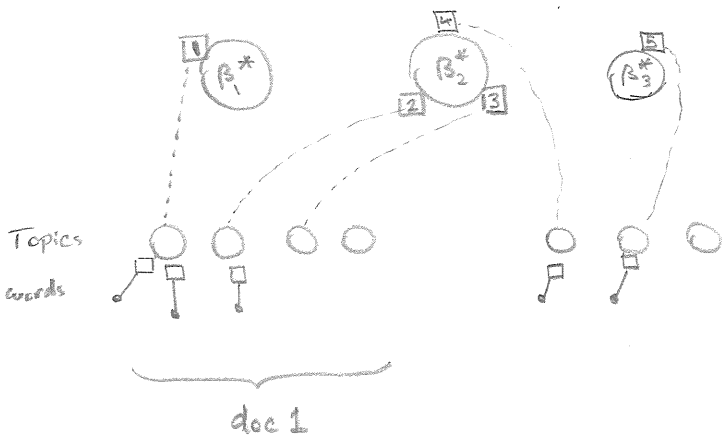


- Logistic Model → capture dependance between component
  ↘ is not conjugate
- topics in the course of time are dependant $\beta_{t,k} | \beta_{t-1,k} \sim \mathcal{N}(\beta_{t-1,k}, \sigma^2)$
- Dynamic Topic Model :
  - if some new topic introduces because of low variance estimation before it was penalized in previous model → so we drift the mean toward that a little bit to explain that
  - So we need to ease the acceptance of new topics in future, so they should give feedback to previous stages.

---

* Baysian Non parametric Models

- Topic Models → fixed number
  ↘ regularization parameter → CV Model selection
  ↘ using non-parametric models
- Non Param. Models (BNPM) → extract new topics in corpus in future time.
- Chinese Resturant Process (CRP)
- The chinese restaurant franchise



→ infinite # of tables

Graphical Models (Wainwright - Berkeley)

* Computational Challenges:
  - Efficiently compute MAP

$$\hat{x} = \arg\max_{x_1,\dots,x_N} p(x_1, x_2, \dots, x_N \mid \underbrace{y_1, \dots, y_N}_{\text{observation}})$$

  - Normalization Constants

$$Z = \sum_{x \in X^N} \prod_{c \in C} \psi_c(x_c)$$

☑ Shannon Experiment of Generating English Sentences with Markov Network

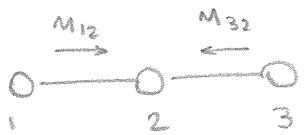
* Efficiently Solving MAPs ⟶ Belief propagation, message passing, max-product
  - No cycles in graphs
  - Compute MAP on tree ⟶ Extension on Viterbi Algorithm

$$\hat{x} = \arg\max_{x \in X^N}$$

  - what info each node need to recieve from neighbor to do the computation
    - ↳ Message: vectors of number, blue arrows
    - ↳ Divide & Conquere principle

☑ $$\max_{x_1,x_2,x_3} p(x_1, x_2, x_3) \propto \max e^{\theta_1(x_1)} e^{\theta_2(x_2)} e^{\theta_3(x_3)} e^{\theta_{12}(x_1,x_2)} e^{\theta_{23}(x_2,x_3)}$$

$$\propto \max_{x_2} e^{\theta_2(x_2)} \times \left\{ \max_{x_1} e^{\theta_1(x_1) + \theta_{12}(x_1,x_2)} \right\} \times \left\{ \max_{x_3} e^{\theta_3(x_3)} \cdots \right\}$$

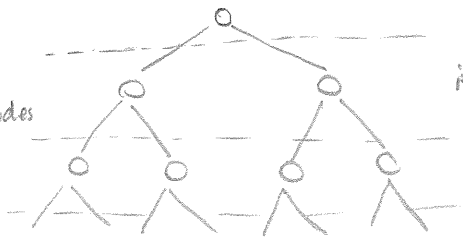


brute force: (2^n) ⟶ message passing: (n^2)

won't work if there is an edge from 1 to 3 ⟶ cycle ⟶ cannot simplify

  - Message start from leaf ⟶ start to complete subtrees ⟶ when a node has enough info
  - iter # ⟶ diameter of graph: longest path
  - parallel version of Max-Product vs. flooding schedule: the trends in computation and implementation: 2 sweep
  - Generalization ⟶ Junction Trees

* Max-product on graph with cycles
  - Messages as rumors → inexact best guesses
  - when you hear a rumor → from more people : stronger belief
    → from a far distance : it gets weaker → long chains
  - No general guarantee → Can algorithm lie? yes! → converge to bad answers
    - Computation tree
      ↓
      there is repetetive nodes



wheres the information flowing in graph through iterations

    - Solving Max-product on the computation tree
      ↳ same potential graph
      ↳ lots of copies on one nodes → some visited more often → higher degree
      ↳ not balanced in computation
      ↳ having higher weighs
    - Fix repetative node problems → instead of telling a lie says I'm confused.
      ↳ weighting the edges

---

* Tree-reweighted Max-Product
  - I won't get back any of my own messages.
  - Visit every node and edge the same # of times → by weighting them
  - where the weights come from? → one possibilities
    ↳ put all spanning tree with same prob. in a hat!
    ↳ what is the prob. that an edge appear in spanning trees → count 'em!
  - converge : if you schedule messages correctly → won't lie → MAP result guaranteed or say i'm confuse
  - rewrite the MAP as Linear Programming

$$\hat{x} = \arg\max_{x_1,\dots,x_N} \underbrace{\left\{ \sum_{s\in V} \theta_s(x_s) + \sum_{(s,t)\in E} \theta_{st}(x_s, x_t) \right\}}_{J(x,\theta) \to \text{cost function}}$$

instead of max over x, maximize on distribution supporting them

$$\hat{x} = \max_{q(\cdot)} \left\{ \sum q(x)\ J(x,\theta) \right\} \qquad q(\cdot) \geq 0,\ \sum q = 1$$

$$\hat{x} = \max \left\{ \sum_{s \in V} \sum_{x_s} \mu_s(x_s)\, \theta_s(x_s) + \sum_{(s,t) \in E} \sum_{x_s, x_t} \mu_{st}(x_s, x_t)\, \theta_{st}(x_s, x_t) \right\}$$

local marginals $\{\mu_{1(\cdot)}, \ldots, \mu_{st}\}$

$$\begin{bmatrix} \mu_{1(0)} \\ \mu_{s(m-1)} \end{bmatrix} \geq 0 \qquad \begin{bmatrix} \mu_{st}(0,0) & & \\ & & \\ & & \mu_{st}(m-1, m-1) \end{bmatrix}$$

$$\sum_{x_s} \mu_s = 1 \qquad \sum_{x_t} \mu_{st}(x_s, x_t) = \mu_s(x_s)$$

↳ constraints of Linear Programming

↳ messages in max-products $\equiv$ lagrange product in LP

• Marginal polytope

$$\mathbb{M}(G) = \left\{ \mu_s, \mu_{st} \;\middle|\; \mu_s(x_s) = \sum_{x_u, u \neq s} q(x_1, \ldots, x_N) \quad \text{for some } q(\cdot), \;\; \mu_E(x) = \sum_{x_u, u \neq t,s} q(x_1, \ldots, x_N) \right\} \longrightarrow \text{convex set}$$

• Psudo marginals (semi-definite relaxation)

$$\mathbb{L}(G) = \left\{ \tau_s, \tau_{st} \;\middle|\; \tau_s(x_s) \geq 0, \; \sum_{x_s} \tau_s(x_s) = 1, \; \sum_{x_t} \tau_{s,t}(x_s, x_t) = \tau_s(x_s) \right\}$$

• How many of Polytope halfspaces we need? $\mathbb{L}(G)$ has few constraint → easy to optimize

How many constraint needed in $\mathbb{M}(G)$?

☑ $\tau \notin \mathbb{M}(G)$ by semi definite constraints

$x_s \in \{0,1\}$, $s = 1,2,3$ and figue in P 27/35

$$\mathbb{E}(x_s) \overset{?}{=} 0.5 \quad \text{for } s = 1,2,3$$

$$\mathbb{E}(x_s, x_t) \overset{?}{=} \begin{cases} 0.9 & \text{for } (s,t) \in \{(1,2),(2,3)\} \\ 0.1 & \text{for } (s,t) = (1,3) \end{cases}$$

$$\mathbb{E}\left[\begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} (1\, x_1\, x_2\, x_3) \right] = \mathbb{E}\begin{bmatrix} 1 & x_1 & x_2 & x_3 \\ x_1 & x_1 & x_1x_2 & x_1x_3 \\ x_2 & x_1x_2 & x_2 & \cdots \\ x_3 & x_1x_3 & x_2x_3 & x_3 \end{bmatrix}$$

binary values : $x_1^2 = x_1$

PSD

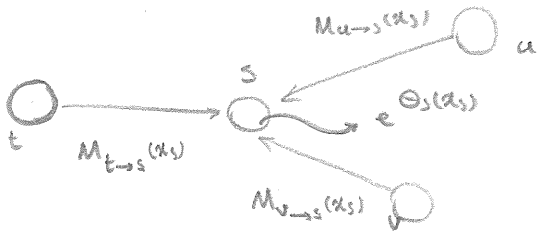
$$= \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.4 & 0.4 \\ 0.5 & 0.4 & 0.5 & 0.1 \\ 0.5 & 0.4 & 0.1 & 0.5 \end{bmatrix} \overset{?}{\succeq} 0$$

→ eigen value should be positive because the first $\mathbb{E}$ is rank one → but it is not P.S.D !

↳ positive semi definite

✳ Sum - Product message Passing

• replcing max operator by sum in all formula



$$P(X_s = \bar{x}_s) \propto \exp\{\theta_s(\bar{x}_s)\} \prod_{u \in N(s)} M_{u\to s}(\bar{x}_s)$$

Kalman Filters ⟶ special case with Gaussian Dist

↳ Msgs: means + covariances

↳ use these relations and put Gaussian in them too.

Fast Fourier Transform ⟶ special case too

[Aji & McLiece, 2001 ⟶ survey]

- using bethe entropy approx.
  - information on edges
  - using entropy as cost function

☑ Entropy and Markov Chain

$$p(x_1, x_2, x_3) = \mu_1(x_1)\ \mu_{2|1}(x_2 \mid x_1)\ \mu_{3|2}(x_3 \mid x_2)$$

$$= \mu_1(x_1) \times \frac{\mu_{12}(x_1, x_2)}{\mu_1(x_1)} \times \frac{\mu_{23}(x_2, x_3)}{\mu_2(x_2)}$$

$$= \mu_1(x_1) \times \mu_2(x_2) \times \mu_3(x_3) \times \frac{\mu_{12}}{\mu_1 \times \mu_2} \times \frac{\mu_{23}}{\mu_2 \mu_3}$$

$$H(p) = -\sum_{x_1, x_2, x_3} p(x_1, x_2, x_3) \log p(x_1, x_2, x_3)$$

$$H_{\text{Bethe}}(\mu, p) = \sum_{s \in V} H_s(x_s) - \sum_{(s,t) \in E} \rho_{st}\, I_{st}(\mu_{st})$$

↳ mutual info

- why entropy arise in duality

$$p(x_1, \ldots, x_N; \theta) = \frac{1}{Z(\theta)} \exp\left\{ \sum_{c \in C} \theta_c(x_c) \right\} \rightsquigarrow \text{exponential family}$$

☑ $\Phi(x_i) = x$ ⟶ $\mathbb{E}(X) = \hat{\mu}$: sample mean

☑ Max Entropy ⨯ Max Likelihood

$x \in \{0,1\}$ $\quad p(x, \theta) = \exp\{\theta x - A(\theta)\}$ $\quad A(\theta) = \log Z(\theta) = \log(1 + e^{\theta})$

$\mu = P(x = 1)$ $\quad -H(\mu) = \mu \log \mu + (1-\mu)\log(1-\mu) = A^*(\mu)$

(A, A^*) ⟶ conjugate dual pair

$$A^*(\mu) = \sup_{\theta \in \mathbb{R}} \{\theta\mu - A(\theta)\}$$

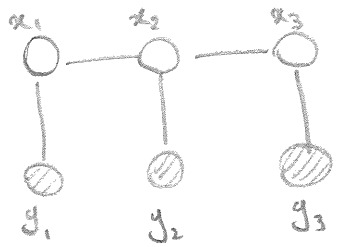
derivative w.r.t. $\quad \mu - \frac{e^{\theta}}{1 + e^{\theta}} = 0$ } Moment matching problem

$1 + e^{\theta} = p(x=1; \theta)$

if $\mu \in (0,1)$ has an answer : $\theta = \log \frac{\mu}{1-\mu}$

$$\Rightarrow A^*(\mu) = \begin{cases} \mu \log \mu + (1-\mu)\log(1-\mu) & \text{if } 0 < \mu < 1 \\ +\infty & \text{otherwise} \end{cases}$$

- why do we care about partition function?



$$p(x,\theta) \propto \prod_{c \in C} e^{\theta_c(x_c)} \quad \} \text{ prior}$$

$$p(y|x) \propto \prod_i p(y_i|x_i)$$

$$p(x|y,\theta) = \frac{p(x;\theta)p(y_i|x)}{p(y,\theta)}$$

$$H_{\text{Bethe}}(T,\rho) = \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} \rho_{st} I_{st}(\mu_{st})$$

singleton entropy; edge weight; mutual info

- we need upper bound

$$A(\theta) = \log z(\theta) = \log \sum_x \prod_c e^{\theta_c(x_c)}$$

$$A\left(\sum_T \rho(T)\theta(T)\right) \leq \sum \rho(T) A(\theta(T)) \longrightarrow A \text{ is convex, apply Jensen's}$$

- many trees in a graph → number grows exponentially

- use duality → put exponential explosion aside

✳ Learning Graphs from Data for pairwise models

- $$Q(x_1 \dots x_p;\theta) = \frac{1}{Z(\theta)} \exp\left\{ \sum_{s \in V} \theta_s x_s^2 + \sum_{(s,t) \in E} \theta_{st} x_s x_t \right\}$$

(a) $x_s \in \mathbb{R}$ → multivariate Gaussian (zero-mean)

(b) $x_s \in \{-1,+1\}$ → Ising Model

Unknown : edge structure and weights

Parameter Matrix: $\Theta \in \mathbb{R}^{p \times p}$ → p = # of nodes

Learning Problem : collect sample $X_{i,} = (X_{i1}, \dots, X_{ip})$

$$x_1^n = \begin{bmatrix} - x_1^T - \\ \vdots \\ - x_n^T - \end{bmatrix} \in \mathbb{R}^{n \times p} \quad (\text{or } \in \{0,1\}^{n \times p})$$

Wanted: Support on Θ → non-zeros { all edges, no false edges

$$\text{supp}(\Theta) = \{(s,t) \mid \Theta_{(s,t)} \neq 0\}$$

Estimator : $X_1^n \longmapsto \hat{\Theta}$

☑ US Senate Net ⟶ p: # of senators
↘ n: # of votes in DB

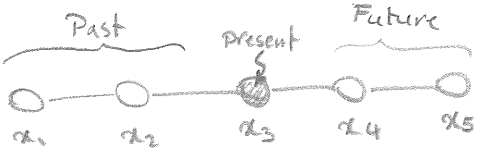
- with iid sample

$$\ell(\Theta; X_1^n) = -\frac{1}{n}\sum \log Q(\cdot) = \log Z(\Theta) - \sum_s \hat{\mu}_s \theta_s - \sum_{(s,t)} \hat{\mu}_{st}\theta_{st}$$

$\hat{\mu}$: from data ← empirical count

- depends on Z : partition function
- for gaussian case ⟶ covariance selection : log-determinant program
- generaly use psudo-likelihood ⟶ fake likelihood
- for tree : problem of ML ⇒ Max Spanning Tree

☑ Markov Chain :

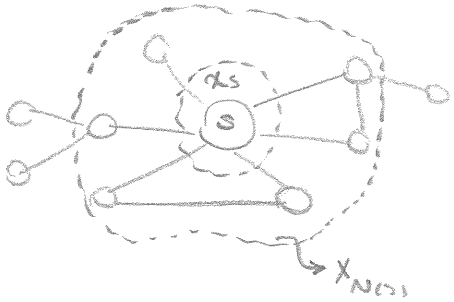


$X_{Past} \perp\!\!\!\perp X_{Future} \mid X_3$

if we cut X_3 ⟶ the graph breaks ⟶ cutset

if we remove cutset ⟶ the two partitions should be independant

- what kind of cutset would be useful?



- reduce to neighborhood selection

$N(s) = \{t \in V \mid (s,t) \in E\}$

$N(s)$ ⟶ is a cut set

$X_{N(s)} = \{x_t \mid t \in N(s)\}$ ⟶ Markov Blanket

- Finding Cutsets,
  - if we know the size ⟶ Naive way = test all sets of nodes, cut them and check computational independance
  - advanced papers
- Graph Selection via neigh. regression

$$\hat{\Theta}[s] = \arg\min_{\Theta} \left\{ -\frac{1}{n}\sum \text{local log likelihood} + \lambda \times \text{regularization} \right\}$$

$$\ell_1\text{-norm} = \|\Theta\|_1 = \sum_{j=1}^{p} |\theta_j| \longrightarrow \text{induce sparsity} \checkmark \longrightarrow \begin{cases} \text{few large numbers} \\ \text{lots of zeros} \end{cases}$$

it could be logistic regression

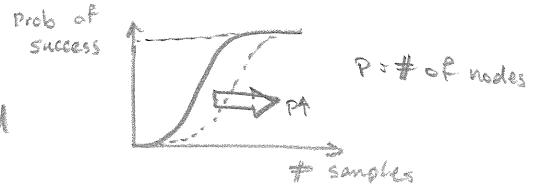
$$\Theta(X_s = 1 \mid X_{V\setminus s}) = \frac{\exp\left(\sum_{X_{N(s)}} \theta_{st} x_t\right)}{1 + \exp(\sim)}$$

binary model
conditional likelihood

Markov property: $X_{V\setminus s} = X_{N(s)}$

- bigger graphs need more samples → harder to estimate more edges

star graph → show method works
→ how many samples do we need



degree ↑ ⟹ more samples needed

d = maximum degree, p = graph size, n = # samples

For guarantee → no false positive
→ no false negative (no unfound edge)

$$\Rightarrow \frac{n}{d^3 \log p} > c\text{-}t$$

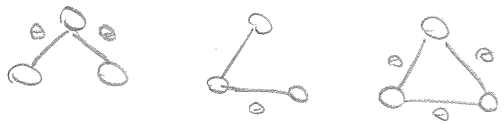
$$\Rightarrow \lambda_n \geq c_1 \sqrt{\frac{\log p}{n}}$$

with prob greater than $1 - 2e^{-c_2 n \epsilon_n^2}$

- Challenges of Graph Selection

- Lower bound on edge weights ⟹ SNR big enough
- Need upper bound on $|\theta_{st}|$

☑



$(x_1, x_2, x_3) \in \{-1, +1\}^3$

$n \to +\infty$

these three graph will be similar because of large weights

- Graph selection as channel coding

- environment give you the info via noisy samples
- $G_{p,d}$: all graphs on p vertices with max-degree channel

$$\theta_{min} = \min_{(s,t) \in E} |\theta_{st}|$$

$$\omega(\theta) := \max_{s \in V} \sum_{t \in N(t)} |\theta_{st}|$$

graph size ↑ ⟹ # of samples needed ↑↑↑ exponentially

$$n < \max\left\{\frac{\log p}{2\theta_{min} \tanh(\theta_{min})}, \dots\right\} \Rightarrow \omega(\theta) \geq d\theta_{min} \to \theta_{min} \leq \frac{1}{d}$$

$n \gtrsim d^2 \log p \leftarrow \simeq \frac{\log}{\cdot \theta_{min} \epsilon} \to$ no really small $\theta_{min} \iff \theta_{min} \asymp \sqrt{\frac{\log p}{n}}$

High Dimensional Sampling : (Vempala , GaTech)

* Problems in High Dimension

  - Optimization
  - Integration
  - Sampling
  - Rounding
  - Learning

* High D Problem

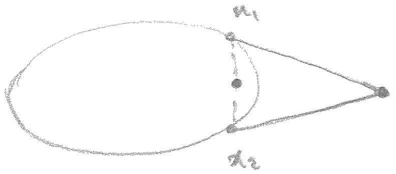
  - Hard ⟶ intractable for ⟶ linear func.
    ↘ polytope & quadratic func.

  - NP ⟶ is there a solution or not?
    #P ⟶ what is the number of solution

  - Ellipsoid Algorithm ⟶ Optimization for Convex sets
    DFK Algorith ⟶ Integration

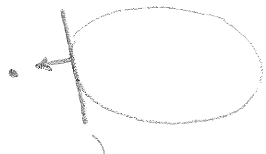
  - How to specify a convex set?
    - explicit list of constraints
    - set of positive semi definite matrices ?
    - a point that is not in the set ⟶ there is a hyper plane that seperate those ⟶ there is a unique point (in set) that is closest to the point

برهان خلف یکتایی نقطه : اگر x_1 و x_2 هم فاصله باشند، نقطه $\frac{x_1+x_2}{2}$ (in set) از آنها نزدیکتر است.



$x \succcurlyeq 0$

$v^T x v < 0 \Rightarrow x . vv^T < 0$

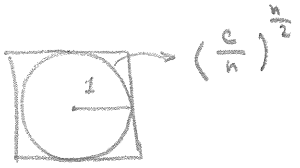


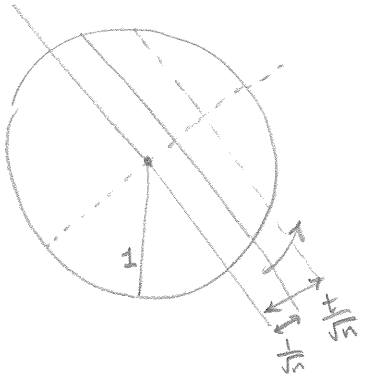
seperator half space

  - Majority Algorithm
    - input $x_1, \dots, x_k$ ⟶ x_{k+1} what should we decide?
      $f(x_1), \dots, f(x_k)$ ⟶ ?
    - majority of previous one? ✗
    - nearest neighbor alg ✓
    - set a hyper plane $S_k = \{ w \mid \forall i , w^T (\text{sgn}(x_i) x_i) > 0 \}$ ⟶ err $\leq bn$ ✓ ; set is large ✗

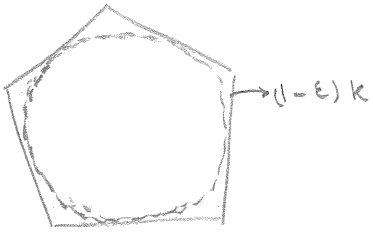
- Random Algorithm → Choose a point and set hyperplane accordingly
  → $\mathbb{E}$(# wrong guesses) $\leq bn$ → # of bits of sample

- Volume of sampling area





$\text{Vol}(\text{outside } \frac{t}{\sqrt{n}}) \leq 2e^{-\frac{t^2}{2}}$



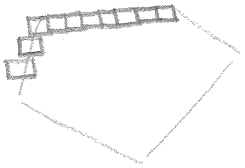
- Brunn Minkowski Inequality

$$\left(\prod^{n}(a_i+b_i)\right)^{\frac{1}{n}} \geq \left(\prod a_i\right)^{\frac{1}{n}} + \left(\prod b_i\right)^{\frac{1}{n}}$$

because : $\frac{1}{n}\sum x_i \geq \left(\prod x_i\right)^{\frac{1}{n}}$

$$1 \geq \left(\prod \frac{a_i}{a_i+b_i}\right)^{\frac{1}{n}} + \left(\prod \frac{b_i}{a_i+b_i}\right)^{\frac{1}{n}}$$

→ other proof by taking log

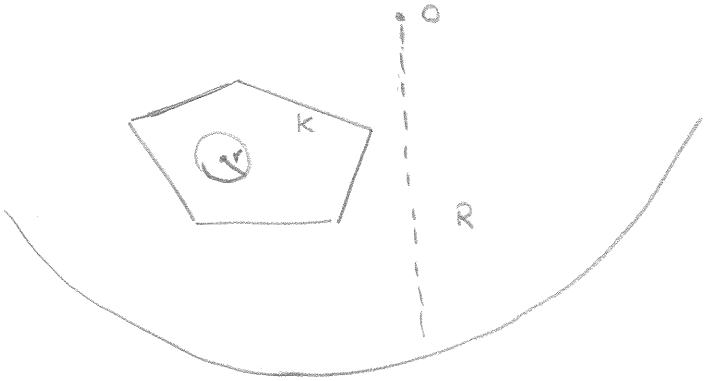


- Convex Feasability

  K is isotropic

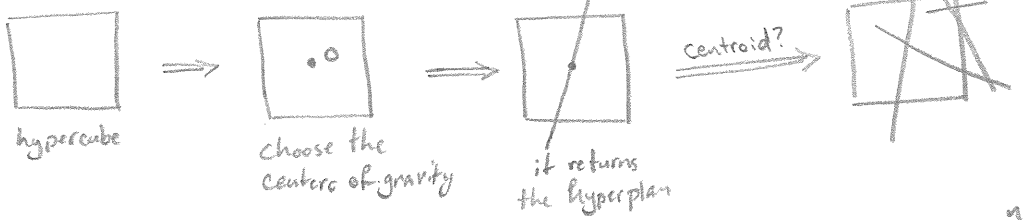
  if $E_K(X)=0$

  $E_K(XX^T)=I$



- How to choose oracle querie

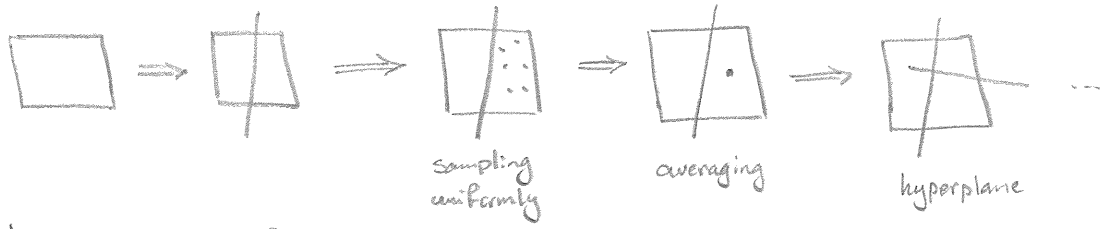
— naïve



start with R^n and ends with r^n → how many steps? $\log(\frac{R}{r})^n = n\log\frac{R}{r}$

we don't know how to cakulate centroids ... $\neq P$ solutions

- Bertsimas



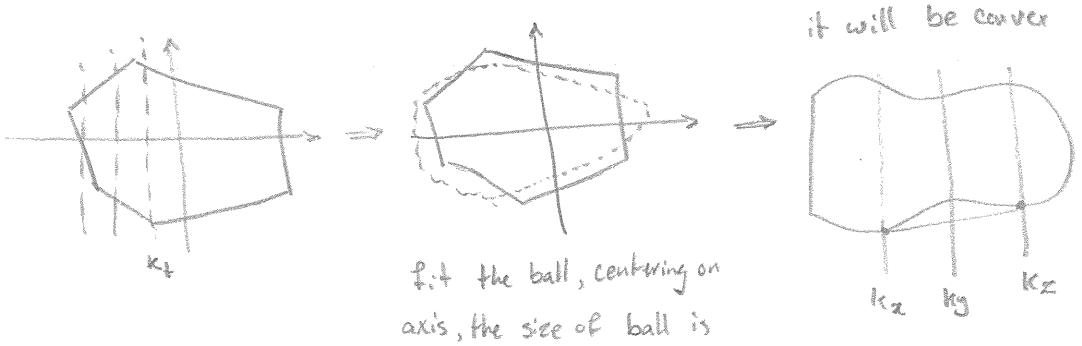
how many samples? are we doing any progress?

- Centroid Alg

↳ cut vol by a constant fraction (at least $\frac{1}{e}$) $\Rightarrow \text{vol}(K \cap H^+) \geq \frac{1}{e}\text{vol}(K)$

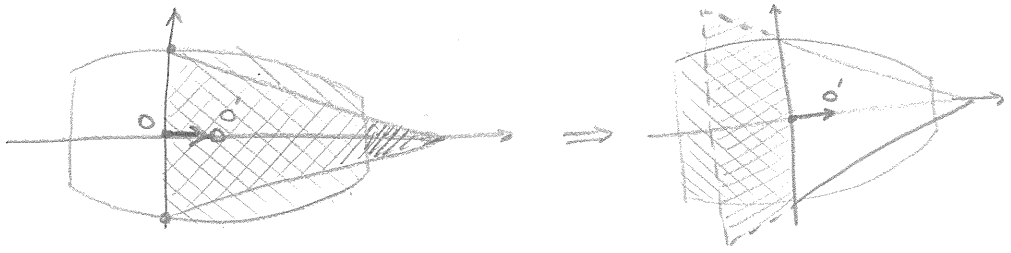
↳ number of steps $\log_{\frac{e}{e-1}}\left(\frac{R}{r}\right)^n = O(n \log \frac{R}{r})$

↳ centroid cuts are balanced



$z \geq \frac{x+y}{2} \Rightarrow Y_z \geq \frac{Y_x + Y_y}{2}$

$\text{vol}(K_z)^{\frac{1}{n-1}} \geq \frac{1}{2}\text{vol}(K_x)^{\frac{1}{n-1}} + \frac{1}{2}\text{vol}(K_y)^{\frac{1}{n-1}} \xrightarrow{f_{(n-1)}\, r_z^{n-1}} Y_z \geq \frac{1}{2}Y_x + \frac{1}{2}Y_y$



make the triangle in away that the area is equivalent to the shape $\Rightarrow$ shift center of gravity right

extend lines so the left part equals the tails of triangle $\Rightarrow$ again shift the center right

↳ how many sample?

- * Volume Estimation
  - it is told that we have a unit ball, a convex set is contains in that. you should guess the volume.
    all you can do is you can ask if a point is in the ball or not.
    the convex hull of points is the answer ⟶ number of questions required is required

• volume from sampling

K_0 = ball

$\vdots$

$K_m = K$

$\frac{vol(K_{i+1})}{vol(K_i)} \sim$ small

claim $vol(K_{i+1}) \leq 2\, vol(K_i)$

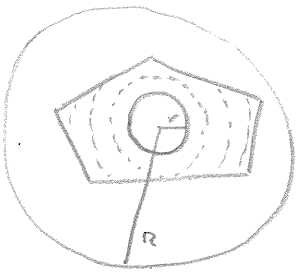
$$vol(K) = vol(B) \times \frac{vol(K_1)}{vol(K_0)} \times \cdots \times \frac{vol(K_m)}{vol(K_{m-1})}$$

$$K_{i+1} = K_i \cap 2^{\frac{i+1}{n}} B \subseteq 2^{\frac{1}{n}} (K \cap 2^{\frac{i}{n}} B)$$

$$vol(K_{i+1}) \leq vol\left(2^{\frac{1}{n}}(K \cap 2^{\frac{i}{n}} B)\right)$$

$$= vol(2^{1/n} K_0) = 2\, vol(K_0)$$

total # of sample $= m \times \frac{m}{\epsilon^2} = O^*(n^2)$



Density Ratio Estimation & (Sugiyama - Titech)

* Problem
  - multiple prob. dist
  - Solving → no need to individual prob
    → $r(x) = \frac{P(x)}{q(x)}$
  - Vapnik's principle : should not solve a more general problem as intermediate step
  - Kernel least-squares (kLS) needed only!
    - importance sampling
    - KL divergence estimation
    - Mutual information
    - Conditional prob. estimation

* Density Ration Estimation Method:
  - problem: estimate r from two set of iid samples
  - Naive
    1. density estimation
    2. ratios of densities

    ⊨ seperated steps ×
  - Probablistic Classification
    - seperate numerators and denominator samples by a probablistic classifier
    - using logistic regression → model correct ⇒ achieve min
    - not reliable for misspecified models
    - parallel processing
  - Moment Matching
    - Match moments of $\hat{P}_{nu}$ and P_{nu} → $\hat{P}_{nu} = \hat{r}(x) P_{de}(x)$
      → better to do in higher dimensions
      → computation ↑
      → need kernel trick → work well if σ of gaussian kernel is good
    - In practice : using QP for optimization
    - Setting σ ① → Median distance between samples ✓ → fails in multimodal

- Density Fitting :
  - KL Importance → Minimize KL divergance from $P_{nu}(x)$
  - Gradient Ascent on KL → project onto feasible region
- Density Ratio Fitting :
  - Minimize Squared-loss
    - ↳ like regression, but different cause « r_n should be sampled
  - Solutions → Constraint LSIF → constraints → non-negativity → ℓ_1 norm
    - → Can compute without QP Solvers
    - → unconstrained LSIF → constraint → ℓ_2 norm

Usage :

- learning under Covariate shift → train / test dist. different
  - → target function fix
- Covariance Shift → $n \to \infty$ Can use ordinary least sq.
  - → $n \downarrow$ not consistent
  - ↳ use importance weighting → importance weighing Least Squares
    - ↳ bias-var problem → flattened importance weight
- Model Selection → AKaike info. Criterion
  - → Subspace info. "
  - → Cross Validation

Computational Methods for Data Analysis 8 (Kutz - U of Washington)

* Data
  - for sciences & non-scientific applications (e.g. data mining)
  - data drives the revolutions → complex dynamical systems
    → purely data driven things ☑ Marketplace
    ☑ Mining DBs for information
  - broad range of ways to tackle data
    - signal processing
    - statistical methods → time and frequency analysis
    - dimensionality reduction & PCA
    - machine learning → integrating dynamics
      → data collection
      → teach computer how to see bigger trends in data
      → construct models to predict future states

* Overview of Course
  - time frequency analysis
    - noisy signals in time
    - representing them in time & frequency
    - prediction
  - machine learning
    - computer science + statistics → pattern recognition
    - data features
    - singular value decomposition
  - complex dynamical systems
    - dimension reduction
    - equation free modeling
    - dynamic mode decomposition

* Time - Frequency Analysis

↳ Fourier Transform
↳ Wavelets

• Fourier Transform

– idea: I can take a given function and I can represent any function as a sum of cosine and sines.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx) \qquad x \in [-\pi, \pi]$$

– how to compute these a_i and b_i?

↳ mulliply both sides to $\cos mx$ → then integrate it from $-\pi$ to π

$$\int_{-\pi}^{\pi} f(x) \cdot \cos mx \cdot dx = \frac{a_0}{2} \int_{-\pi}^{\pi} \cos mx \cdot dx + \sum_{n=1}^{\infty} a_n \int_{-\pi}^{\pi} \underbrace{\cos nx \cos mx \, dx}_{\begin{array}{l}\text{if } n \neq m \to 0 \\ \text{if } n = m \to \pi\end{array}} + b_n \int_{-\pi}^{\pi} \underbrace{\sin nx \cdot \cos mx \, d}_{\begin{array}{l}n, m \in \mathbb{Z} \\ \sin nx \cdot \cos mx = 0\end{array}}$$

orthogonal function

↳ orthogonality in function space → $\cos 1x, \cos 2x, \ldots$ are directions in func. space orthogonal to $\sin 1x, \sin 2x, \sin 3x, \ldots$

↳ $a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cdot \cos nx \cdot dx$

↳ $b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cdot \sin nx \cdot dx$

– Complex representation

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{i n \frac{\pi}{L} x} \qquad x \in [-L, L]$$

$$c_n = \frac{1}{2L} \int_{-\pi}^{\pi} f(x) \cdot e^{-i n \frac{\pi}{L} x} \cdot dx$$

– Fourier says that we can represent a function as sines and cosines

↳ we think of $f(x)$ as a time signal.

↳ we care about coefficients → $n=1 : c_1$, $n=2 : c_2 \ldots$

↳ $n=1 \rightarrow a_1 \rightarrow \cos \alpha \rightarrow$ has a frequency

↳ take a signal in time and show it as a collection of frequencies components

— MATLAB

we have a function u defined (as vector)

transform to Fourier domain `ut = fft(u)` ⇝ fast fourier transform

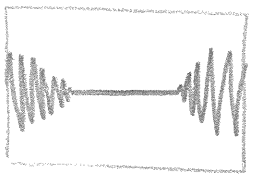
fft → late 60's by Tookie & Cooley

↘ reduce the complexity of FT from $O(N^2)$ to $O(N \log N)$

↘ using an observation → correlation between coefficients → break a problem of size N into 2 problems of size $\frac{N}{2}$

↳ break problem into N problem of size 1

the glue it together using an algorithm called butterfly alg.



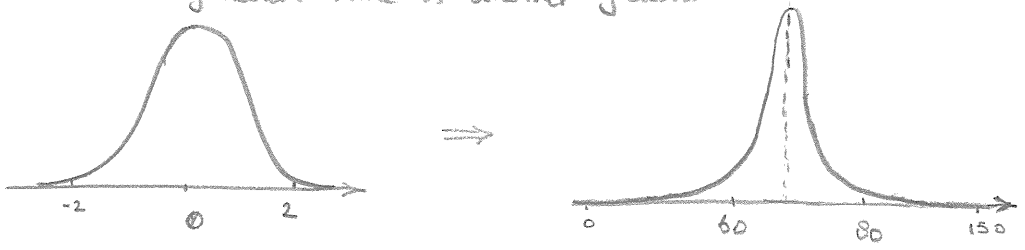
→ butterfly

↳ to fix it, we need to shift it, flip the sides and multiply other components by -1

or use ~~fftshift~~ and absolute value of it.

```
ut = fft(u)
plot(abs(fftshift(ut)))
```

☑ FT of a guassian func is another guassian



→ this axis is frequency but we didn't order this

↳ matlab thinks we're working on 2π periodic domain

but we work on L period!

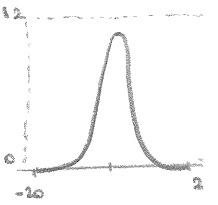
and we should rescale → $\frac{2\pi}{L}$

↳ $[0 : \frac{n}{2}-1 \quad -\frac{n}{2} : -1]$ → shifted integers (because fft is shifted too)

so

$k = \frac{2\pi}{L} \times [0 : \frac{n}{2}-1 \quad -\frac{n}{2} : -1]$

`plot(fftshift(k), abs(fftshift(ut)))` } ⇒ Spectral Content of u



- Spectral Content

↳ how much energy is at cosine of x? ...

↳ tells us the strength of all of the freq. content

- always have it in mind that it is shifted

• Derivative Relations

$$\widehat{f^{(n)}} = (ik)^n \hat{f}$$

$\hat{f}$ → represents Fourier transform

$f^{(n)}$ → n^{th} derivative

- to find 3rd derivative of f

1. Fourier transform it
2. multiply it by $(ik)^3$
3. inverse transform the result!

- if the boundary conditions are satisfied, this is one of the most accurate ways to cale derivatives

```
ujds = ifft((i*k).^j.*ut);
```

→ j^{th} derivative of function u using spectral approximation

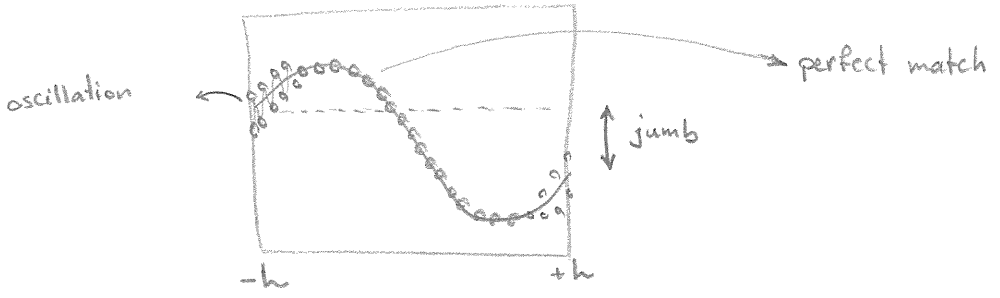
↳ it's accurate to the 10^{-6} magnitude

- if the domain is smaller, e.g. $L = 20 \rightarrow L = 4$

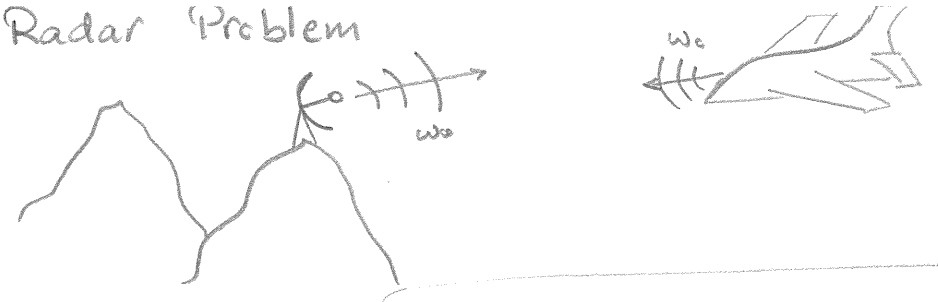
↳ violates the conditions of boundary

↳ it supposed to be 2π periodic function, but there will be a jump between the value of $u(-L)$ and $u(L)$ if L is small.

↳ result has a "Gibbs's phenomenon" - oscillations near the jump

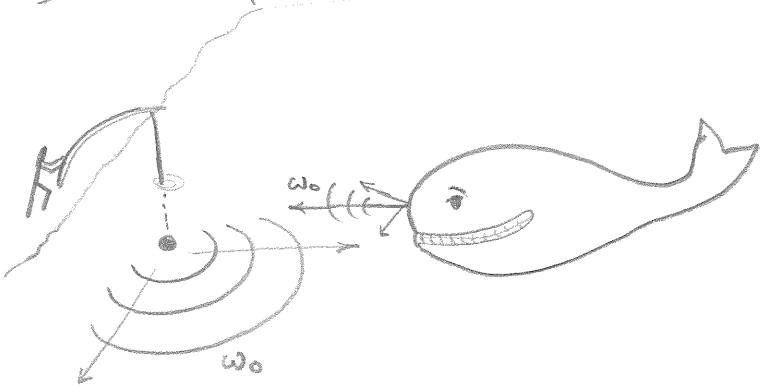


* Radar Problem



Radar

Sonar



- Transmitter sends signal with ω_0

  Jets try to absorb that signal to be stealth

  Planes reflects signal with freq. ω_0

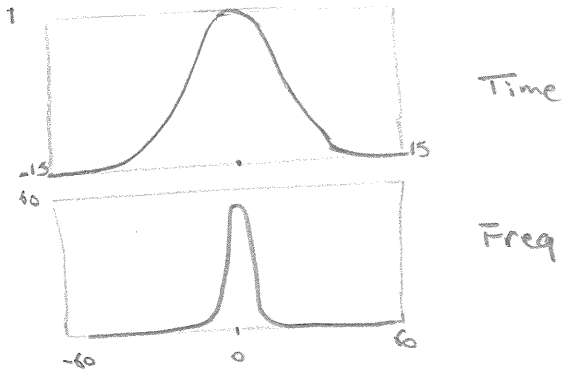
  AVAX planes flood the area with ω_c freq to cover the jets from being identified

  There all lots of electromagnetic noise

- Heisenberg Uncertainty Principle

  - There's a relationship between the width of the pulse in time and the width of the pulse in frequency.

    ↳ width of time × width of freq = 2π



  ↳ Quantum Mechanics was built on this idea

    ↳ you can represent the state / function of a particle in terms of its probability density and a FT of that represent its momentum.

      ↳ you can never know the momentum and position at the same time because there's an uncertainty relationship between them

        ↳ width of the momentum and width of the location have to be related ($= 2\pi$)

  - narrow signal in time ⟶ wide in freq: lots of freq components

    broad in time ⟶ becomes almost a single frequency

- White Noise

  - white light = collection of all colors
  - white noise = collection of all frequencies

Sequential Monte Carlo 8 (Doucet - Oxford)

* State - Space Model

- latent Markov process ——> no time dependant X_1
- observation Y_t
- HMM $X_t = \Psi(X_{t-1}, V_t)$ $Y_t = \Phi(X_t, W_t)$
- Aim : infer X_t given observations Y_t
- Inference

$$X_1 \sim \mu \qquad X_t \mid (X_{t-1} = x) \sim f(\cdot|x)$$

$$Y_t \mid (X_t = x) \sim g(\cdot|x)$$

$$p(y_{1:t}) = \int \cdots \int p(x_{1:t}, y_{1:t})\, dx_{1:t}$$

- MC Basics

$$\hat{p}(x_{1:t} \mid y_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{1:t}^{(i)}}(x_{1:t})$$

- integration & marginalization is easy
- Problem ——> sampling from non-Gaussian non-linear models
  - even if we can, it takes $O(t)$
- Solution ——> use MCMC for sampling approximately
- SMC Methods —— break Problem of sampling to easier one

- Bayesian Recursion

$$p(x_t \mid y_{1:t-1}) = \int p(x_{t-1}, x_t \mid y_{1:t-1})\, dx_{t-1}$$

$$= \int f(x_t|x_{t-1})\, p(x_{t-1} \mid y_{t-1})\, dx_{t-1}$$

معادله شکن - کارکرد

- Bayesian Recursion on Path Space

* typos

p 47 ——> $p(x_t \mid y_t, x_{t-1}) = \dfrac{f(\cdot)\, g(y_t|x_t)}{p(y_t|x_{t-1})}$

Submodularity & (Bach - Inria)

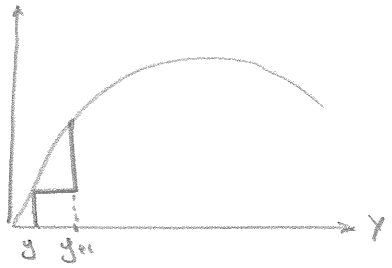
* Problem

- F is submodular iff $F : 2^V \longrightarrow \mathbb{R}$

$$\forall A, B \subset V, \quad F(A) + F(B) \geqslant F(A \cap B) + F(A \cup B)$$

$$F(\emptyset) = 0 \longrightarrow \text{submodular}$$

- other definition



- enough to show that

take $A \longrightarrow$ Add one item to A

$$\forall A, k, j \qquad F(A \cup \{k,j\}) + F(A) \leqslant F(A \cup \{j\}) + F(A \cup \{k\})$$

- Closedness Properties

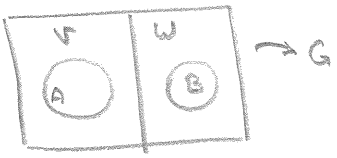
– A & B : submodular $\longrightarrow$ linear combination

Marginalizatio $\quad A \longmapsto F(A \cap B)$

Conditioning $\quad A \longmapsto F(A \cup B) - F(B)$

- Partial Minimization

$$F(A) = \min_B G(A \cup B) - \min_B G(B)$$



- Modular function

$$S(A) = s^T I_A \longrightarrow \text{indicator vector of } A$$

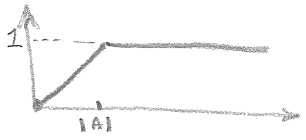
$$S(A) = \sum_{k \in A} s_k$$

$$s = I_A \longrightarrow S(A) = |A|$$

– $F : A \longmapsto g(S(A))$ is sub modular if g is a concave function

$$F(A) = g(|A|)$$

proof: $g(|A|+2) + g(A) \leqslant g(|A|+1)\, g(|A|+1) \longrightarrow g(k+1) - g(k) \geqslant g(k+2) - g(k+1)$



* Matroids
  - Essentially a column of matrix ⟶ independent sets
  - Rank Function ⟶ biggest sub matrix in matroid that has independence

Foundation of Computer Graphics 8 (Ramamoorti - Berkley)

* Technical Challenges
  - Put object on screen correctly in its position
  - How to draw surfaces
  - How to light an object correctly
    - Highlights on it
    - Smooth shading
  - How are objects lit in real world?
  - Build Softwares and Hardwares

* 3D Graphics Pipeline

1. Modeling : Creating geometric models of object
   - Simple object ☑ Teapot
   - Complicated Mesh ☑ Cat Sculpture
   - Million Polygon Mesh ☑ Michealangelo's David
2. Animation : Motion of character
3. Rendering : Creating realistic images, given the geometry and animation
   - Simulate the way light propagate on scene → realistic & intricate shadows

* Rasterize vs Ray Trace
  - Rasterization: Goes through all the geometric primitives and determines where in the screen they should go
  - Ray Tracing: Goes to each point / pixel in the screen and determines which geometric primitive that corresponds to.

RECEIPT

Paid by : Meshgi Kourosh
(meshgi-k@sys.i.kyoto-u.ac.jp)
Kyoto University

Paid to : Machine Learning Summer School'12 in Kyoto

MLSS12 Secretariat Office,

Graduate School of Informatics, Kyoto University

Research Bldg. No.1/Project Lab, Room 303,

36-1 Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 JAPAN

+81 (0)75-753-5911

The sum of: 40,000 JPY **Date:** August 23, 2012

corresponding to:

Description	Quantity	Unit price	Amount
Registration for the MLSS'12 in Kyoto, August 27 - September 7 2012 - Student, 2 weeks	1	40,000 JPY	40,000 JPY

Total: 40,000 JPY (Tax: 0 JPY)

Terms and conditions

The registration includes: access to all the lectures publicized on the MLSS schedule, light snacks during coffee breaks between lectures, banquet and MLSS party event described in the "Social Events" section of the MLSS website.

機械学習
サマースクール
京都２０１２

	Mon. 27	Tue. 28	Wed. 29	Thu. 30	Fri. 31
8:30 - 10:10	**Opening**	**Domingos** *Stat. Rel. Learn.*	**Vandenberghe** *Convex Optim.*	**Vandenberghe** *Convex Optim.*	**Lin** *ML Software*
10:30 - 12:10	**Rakhlin** *Stat. Learning*	**Rakhlin** *Stat. Learning*	**Vandenberghe** *Convex Optim.*	**Müller** *Brain C. I.*	**Lin** *ML Software*
	Lunch Break				
13:50 - 15:30	**Rakhlin** *Stat. Learning*	**Tsuda** *Graph Mining*	**Tsuda** *Graph Mining*	**Müller** *Brain C. I.*	**Schapire** *Boosting*
15:50 - 17:30	**Domingos** *Stat. Rel. Learn.*	**Tsuda** *Graph Mining*	**Müller** *Brain C. I.*	**Schapire** *Boosting*	**Schapire** *Boosting*
17:50 - 19:30	**Domingos** *Stat. Rel. Learn.*	**Poster I** *Intern. Hall I&II*	**Doya** *Brain & R. L.*	**Poster II** *Intern. Hall I&II*	**Okada** *ML & Med. Im.*

	Mon. 3	Tue. 4	Wed. 5	Thu. 6	Fri. 7
8:30 - 10:10	**Wainwright** *Graphical Models*	**Blei** *Topic Models*	**Blei** *Topic Models*	**Vempala** *High-dim. Sampling*	**Fukumizu** *Kernel Methods*
10:30 - 12:10	**Wainwright** *Graphical Models*	**Blei** *Topic Models*	**Vempala** *High-dim. Sampling*	**Fukumizu** *Kernel Methods*	**Fukumizu** *Kernel Methods*
	Lunch Break				
13:50 - 15:30	**Doucet** *Seq. Monte Carlo*	**Doucet** *Seq. Monte Carlo*	**Vempala** *High-dim. Sampling*	**Bach** *Submodularity*	**Bach** *Submodularity*
15:50 - 17:30	**Doucet** *Seq. Monte Carlo*	**Wainwright** *Graphical Models*	**Takemura** *Holonomic Grad.*	**Bach** *Submodularity*	**Sugiyama** *Density Ratio*
17:50 - 19:30	**Poster III** *Intern. Hall I&II*	**Amari** *ML & Info. Geom.*	**Banquet**	**Iwata** *Submodular Optim.*	