Summary

Lucas Swiniarski
NetID : ls4411
lucas.swiniarski@nyu.edu

Contents

Part I
Various contents

1 Mathematics

1.1 Analysis

1.1.1 Convexity

$f : X \in \mathbb{R}^n \mapsto \mathbb{R}$ convex iif :

$$f(y) - f(x) \geq \nabla f(x) \cdot (y - x) \tag{1}$$
$$\nabla^2 f(x) \geq 0 \tag{2}$$

1.2 Inequalities

$$\text{Markov's inequality} : P(X \geq a) \leq \frac{E[x]}{a} \begin{cases} X \geq 0 \\ a \geq 0 \end{cases} \tag{3}$$

$$\text{Bienaymé-Tchebychev} : P(\|X - E[x]\| \geq \alpha) \leq \frac{\sigma^2}{\alpha^2} \tag{4}$$

$$\text{Tchébychev} : P(\|X\| \geq \alpha) \leq \frac{E[\|X\|^p]}{\alpha^p} \tag{5}$$

$$\text{Caushy-Shwartw} : |\mathbf{E}[XY]| \leq \sqrt{\mathbf{E}[X^2]\mathbf{E}[Y^2]} \tag{6}$$

$$\text{Hoeffding} : \forall k \in [1..n], P(a_k \leq X_k \leq b_k) = 1 \tag{7}$$

$$X_k \text{ indépendent}, S_n = \sum_{i=1}^{n} X_i \tag{8}$$

$$P(S_n - E[S_n] \geq t) \leq \exp -\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)} \tag{9}$$

$$\text{Jensen} : f \text{ convex}, f(E[X]) \leq E[f(X)] \tag{10}$$

1.3 Probability

Identity

$$\text{If} : \forall \varepsilon, P(X \geq \varepsilon) \leq f(\varepsilon), \delta > 0 \tag{11}$$

$$P(X \leq f^{-1}(\delta)) \leq 1 - \delta \tag{12}$$

1.3.1 Conditional Independence

A, B conditionally indepent given C if :

$$P(A|B \cap C) = P(A|C) \tag{13}$$

$$P(A|B, C) = P(A|B \cap C) \tag{14}$$

$$P(A \cap B|C) = P(A|C)P(B|C) \tag{15}$$

1.4 Information Theory

Entropy :

$$H(X) = -E[log(X)] = -\sum_{x \in X} p(x) log(p(x)) \tag{16}$$

Properties :

- Measure of uncertainty of X.
- $H(X) \geq 0$.
- Maximal for uniform distribution.

Relative Entropy : or Kullback-Leibler divergence

$$D(p||q) = E_p[log(\frac{p(X)}{q(X)})] = \sum_{x \in \mathcal{X}} p(X) log \frac{p(X)}{q(X)} \tag{17}$$

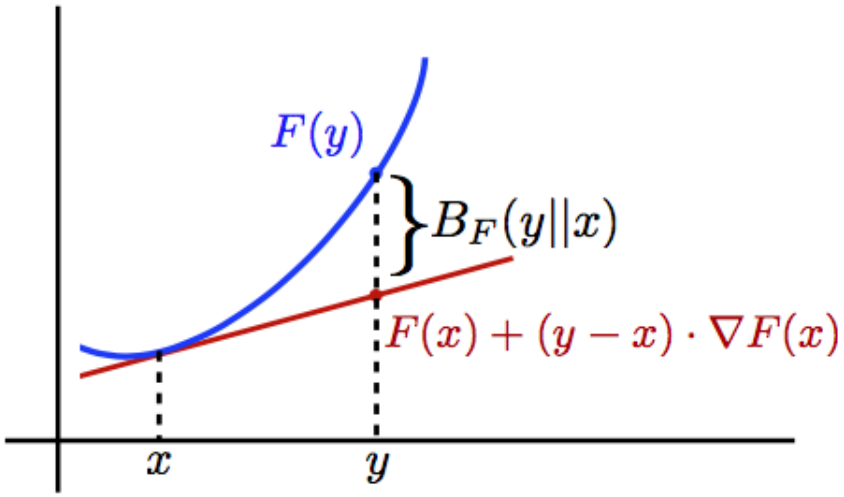
Properties :

- Asymmetric.
- Non-negative.
- Definite.

Bregman Divergence : let F be a convex and differentiable function defined over a convex set C in a Hilbert space $\mathbb{H}$. Then, the Bregman divergence BF associated to F is defined by :

$$B_F(x||y) = F(x) - F(y) - \langle \nabla F(y), x - y \rangle \tag{18}$$

Figure 1: Bregman Divergence



Conditional relative Entropy : Let p and q be two probability distributions over$\mathcal{X} \times \mathcal{Y}$. Then, the conditional relative entropy of p and q with respect to distribution r over $\mathcal{X}$ is defined by :

$$\mathop{\mathbb{E}}_{X \sim r}[D(p(\cdot|X)||q(\cdot|X))] = \sum_{x \in \mathcal{X}} r(x) \sum_{y \in \mathcal{Y}} p(y|x) log \frac{p(y|x)}{q(y|x)} \tag{19}$$

$$= D(p||q) \tag{20}$$

$$p(x,y) = r(x)p(y|x) \tag{21}$$

Information gain : In general terms, the expected information gain is the change in information entropy H from a prior state to a state that takes some information as given:

$$IG(T,a) = H(T) - H(T|a) \tag{22}$$

The expected value of the information gain is the mutual information.

Mutual information : Intuitively, mutual information measures the information that X and Y share: it measures how much knowing one of these variables reduces uncertainty about the other.

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{23}$$

$$= E(D(X||Y)) \tag{24}$$

1.5 Concentration of Measures - SVD in-depth

From [13].

Singular Value Decomposition :

$$\text{Given : } M \in \mathcal{M}_{m,n}(\mathbb{R}) \tag{25}$$
$$\exists U \in \mathcal{M}_{m,m}(\mathbb{R}) \text{ Unitary.} \tag{26}$$
$$\exists \Sigma \in \mathcal{M}_{m,n}(\mathbb{R}) \text{ Diagonal.} \tag{27}$$
$$\exists V \in \mathcal{M}_{n,n}(\mathbb{R}) \text{ Unitary.} \tag{28}$$
$$M = U\Sigma V^* \tag{29}$$

Principal Component Analysis. Statistical procedure : orthogonal transformation converting a set of observations possibly correlated values into a set of values of linearly uncorrelated variables called principal components.

$$\text{Given : } X \in \mathcal{M}_{n,p}(\mathbb{R}) \text{ zero-mean columns.} \tag{30}$$
$$w_{(k)} = (w_1, \cdots, w_n)_{(k)} \tag{31}$$
$$t_{(i)} = (t_1, \cdots, t_n)_{(i)} \tag{32}$$
$$t_k(i) = x_{(i)} \cdot w_{(k)} \tag{33}$$

Transform the p features in a new space such that each new feature will have the maximum variance of x, given w unit vector.

$$w_{(1)} = \underset{\|w\|=1}{\text{argmax}} \left\{ \sum_i (t_1)^2_{(i)} \right\} \tag{34}$$
$$= \underset{\|w\|=1}{\text{argmax}} \left\{ \sum_i (x_{(i)} \cdot w)^2 \right\} \tag{35}$$
$$\hat{X}_k = X - \sum_{s=1}^{k-1} X w_{(s)} w^T_{(s)} \tag{36}$$

(Fast) Randomized SVD :

$$\text{Given : } A \in \mathcal{M}_{m,n}(\mathbb{R}) \tag{37}$$
$$Q \in \mathcal{M}_{m,p}(\mathbb{R}) \tag{38}$$
$$\tag{39}$$

If A has a lot of rows, then we cannot compute its Singular Value Decomposition. The trick is to find Q with few columns such that :

$$A \approx QQ^T A \tag{40}$$
$$Q^T A \in \mathcal{M}_{p,n}(\mathbb{R}) \tag{41}$$
$$Q^T A = S\Sigma V^T \tag{42}$$
$$A \approx QS\Sigma V^T \tag{43}$$

Fast randomized SVD algorithm : We need to estimate the range of matrix A. Take $\Omega_1, \cdots, \Omega_p$ random vectors and look at the subspace formed by the action of A. $Y = A\Omega = QR$, Q orthonormal vectors, R triangular superior, we just found a good Q !

Some issues arise with finite precision arithmetic : we can drive the specter of Y down.

$$A \to \Sigma \tag{44}$$
$$(AA^T)^p A \to \Sigma^{2p+1} \tag{45}$$

The spectrum decays exponentially with this trick, $p = 1, 2$ is enough in practice.

Figure 2: Chernoff inequality

Theorem 2 (Chernoff inequality) Let $X_1, \ldots, X_n$ be independent scalar random variables with $|X_i| \leq K$ almost surely, with mean μ_i and variance σ_i^2. Then for any $\lambda > 0$, one has

$$\mathbf{P}(|S_n - \mu| \geq \lambda\sigma) \leq C \max(\exp(-c\lambda^2), \exp(-c\lambda\sigma/K)) \quad (11)$$

for some absolute constants $C, c > 0$, where $\mu := \sum_{i=1}^n \mu_i$ and $\sigma^2 := \sum_{i=1}^n \sigma_i^2$.

Figure 3: Azuma inequality

Theorem 3 (Azuma's inequality) Let $X_1, \ldots, X_n$ be a sequence of scalar random variables with $|X_i| \leq 1$ almost surely. Assume also that we have the martingale difference property

$$\mathbf{E}(X_i|X_1, \ldots, X_{i-1}) = 0 \quad (12)$$

almost surely for all $i = 1, \ldots, n$ (here we assume the existence of a suitable disintegration in order to define the conditional expectation, though in fact it is possible to state and prove Azuma's inequality without this disintegration). Then for any $\lambda > 0$, the sum $S_n := X_1 + \ldots + X_n$ obeys the large deviation inequality

$$\mathbf{P}(|S_n| \geq \lambda\sqrt{n}) \leq C \exp(-c\lambda^2) \quad (13)$$

for some absolute constants $C, c > 0$.

Bounds on the sum of random variables :

Figure 5: Mc Diarmid inequality

Theorem 7 (McDiarmid's inequality) Let $X_1, \ldots, X_n$ be independent random variables taking values in ranges $R_1, \ldots, R_n$, and let $F : R_1 \times \ldots \times R_n \to \mathbf{C}$ be a function with the property that if one freezes all but the i^{th} coordinate of $F(x_1, \ldots, x_n)$ for some $1 \leq i \leq n$, then F only fluctuates by most $c_i > 0$, thus

$$|F(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) -$$

$$F(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)| \leq c_i$$

for all $x_j \in X_j$, $x_i' \in X_i$ for $1 \leq j \leq n$. Then for any $\lambda > 0$, one has

$$\mathbf{P}(|F(X) - \mathbf{E}F(X)| \geq \lambda\sigma) \leq C \exp(-c\lambda^2)$$

for some absolute constants $C, c > 0$, where $\sigma^2 := \sum_{i=1}^n c_i^2$.

Figure 6: Talagrand inequality

Theorem 9 (Talagrand concentration inequality) Let $K > 0$, and let $X_1, \ldots, X_n$ be independent complex variables with $|X_i| \leq K$ for all $1 \leq i \leq n$. Let $F : \mathbf{C}^n \to \mathbf{R}$ be a 1-Lipschitz convex function (where we identify $\mathbf{C}^n$ with $\mathbf{R}^{2n}$ for the purposes of defining "Lipschitz" and "convex"). Then for any λ one has

$$\mathbf{P}(|F(X) - \mathbf{M}F(X)| \geq \lambda K) \leq C \exp(-c\lambda^2) \quad (18)$$

and

$$\mathbf{P}(|F(X) - \mathbf{E}F(X)| \geq \lambda K) \leq C \exp(-c\lambda^2) \quad (19)$$

for some absolute constants $C, c > 0$, where $\mathbf{M}F(X)$ is a median of $F(X)$.

Bounds on functions of random variables

1.6 Random Projections and Compressed sensing

From [11].

Figure 4: Recap of various bounds depending on assumptions

- The zeroth moment method bound (1), which requires no moment assumptions on the X_i but is only useful when X_i is usually zero, and has no decay in λ.
- The first moment method bound (2), which only requires absolute integrability on the X_i, but has only a linear decay in λ.
- The second moment method bound (5), which requires second moment and pairwise independence bounds on X_i, and gives a quadratic decay in λ.
- Higher moment bounds (7), which require boundedness and k-wise independence, and give a k^{th} power decay in λ (or quadratic-exponential decay, after optimising in k).
- Exponential moment bounds such as (11) or (13), which require boundedness and joint independence (or martingale behaviour), and give quadratic-exponential decay in λ.

MRI signals are sparse in different transform domains. For example, the brain image is sparse in the wavelet domain. This is good news if we are trying to recover an image from undersampled data. We cannot estimate more than m parameters from m measurements, so in principle it is hopeless to try to estimate an n-dimensional signal from data given by 18. However, if the signal is sparse in some domain and can be parametrized by s coefficients for instance, where $s < m$, then recovery may be possible.

In MRI we solve :

$$\min \|\tilde{c}\|_1 \tag{46}$$

$$\text{Subject to : } y = F_\Omega W \tilde{c} \tag{47}$$

W is an orthonormal wavelet transform.

Theorem 3.1. *Assume there exists a signal $x \in \mathbb{R}^n$ with s nonzeros such that*

$$Ax = y \tag{27}$$

for a random matrix $A \in \mathbb{R}^{m \times n}$ with iid Gaussian entries with zero mean and unit variance. The solution to Problem (106) is equal to x with probability at least $1 - \frac{1}{n}$ as long as the number of measurements satisfies

$$m \geq Cs \log n, \tag{28}$$

for a fixed numerical constant C.

A can be found from the columns of a unitary matrix.

$$U \in \mathcal{C}^{n \times n} \tag{48}$$

$$\mu(U) = \sqrt{n} \max_{i,j} |U_{i,j}| \tag{49}$$

$\mu(U)$ how spiky U is, if rows are too localized it might miss entries (?!?!?).

Lemma 3.2. *Let T be the indices of the nonzero entries in x. If A_T is full rank and there exists a vector $v \in \mathbb{R}^m$ such that*

$$\left(A^T v\right)_i = \text{sign}\,(x_i) \qquad \text{if } x_i \neq 0 \tag{31}$$

$$\left|\left|\left(A^T v\right)_i\right|\right|_\infty < 1 \qquad \text{if } x_i = 0 \tag{32}$$

then x is the unique solution to Problem (106).

Lemma 3.3 (Dual problem)**.** *The dual problem to Problem (106) is*

$$\text{maximize} \quad y^T \tilde{v} \tag{33}$$

$$\text{subject to} \quad \left|\left|A^T \tilde{v}\right|\right|_\infty \leq 1, \tag{34}$$

where the dual variable $\tilde{v}$ has dimension n.

The dual certificate is feasible for this problem and achieves a cost-function value of $||x||_1$, since

$$y^T v = x^T A^T v \tag{35}$$

$$= \sum_{i \in T} x_i \,\text{sign}\,(x_i) \tag{36}$$

$$= ||x||_1 . \tag{37}$$

Proposition 3.4. *Fix a set $T \subset \{1, 2, \ldots, n\}$ such that $|T| \leq s$. For any unit-norm vector x with support T*

$$1 - \epsilon \leq \frac{1}{\sqrt{m}} ||Ax||_2 \leq 1 + \epsilon \tag{41}$$

with probability at least

$$1 - 2\left(\frac{12}{\epsilon}\right)^s \exp\left(-\frac{m\epsilon^2}{32}\right). \tag{42}$$

Setting $\epsilon = 1/2$ implies that the minimum singular value of A_T is lower bounded,

$$\sigma_{\min}(A_T) \geq \frac{\sqrt{m}}{2}, \tag{43}$$

with probability at least $1 - \exp\left(-\frac{Cm}{s}\right)$ for a certain constant C. This implies $A_T^T A_T$ is invertible and consequently that condition (31) is satisfied,

$$(q_{\ell_2})_T = A_T^T A_T \left(A_T^T A_T\right)^{-1} \operatorname{sign}(x_T) \tag{44}$$

$$= \operatorname{sign}(x_T). \tag{45}$$

All is left is to bound q_{ℓ_2} on T^c. We define

Definition 3.10 (Restricted isometry property)**.** *If a matrix M satisfies the restricted isometry property with constant ϵ_s then for any s-sparse vector x*

$$(1 - \epsilon_s) ||x||_2 \leq ||Mx||_2 \leq (1 + \epsilon_s) ||x||_2. \tag{79}$$

Theorem 3.11 (Restricted isometry property for Gaussian matrices)**.** *Let $A \in \mathbb{R}^{m \times n}$ be a random matrix with iid Gaussian entries with zero mean and unit variance. $\frac{1}{\sqrt{m}} A$ satisfies the restricted isometry property with a constant ϵ_s with probability $1 - \frac{C_2}{n}$ as long as the number of measurements*

$$m \geq \frac{C_1 s}{\epsilon_s^2} \log\left(\frac{n}{s}\right) \tag{82}$$

for two fixed constants $C_1, C_2 > 0$.

Restricted Isometry Property It good to remember the Nyquist-Shannon theorem : If a signal has maximum frequency in its decomposition of f, you need only $2f + 1$ samples. In compressed sensing, if signal sparse you need $O(s \log(2f + 1))$.

Part II

First Semester

2 Statistical Mathematics

2.1 Probability Basics

Chain rule :

$$P(\cap_i S_i) = P(S_1)P(S_2|S_1)P(S_3|S_1 \cap S_2)... \tag{50}$$

Law of total probability :

$$A_1, ..., A_n \text{ partition of } \Omega \tag{51}$$

$$P(S) = \sum_i P(S \cap A_i) \tag{52}$$

Bayes' rule :

$$P(A_i|S) = \frac{P(S|A_i)P(A_i)}{\sum_j P(S \cap A_j)} \tag{53}$$

Independence :

$$P(A \cap B) = P(A)P(B) \tag{54}$$

Conditional Independence :

$$P(A \cap B|C) = P(A|C)P(B|C) \tag{55}$$

2.2 Random Variables

Important distributions :

$$^2(n) \tag{56}$$

$$U = \sum_{i=1}^{n} X_i^2, X_i \in \mathcal{N}(0,1) \tag{57}$$

$$f_X(x) = \frac{x^{\frac{n}{2}-1} \exp(-\frac{x}{2})}{2^{\frac{n}{2}} \Gamma(\frac{n}{2})} \tag{58}$$

2.3 Multivariate Random Variables

$$P_{X,Y}(x,y) = P(X = x, Y = y) \tag{59}$$

$$p_X(x) = \sum_y P(X = x, Y = y) \tag{60}$$

$$P_{Y|X}(y|x) = \frac{P_{X,Y}(x,y)}{p_X(x)} \tag{61}$$

Mutually independent RV $X_1, ..., X_n$:

$$P_X(x) = \prod_i P_{X_i}(x_i) \tag{62}$$

Mutually conditionally independent random variables "

$$\mathcal{I}, \mathcal{J} \subset \{1, ..., n\} \tag{63}$$

$$P_{X_\mathcal{I}|X_\mathcal{J}}(x_i|x_j) = \prod_{i \in \mathcal{I}} P_{X_i|X_\mathcal{J}}(x_i|x_\mathcal{J}) \tag{64}$$

Remark :

$$Z = X + Y \tag{65}$$

$$f_Z(z) = \int_{u \in \mathbb{R}} f_X(z-u) f_Y(u) du \tag{66}$$

Gaussian Random Vector :

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma(x-\mu)) \tag{67}$$

Chain rule for discrete and continuous Random Variables :

$$p_D(d)f_{C|D}(c|d) = f_C(c)p_{D|C}(d|c) \tag{68}$$

2.4 Expectation

$$\mathbf{E}[g(X)] = \sum_x g(x)p_X(x) \tag{69}$$

Pearson's correlation coefficient :

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X\sigma_Y} \tag{70}$$

Covariance matrix :

$$\Sigma_{AX+b} = A\Sigma_X A^T \tag{71}$$

Whitening :

$$\Sigma_X = U\Lambda U^T \tag{72}$$

$$\text{Whiten} : \sqrt{\Lambda^{-1}}U^T X \tag{73}$$

Conditionnal expectation and iterated expectation :

$$\mathbf{E}[g(X,Y)|X] = h(X) \tag{74}$$

$$h(x) = \mathbf{E}[g(X,Y)|X = x] \tag{75}$$

$$\mathbf{E}[g(X,Y)] = \mathbf{E}[\mathbf{E}[g(X,Y)|X]] \tag{76}$$

2.5 Random Process

2.5.1 Definitions

Random Process Probability space $(\Omega, \mathcal{F}, P)$, A random process is a function :

$$\begin{aligned} \tilde{X} : \Omega &\to \mathbb{R}^{\mathcal{T}} \\ \omega &\mapsto \tilde{X}(\omega, .) \end{aligned}$$

$\mathcal{T}$ can be discrete or continuous. If we fix ω it's a deterministic function of time. If we fix t it's a random variable. It can be discrete-state or even finite-state RP.

n-th order distribution It's the joint distribution of $\tilde{X}(t_1), \cdots, \tilde{X}(t_n)$.

Strictly stationary process If $\forall n, \forall \tau$, $\tilde{X}(t_1), \cdots, \tilde{X}(t_n)$ and $\tilde{X}(t_1+\tau), \cdots, \tilde{X}(t_n+\tau)$ have the same joint distribution.

Mean

$$\mu_{\tilde{X}}(t) = E[\tilde{X}(t)] \tag{77}$$

Auto-covariance

$$R_{\tilde{X}(t_1,t_2)=Cov(\tilde{X}(t_1),\tilde{X}(t_2))} \tag{78}$$

Weakly stationary process If the mean is constant and $\forall t_1, t_2, \tau$:

$$R_{\tilde{X}(t_1,t_2)=R_{\tilde{X}(t_1+\tau,t_2+\tau)}} \tag{79}$$

$$\text{We can define : } R_{\tilde{X}(s)=R_{\tilde{X}(t,t+s)}} \tag{80}$$

2.5.2 Important Random Processes

Independent identically-distributed sequences

- $\tilde{X}(i)$ has the same distribution $\forall i$.
- $\tilde{X}(i_1), \cdots, \tilde{X}(i_n)$ are mutually independent $\forall n, i_1, \cdots, i_n$.

It has constant mean and autocovariance 0 if $i \neq j$.

Gaussian Process If joint distribution of $\tilde{X}(t_1), \cdots, \tilde{X}(t_n)$, they are fully characterized by mean and autocovariance function.

Poisson Process

- Each event occurs independently from every other event.
- A given event has the same probability of occurring at any given point of the time interval.
- Events occur at a rate of λ events per time interval.

Definition : $\tilde{N}(t)$ number of events happening between 0 and t.

- $\tilde{N}(0) = 0$.
- $\tilde{N}(t_2) - \tilde{N}(t_1)$ is a Poisson random variable of paramter $(t_2 - t_1)$.
- $\tilde{N}(t_2) - \tilde{N}(t_1)$ and $\tilde{N}(t_4) - \tilde{N}(t_3)$ are independent if no overlap.

Mean λt, autocovariance $\min(t_1, t_2)$.

3 Foundation of Machine Learning

3.1 Optimization

Constrained optimisation problem

$$f, g_i : X \mapsto \mathbb{R} \tag{81}$$

$$\min_{x \in X} f(x) \tag{82}$$

$$\text{Subject to : } g_i(x) \leq 0, \forall i \in [1..n] \tag{83}$$

Lagrange Function

$$\forall x \in X, \forall \alpha \geq 0 L(x, \alpha) = f(x) + \sum_{i=1}^{n} \alpha_i g_i(x) \tag{84}$$

Theorem If P constrained optimization problem, $X = \mathbb{R}^n$, (x^*, α^*) saddle point and :

$$\forall x \in \mathbb{R}^N, \forall \alpha \geq 0, L(x^*, \alpha) \leq L(x^*, \alpha^*) \leq L(x, \alpha^*) \tag{85}$$

Then (x^*, α^*) is a solution of P.

Strong Slater's condition or Strong constraint qualification :

$$\text{if : } int(X) \neq \emptyset, \exists \bar{x} \in int(X) : g(\bar{x}) < 0 \tag{86}$$

Weak Slater's condition or Weak constraint qualification :

$$\text{if : } int(X) \neq \emptyset, \exists \bar{x} \in int(X) : \forall i \in [1..n], g_i(\bar{x} < 0) \text{ or } g_i(\bar{x} = 0) \text{ and } g_i \text{ affine.} \tag{87}$$

Theorem f and $g_i, i \in [1..n]$ convex functions and Slater's condition holds, x sol of the constrained optimisation problem, then $\exists \alpha \geq 0$ (x, α) is a saddle point of the Lagrangian.

Theorem f and $g_i, i \in [1..n]$ convex functions, differentiable and the weak Slater's condition holds. If x is a solution of the constrained optimisation problem, then $\exists \alpha \geq 0$, (x, α) saddle point of the Lagrangian.

Kuhn-Tucker's theorem f, g_i convex differentiable functions with qualified constraints. $\bar{x}$ solution of the constrained optimisation problem iff $\exists \bar{\alpha} \geq 0$:

$$\nabla_x L(\bar{x}, \bar{\alpha}) = \nabla_x f(\bar{x}) + \bar{\alpha} \cdot \nabla_x g(\bar{x}) = 0 \tag{88}$$

$$\nabla_\alpha L(\bar{x}, \bar{\alpha}) = g(\bar{x}) \leq 0 \tag{89}$$

$$\bar{\alpha} \cdot g(\bar{x}) = \sum_{i=1}^{n} \bar{\alpha}_i g_i(\bar{x}) = 0 \tag{90}$$

Remarks Constraints are qualified if g_i linear, or see wikipedia ... Since constraints are qualified, if $\bar{x}$ solution, $\exists \bar{\alpha}$ such as $(\bar{x}, \bar{\alpha})$ saddle point implies the three KKT conditions.

Primal and dual problem

$$\text{Primal optimisation problem : } \min_{x \in X} f(x) \tag{91}$$

$$\text{subject to : } g(x) \leq 0 \tag{92}$$

$$\text{Dual optimisation problem : } \max_{\alpha} \inf_{x \in X} L(x, \alpha) \tag{93}$$

$$\text{subject to : } \alpha \geq 0 \tag{94}$$

$$\tag{95}$$

3.2 Learning with finite hypothesis set

Errors

$$\text{Generalization error : } R(h) = \mathop{E}_{(x,y) \sim \mathfrak{D}}[L(h(x), y)] \tag{96}$$

$$\text{Empirical error : } \hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), g_i) \tag{97}$$

$$\text{Bayes error : } R^* = \inf_{h \, measurable} R(h) \tag{98}$$

$$\text{Noise : } \forall x \in X, noise(x) = min(P(1|x), P(0|x)) \tag{99}$$

$$E[noise(x)] = R^* \tag{100}$$

Bayes Error : In the deterministic case, $R^* = 0$. Given a stochastic case we have :

$$\forall x \in \mathcal{X}, h_{bayes}(x) = \underset{y \in \{0,1\}}{\text{argmax}} P[y|x] \tag{101}$$

The average error is then : $min(P[0|x], P[1|x]) = noise(x)$ and $E[noise(x)] = R^*$.

Estimation & approximation : Let's consider $h, h^* \in \mathcal{H}$, h hypothesis and h^* best hypothesis in $\mathcal{H}$.

$$\text{Estimation error : } R(h) - R(h^*) \tag{102}$$
$$\text{Approximation error : } R(h^*) - R^* \tag{103}$$

The Empirical Risk Minimization hypothesis is an unbiased estimator of h^*. The complexity given $\mathcal{H}_n$ can be regularization for instance :

$$h_S^{ERM} = \underset{h \in \mathcal{H}}{\text{argmin}} \hat{R}_S(h) + (\text{complexity}(\mathcal{H}_n, n)) \tag{104}$$

General algorithm families : Empirical risk minimisation : $h = \underset{h \in H}{argmin} \hat{R}(h)$

Structural risk minimisation : $H_1 \subset H_2 ... \subset H_n$, $h = \underset{h \in H_n}{argmin} \hat{R}(h) + penalty(H_n, n)$

Regularization-base algorithms...

Probability Approximately correct learning or PAC learning

$c : X \mapsto \{0, 1\}$ target concept.
C concept class, a set of target concept c.
$\mathfrak{D}$ target distribution, fixed probability distribution over X.
$\mathbb{H}$ hypothesis, set of all linear classifiers.
Learning algorithms receives S samples, selects h_s from $\mathbb{H}$ approximating c.

$$\text{True error : } R(h) = \underset{x \sim \mathfrak{D}}{P}[h(x) \neq c(x)] = \underset{x \sim \mathfrak{D}}{E}[\mathbb{1}_{h(x) \neq c(x)}] \tag{105}$$
$$\text{Empirical error : } \hat{R}_s(h) = \underset{x \sim \tilde{\mathfrak{D}}}{P}[h(x) \neq c(x)] \tag{106}$$

C is PAC-learnable if $\exists L$ learning algorithm :

$$\forall c \in C, \forall \varepsilon > 0, \forall \delta > 0, \forall \mathfrak{D} \underset{s \sim \mathfrak{D}^n}{P}[R(h_s) \leq \varepsilon] \geq 1 - \delta \tag{107}$$

Cost representation of c :

- cost for $x \in X$ $O(n)$
- cost for $c \in C$ $O(size(c))$

Extension in $O(poly(\frac{1}{\varepsilon}, \frac{1}{\delta}, n, size(c)))$.

3.3 Rademacher Complexity

Definitions :

$$\text{Arbitrary loss} : L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R} \tag{108}$$
$$\forall h \in \mathcal{H},\ h : \mathcal{X} \mapsto \mathcal{Y} \tag{109}$$
$$g \colon \mathcal{Z} = \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R} \tag{110}$$
$$(x, y) \to L(h(x), y) \tag{111}$$
$$\mathcal{G} : \text{Family of loss functions associated to } \mathcal{H} \tag{112}$$

Empricial Radamacher Complexity : The Radamacher complexity is a measure of how well the function class $\mathcal{G}$ correlates with random noise on S. It's the richness of the family.

$$S = (z_1, \cdots, z_n)$$
$$\hat{R}_S(\mathcal{G}) = \mathop{\mathbb{E}}_{\sigma \in \{-1,1\}^n} \left[\sup_{g \in \mathcal{G}} \frac{1}{m} \sum_{i=1}^{n} \sigma_i g(z_i) \right]$$

Radamacher Complexity :

$$R_m(\mathcal{G}) = \mathop{\mathbb{E}}_{S \sim \mathcal{D}^n} [\hat{R}_s(\mathcal{G})] \tag{113}$$

Various results : See course for more informations.

$$E[g(z)] \leq \frac{1}{m} \sum_{i=1}^{m} g(z_i) + 2R_m(\mathcal{G}) + \sqrt{\frac{log\frac{1}{\delta}}{2m}} \tag{114}$$
$$\leq \frac{1}{m} \sum_{i=1}^{m} g(z_i) + 2\hat{R}_S(\mathcal{G}) + 3\sqrt{\frac{log\frac{2}{\delta}}{2m}} \tag{115}$$
$$\hat{R}_S(\mathcal{G}) = \frac{1}{2} \hat{R}_{S_x}(\mathcal{H}) \tag{116}$$
$$R(h) \leq \hat{R}(h) + R_m(\mathcal{H}) + \sqrt{\frac{log\frac{1}{\delta}}{2m}} \tag{117}$$

Growth Function : Measure of Richness of $\mathcal{H}$.

$$\Pi_{\mathcal{H}} : \mathbb{N} \mapsto \mathbb{N}$$
$$m \to \max_{\{x_1, \cdots, x_m\} \in \mathcal{X}} |\{(h(x_1), \cdots, h(x_n)) : h \in \mathcal{H}\}|$$

VC-Dimension : The VC dimension of $\mathcal{H}$ is the size of the largest set that can be fully shattered by $\mathcal{H}$. It's a worst case scenario measure.

$$VCdim(\mathcal{H}) = max\{m, \Pi_{\mathcal{H}(m)} = 2^m\} \tag{118}$$

3.4 Maximum Entropy Models

3.4.1 Density Estimation

Maximum Likelihood Solution : Select distribution $p \in \mathcal{P}$ that maximizes likelihood :

$$p_{Ml} = \underset{p\in\mathcal{P}}{\text{argmax}} Pr[S|p] \tag{119}$$

$$= \underset{p\in\mathcal{P}}{\text{argmin}} D(\hat{p}_s||p) \tag{120}$$

Maximum a Posteriori : or MAP

Select distribution $p \in \mathcal{P}$ that is the most likely, given the observed sample S and assuming a prior distribution $Pr[p]$ over $\mathcal{P}$.

$$p_{MAP} = \underset{p\in\mathcal{P}}{\text{argmax}} Pr[p|S] \qquad = \underset{p\in\mathcal{P}}{\text{argmax}} Pr[S|p]Pr[p] \tag{121}$$

3.4.2 Density Estimation with features

Training data : sample S of size m drawn i.i.d. from set $\mathcal{X}$ according to some distribution $\mathcal{D}$

Features :

$$\Phi\colon \mathcal{X} \to \mathbb{R}^N$$
$$x \mapsto \Phi(x) = [\Phi_1(x), ..., \Phi_N(x)]$$

Maxent Formulation : With $\beta = 0$ standard Maxent and $\beta > 0$ regularized Maxent.

$$\underset{p\in\Delta}{\min} D(p||p_0) \tag{122}$$

$$\text{Subject to : } \left\| \underset{x\sim p}{E}[\Phi(x)] - \underset{x\sim p_0}{E}[\Phi(x)] \right\|_\infty \leq \beta \tag{123}$$

Convex optimisation problem :

$$\min_p \sum_{x\in\mathcal{X}} p(x) log p(x) \tag{124}$$

$$\text{Subject to : } p(x) \geq 0, \forall x \in \mathcal{X} \tag{125}$$

$$\sum_{x\in\mathcal{X}} p(x) = 1 \tag{126}$$

$$\left| \sum_{x\in\mathcal{X}} p(x)\Phi_j(x) - \frac{1}{m}\sum_{i=1}^{m} \Phi_j(x) \right| \leq \beta \forall j \in [1..N] \tag{127}$$

Gibbs Distributions :

$$p_w(x) = \frac{p_0(x)\exp(w\cdot\Phi(x))}{Z} \tag{128}$$

$$Z = \sum_x p_0(x)\exp(w.\Phi(x)) \tag{129}$$

3.5 Reinforcement Learning

Reinforcement learning is the study of planing and learning in a scenario where a learner actively interacts with the environment to achieve a certain goal. This active interaction justifies the terminology of agent used to refer to the learner. The achievement of the agent's goal is typically measured by the reward he receives from the environment and which he seeks to maximize.

The agent is faced with the dilemma between exploring unknown states and actions to gain more information about the environment and the rewards, and exploiting the information already collected to optimize his reward. This is known as the exploration versus exploitation trade- off inherent in reinforcement learning.

3.5.1 Markov decision process model

Definition 3.1 (MDPs)**.** A Markov decision process (MDP) is defined by :

- A set of states S.
- A start/initial state $s_0 \in S$.
- A set of actions A.
- A transition probability $P(s'|s,a)$, a distribution over destination states.
- A reward probability $P(r'|s,a)$.

There is a finite horizon when the numbers of epochs is finite. An MDP is finite when both S and A are finite. At time t you are at state s_t and take action a_t, you arrive at s_{t+1} and get the reward r_{t+1}.

Definition 3.2 (Policy)**.** A policy is a mapping $\pi : S \mapsto A$.

We can define non-stationnary policy, particularly in the finite horizon case. The agent's objective is to find a policy that maximizes his expected reward.

- Finite horizon ($T < \infty$) : $\sum_{\tau=0}^{T-t} r(s_{t+\tau}, \pi(s_{t+\tau}))$
- Infinite horizon ($T = \infty$) : $\sum_{\tau=0}^{T-t} \gamma^\tau r(s_{t+\tau}, \pi(s_{t+\tau}))$. γ discount future rewards.

Definition 3.3 (Policy Value)**.** The value $V_\pi(s)$ of a policy π at state $s \in S$ is defined as the expected reward returned when starting at s and following policy π.

$$\text{Finite horizon} : V_\pi(s) = E[\sum_{\tau=0}^{T-t} r(s_{t+\tau}, \pi(s_{t+\tau}))|s_t = s] \tag{130}$$

Theorem 3.1 (Bellman equation)**.** *The values $V_\pi(s)$ of policy π at states $s \in S$ for an infinite horizon MDP obey the following system of linear equations:*

$$\forall s \in S, V_\pi(s) = E[r(s,\pi(s))] + \gamma \sum_{s'} Pr[s'|s,\pi(s)]V_\pi(s') \tag{131}$$

Theorem 3.2. *For a finite MDP, Bellman's equation admits a unique solution given by :*

$$\boldsymbol{V_0 = (I - \gamma P)^{-1} R} \tag{132}$$

$$\textit{Bellman vector equation :}\ \boldsymbol{V = R + \gamma P V} \tag{133}$$

Definition 3.4 (optimal policy)**.** A policy π^* is optimal if it has maximal value for all states $s \in S$.

Definition 3.5 (optimal state-action value)**.** The optimal state-action value function Q is defined for all $(s,a) \in S \times A$ as the expected return for taking action $a \in A$ at state $s \in S$ and then following the optimal policy:

$$Q^*(s,a) = E[r(s,a)] + \gamma \sum_{s' \in S} Pr[s'|s,a]V^*(s') \tag{134}$$

$$\text{Therefore} : V^*(s) = \max_{a \in A} Q^*(s,a) \tag{135}$$

$$\text{And} : \forall s \in S, \pi^*(s) = \operatorname*{argmax}_{a \in A} Q^*(s,a) \tag{136}$$

Theorem 3.3 (Bellman equation 2)**.**

$$\forall s \in S, V^*(s) = \max_{a \in A} E[r(s,a)] + \gamma \sum_{s' \in S} Pr[s'|s,a]V^*(s') \tag{137}$$

Note that this is not a linear system of equations due to the max operator.

3.5.2 Planning algorithms

We assume we know the transition probabilities $P(s'|s,a)$ and expected rewards $E[r(s,a)]$.

Algorithm 1 Value Iteration

$\boldsymbol{V} \leftarrow \boldsymbol{V_0}$ *Arbitrary value*
while $\|\boldsymbol{V} - \boldsymbol{\Phi}(\boldsymbol{V})\| \geq \frac{1-\gamma}{\gamma}\boldsymbol{\varepsilon}$ **do**
    $\boldsymbol{V} \leftarrow \boldsymbol{\Phi}(\boldsymbol{V})$
return $\boldsymbol{\Phi}(\boldsymbol{V})$

Value Iteration algorithm : seek to determine the optimal policy. For a vector $\boldsymbol{V} \in \mathbb{R}^{|S|}$ and :

$$\forall s \in S, [\boldsymbol{\Phi}(\boldsymbol{V})](s) = \max_{a \in A}\left\{ E[r(s,a)] + \gamma \sum_{s' \in S} Pr[s'|s,a]V(s') \right\}$$

Maximizing actions $a \in A$ at each state defines the optimal action at each state, thus a policy. We can write matrix equations :

$$\boldsymbol{\Phi}(\boldsymbol{V}) = \max_{\pi}\{\boldsymbol{R_\pi} + \boldsymbol{\gamma P_\pi V}\} \tag{138}$$

$$(P_\pi)_{s,s'} = Pr[s'|s, \pi(s)] \tag{139}$$

$$(R_\pi)_s = E[r(s, \pi(s))] \tag{140}$$

Theorem 3.4. *For any initial value V_0, the sequence defined by $V_{n+1} = \Phi(V_n)$ converges to V. The proof use the fact that Φ is γ-Lipschitz.*

Policy Iteration

Theorem 3.5. *Let $(V_n), n \in N$ be the sequence of policy values computed by the algorithm, then, for any $n \in N$, the following inequalities hold:*

$$V_n \leq V_{n+1} V^* \tag{141}$$

Algorithm 2 Policy Iteration

$\pi \leftarrow \pi_0$ Arbitrary value
$\pi' \leftarrow NIL$
while $(\pi \neq \pi)'$ **do**
    $\boldsymbol{V} \leftarrow \boldsymbol{V_\pi}$ policy evaluation: solve $\boldsymbol{(I - \gamma P_\pi)V = R_\pi}$
    $\pi' \leftarrow \pi$
    $\pi \leftarrow \underset{\pi}{\text{argmax}} \boldsymbol{R_\pi + \gamma P_\pi V}$
return π

Linear Programming Linear programing is solving an optimization problem with a linear objective function and linear constraints.

$$\min_{\boldsymbol{V}} \sum_{s \in S} \alpha(s) V(s) \tag{142}$$

$$\text{Subject to :} \forall (s,a) \in S \times A, V(s) \geq E[r(s,a)] + \gamma \sum_{s' \in S} Pr[s'|s,a]V(s) \tag{143}$$

We can add a constraint for $\alpha(s)$ look like probability. Also there is $|S||A|$ rows and $|S|$ columns in this LP. The complexity of the solutions techniques are more favorable in rows than columns, so we have a formulation based on the dual formulation :

$$\max_{\boldsymbol{x}} \sum_{s\in S, a\in A} E[r(s,a)]x(s,a) \tag{144}$$

$$\text{Subject to : } \forall s \in S, \sum_{a\in A} x(s',a) = \alpha(s') + \gamma \sum_{s\in S, a\in A} Pr[s'|s,a]x(s,a) \tag{145}$$

$$\forall (s,a) \in S \times A, x(s,a) \geq 0 \tag{146}$$

3.5.3 Learning algorithms

This section considers the more general scenario where the environment model of an MDP, that is the transition and reward probabilities , is unknown.

Stochastic Approximations Stochastic approximation methods are iterative algorithms for solving optimization problems whose objective function is defined as the expectation of some random variable, or to find the fixed point of a function H that is accessible only through noisy observations.

Theorem 3.6 (Mean Estimation)**.** *Let X be a random variable taking values in* $[0,1]$ *and let* $x_0, \cdots, x_m$ *be i.i.d. values of X. Define the sequence* $(_m), m \in N$ *by*

$$\mu_{m+1} = (1-\alpha_m)_m + \alpha_m x_m \tag{147}$$

with $_0 = x_0, \alpha_m \in [0,1]$, $\sum_{m\geq 0} \alpha_m = +\infty$ *and* $\sum_{m\geq 0} \alpha_m^2 < +\infty$*.Then,*

$$\mu_m \underset{m\to\infty}{\to} E[X] \tag{148}$$

Stochastic optimization is the general problem of finding the solution to the equation

$$\boldsymbol{x = H(x)} \tag{149}$$

When :

- $\boldsymbol{H(x)}$ cannot be computed, not known or too prohibitive
- An i.i.d. sample of $\boldsymbol{H(x_i) + w_i}$ is available with $E[w] = 0$.

A way to solve this problem is to consider :

$$\boldsymbol{x_{t+1}} = (1-\alpha_t)\boldsymbol{x_t} + \alpha_t[H(\boldsymbol{x_t}) + \boldsymbol{w_t}] \tag{150}$$

$$= \boldsymbol{x_t} + \alpha_t[H(\boldsymbol{x_t}) + \boldsymbol{w_t} - \boldsymbol{x_t}] \tag{151}$$

$$\text{More generally : } \boldsymbol{x_{t+1}} = \boldsymbol{x_t} + \alpha_t D(\boldsymbol{x_t}, \boldsymbol{w_t}) \tag{152}$$

Theorem 3.7 (Supermartingale convergence)**.** *Let* $(X_t)_{t\in\mathbb{N}}$, $(Y_t)_{t\in\mathbb{N}}$ *and* $Z_t)_{t\in\mathbb{N}}$ *be sequence of non-negative RV with* $\sum_{t\in\mathbb{N}} Y_t < +\infty$*. Let* $\mathcal{F}_t$ *denote all the information for* $t' \leq t$ *:* $\mathcal{F}_t = \{(X_t)_{t\leq t'}, (Y_t)_{t\leq t'}, (Z_t)_{t\leq t'}\}$*. If* $E[X_{t+1}|\mathcal{F}_t] \leq X_t + Y_t - Z_t$ *the following holds :*

- X_t *Converges to a limit with probability one.*
- $\sum_{t\in\mathbb{N}} Z_t < \infty$

See the book for more theorems.

3.5.4 Stochastic Approximation Algorithms

TD(0) The algorithm is based on Bellman's linear equations giving the value of a policy :

$$V_\pi(s) = \underset{s'}{E}\Big[r(s,\pi(s)) + \gamma V_\pi(s')|s\Big] \tag{153}$$

However the distribution on which we take the expected value is not known, instead we do this :

- Sampling a new state s'
- Updating the policy value according to :

$$V(s) \leftarrow (1-\alpha)V(s) + \alpha[r(s,\pi(s)) + \gamma V(s')] \tag{154}$$
$$= V(s) + \alpha[r(s,\pi(s)) + \gamma V(s') - V(s)] \tag{155}$$
$$\text{Temporal difference of } V \text{ values : } = r(s,\pi(s)) + \gamma V(s') - V(s) \tag{156}$$

Algorithm 3 TD(0)

$\boldsymbol{V \leftarrow V_0}$ Initialization
for $t \leftarrow 0$ to T **do**
  $s \leftarrow SelectState()$
  for each step for epoch t **do**
    $r' \leftarrow Reward(s, \pi(s))$
    $s' \leftarrow NextState(\pi, s)$
    $V(s) \leftarrow (1-\alpha)V(s) + \alpha(r' + \gamma V(s'))$
    $s \leftarrow s'$
return V

Q-Learning This section presents an algorithm for estimating the optimal state-action value function Q^* in the case of an unknown model. We can derive the optimal policy from the state-action value function.

$$Q^*(s,a) = E[r(s,a)] + \gamma \sum_{s' \in S} Pr[s'|s,a]V^*(s') \tag{157}$$
$$= \underset{s'}{E}[r(s,a) + \gamma \max_{a \in A} Q^*(s,a)] \tag{158}$$

The distribution is unknown again, so the Q-learning algorithm consists of the following :

- Sample state s'
- Update State-action value function :

$$Q(s,a) \leftarrow \alpha Q(s,a) + (1-\alpha)[r(s,a) + \gamma \max_{a' \in A} Q(s',a')] \tag{159}$$

This algorithm can be viewed as a Stochastic Formulation of the Value Iteration algorithm presented in the previous section.

Algorithm 4 Q-Learning(π)

$\boldsymbol{Q \leftarrow Q_0}$ Initialization
for $t \leftarrow 0$ to T **do**
  $s \leftarrow SelectState()$
  for each step for epoch t **do**
    $a \leftarrow SelectAction(\pi(s), s)$ Policy π derived from Q, e.g. ε-greedy
    $r' \leftarrow Reward(s, \pi(s))$
    $s' \leftarrow NextState(\pi, s)$
    $Q(s,a) \leftarrow \alpha Q(s,a) + (1-\alpha)[r' + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
return Q

Theorem 3.8 (Q-learning convergence)**.** *Considering a Finite MDP.* $\forall (s,a) \in S \times A$, $\sum_{t \in \mathbb{N}} \alpha_t(s,a) = \infty$ *and* $\sum_{t \in \mathbb{N}} \alpha_t(s,a)^2 < \infty$ *with* $\alpha_t \in [0,1]$ *then the Q-learning algorithm converges.*

The choice of the policy π according to which an action a is selected is not specified by the algorithm and, as already indicated, the theorem guarantees the convergence of the algorithm for an arbitrary

policy so long as it ensures that every pair (s, a) is visited infinitely many times. In practice, several natural choices are considered for π. One possible choice is the policy determined by the state-action value at time t, Q_t . Thus, the action selected from state s is $\underset{a \in A}{\text{argmax}} Q_t(s, a)$. But this choice typically does not guarantee that all actions are taken or that all states are visited. Instead, a standard choice in reinforcement learning is the so-called ε-greedy policy, which consists of selecting with probability (1ε) the greedy action ($\underset{a \in A}{\text{argmax}} Q_t(s, a)$) and with probability ε a random action. Another choice the Boltzman-Exploration :

$$p_t(a|s, Q) = \frac{\exp \frac{Q(s,a)}{\tau_t}}{\sum_{a' \in A} \exp \frac{Q(s,a')}{\tau_t}} \tag{160}$$

Where τ_t is the temperature which converges to 0 as t increase to infinity. For instance $1/\log(n_t(s))$ where $n_t(s)$ is the number of time s has been visited up to time t.

Reinforcement learning algorithms include two components: a learning policy, which determines the action to take, and an update rule, which defines the new estimate of the optimal value function. For an off-policy algorithm, the update rule does not necessarily depend on the learning policy. Q-learning is an off-policy algorithm since its update rule is based on the max operator and the comparison of all possible actions a, thus it does not depend on the policy π. In contrast, the algorithm presented in the next section, SARSA, is an on-policy algorithm.

SARSA The algorithm is in fact very similar to Q-learning, except that its update rule is based on the action a' selected by the learning policy. Thus, SARSA is an on-policy algorithm, and its convergence therefore crucially depends on the learning policy.

Algorithm 5 SARSA

$\boldsymbol{Q \leftarrow Q_0}$ Initialization e.g. $Q_0 = 0$.
for $t \leftarrow 0$ to T **do**
  $s \leftarrow SelectState()$
  $a \leftarrow SelectAction(\pi(Q), s)$ Policy derived from Q, e.g. ε-greedy.
  for each step for epoch t **do**
    $r' \leftarrow Reward(s, \pi(s))$
    $s' \leftarrow NextState(\pi, s)$
    $a \leftarrow SelectAction(\pi(s), s)$ Policy π derived from Q, e.g. ε-greedy
    $Q(s, a) \leftarrow Q(s, a) + \alpha_t(s, a)[r' + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$
    $a \leftarrow a'$
return Q

TD(λ) algorithm Both TD(0) and Q-learning algorithms are only based on immediate rewards. The idea of TD(λ) consists instead of using multiple steps ahead. Thus, for $n > 1$ steps, we would have the update

$$V(s) \leftarrow V(s) + \alpha(R_t^n - V(s)) \tag{161}$$

$$R_t^n = \sum_{i=1}^{n} \gamma^{i-1} r_{t+i} + \gamma^n V(s_{t+n}) \tag{162}$$

How to choose n ? Instead, TD(λ) use of geometric distribution over R_t^n with $R_t^\lambda = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n R_t^n$. Thus the update rule :

$$V(s) \leftarrow V(s) + \alpha(R_t^n - V(s)) \tag{163}$$

Algorithm 6 TD(λ)())

$\boldsymbol{V} \leftarrow \boldsymbol{V_0}$ Initialization
$\boldsymbol{e} \leftarrow 0$
for $t \leftarrow 0$ to T **do**
    $s \leftarrow SelectState()$
    for each step for epoch t **do**
        $s' \leftarrow NextState(\pi, s)$
        $\delta \leftarrow r(s, \pi(s)) + \lambda V(s') - V(s)$
        $e(s) \leftarrow \lambda e(s) + 1$
        for $u \in S$ **do**
            if $u \neq s$ **then**
                $e(u) \leftarrow \gamma\lambda e(u)$
            $V(u) \leftarrow V(u) + \alpha\delta e(u)$
        $s \leftarrow s'$
return V

$\lambda = 0$ correspond to TD(0) and $\lambda = 1$ to the total future reward.

The scenario faced in many practical applications is more challenging; often, the information the agent receives about the environment is uncertain or unreliable. Such problems can be modeled as partially observable Markov decision processes (POMDPs). POMDPs are defined by augmenting the definition of MDPs with an observation probability distribution depending on the action taken, the state reached, and the observation. The presentation of their model and solution techniques are beyond the scope of this material.

3.5.5 Large State Space

In some cases in practice, the number of states or actions to consider for the environment may be very large. For example, the number of states in the game of backgammon is estimated to be over 10^{20}.

Suppose we wish to estimate the policy value $V_\pi(s)$ at each state s using experience obtained using policy π. To cope with the case of large state spaces, we can map each state of the environment to $\mathbb{R}^n$ via a mapping $\Phi : S \mapsto \mathbb{R}^N$, with N relatively small (200 for backgammon) and approximate $V_\pi(s)$ by a function $f_w(s)$ parameterized by some vector w. For example, f_w could be a linear function defined by $f_w(s) = w\Delta\Phi(s)$ for all $s \in S$, or some more complex non-linear function of w. The problem then consists of approximating V_π with f_w and can be formulated as a regression problem. Note, however, that the empirical data available is not i.i.d.

Suppose that at each time step t the agent receives the exact policy value $V_\pi(s_t)$. Then, if the family of functions f_w is differentiable, a gradient descent method applied to the empirical squared loss can be used to sequentially update the weight vector w via:

$$w_{t+1} = w_t - \alpha\nabla_{w_t}\frac{1}{2}[V_\pi(s_t) - f_{w_t}(s_t)]^2 = w_t + \alpha[V_\pi(s_t) - f_{w_t}(s_t)]\nabla_{w_t} f_{w_t}(s_t) \tag{164}$$

It is worth mentioning, however, that for large action spaces, there are simple cases where the methods used do not converge and instead cycle.

4 Computer Vision

5 Introduction to Data Science

Part III
Second Semester

6 Research

6.1 Generative Adversarial Networks

6.1.1 Tips for training

- All images should be in $[-1, 1]^{3 \times Size \times Size}$. Generator output tanh.
- Flips label when training generator.
- Sample from a gaussian distribution•
- When doing interpolations, do the interpolation via a great circle, rather than a straight line from point A to point B.
- Weights normalization :
  - Convolution should be approx. $\mathcal{N}(0, 0.02)$ with no bias. No bias for avoiding degeneration.
  - Batch Normalization no bias and weights following $\mathcal{N}(1, 0.02)$.
- **Virtual Batch Normalization** : it can help, it means that you normalize using a reference batch that you select. Downside is that you need two forward pass instead of one.
- **Feature Matching** : specifying a new objective for the generator that prevents it from overtraining on the current discriminator. Letting $f(x)$ denote activations on an intermediate layer of the discriminator, our new objective for the generator is defined as:

$$\| \underset{x \sim p_{data}(x)}{E} f(x) - \underset{z \sim p_z(z)}{E} f(G(z)) \|_2^2 \tag{165}$$

- **Mini batch Discrimination** : One of the main failure modes for GAN is for the generator to collapse to a parameter setting where it always emits the same point. When collapse to a single mode is imminent, the gradient of the discriminator may point in similar directions for many similar points. Because the discriminator processes each example independently, there is no coordination between its gradients, and thus no mechanism to tell the outputs of the generator to become more dissimilar to each other. The concept of minibatch discrimination is quite general: any discriminator model that looks at multiple examples in combination, rather than in isolation, could potentially help avoid col- lapse of the generator. Let $f(x_i) \in \mathbb{R}^A$ denote a vector of features for input x_i, produced by some intermediate layer in the discriminator. We then multiply the vector $f(x_i)$ by a tensor $T \in \mathbb{R}^{A \times B \times C}$, which results in a matrix $M_i \in \mathbb{R}^{B \times C}$. We then compute the L1-distance between the rows of the resulting matrix M_i across samples $i \in \{1, 2, \cdots, n\}$ and apply a negative exponential : $c_b(x_i, x_j)$. The output $o(xi)$ for this minibatch layer for a sample x_i is then defined as the sum of the $c_b(x_i, x_j)$'s to all other samples: $o(x_i) \in \mathbb{R}^B$. Next, we concatenate the output $o(x_i)$ of the minibatch layer with the intermediate features $f(x_i)$ that were its input, and we feed the result into the next layer of the discriminator.
- **Historical Averaging** : When applying this technique, we modify each player's cost to include a term $\|\boldsymbol{\theta} - \frac{\mathbf{1}}{t} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{t}} \boldsymbol{\theta}[\boldsymbol{i}]\|^2$. This approach is loosely inspired by the fictitious play algorithm that can find equilibria in other kinds of games.
- **One-sidef label smoothing** : We therefore smooth only the positive labels to α, leaving negative labels set to 0.

6.1.2 Divergences of Probabilistic distribution

Definitions : $\mathcal{X}$ a compact metric set (space of images $[0,1]^d$). Σ set of all Borel subsets of $\mathcal{X}$. $Prob(\mathcal{X})$ space of probability measures defined on $\mathcal{X}$. Defining the elementary distances and divergences between two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$:

- Total Variation distance :

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \tag{166}$$

- Kullback-Leibler divergence :

$$KL(\mathbb{P}_r || \mathbb{P}_g) = \int log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \tag{167}$$

  Where $\mathbb{P}_r$ and $\mathbb{P}_g$ assumed absolutely continuous (thus admits densities) with respect to μ. Recall : $\mathbb{P}_r$ admits a density $p_r(x)$ with respect tp μ ($\forall A \in \Sigma, \mathbb{P}_r(A) = \int_A P_r(x) d\mu(x)$) iff absolutely continuous with respect to $\mu : \forall A \in \Sigma, \mu(A) = 0 \implies \mathbb{P}_r(A) = 0$.

- Jensen-Shannon divergence :

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r || \mathbb{P}_m) + KL(\mathbb{P}_g || \mathbb{P}_m) \tag{168}$$

$$\mathbb{P}_m = (\mathbb{P}_r + \mathbb{P}_g)/2 \tag{169}$$

- The Earth-Mover distance or Wasserstein-1 :

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} [||x - y||] \tag{170}$$

  $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ set of all joint distributions $\gamma(x,y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. $\gamma(x,y)$ indicates how much mass must be transported from x to y in order to transform $\mathbb{P}_r$ into $\mathbb{P}_g$. "EM" is the cost of the optimal transport plan.

6.2 WGAN

Loss

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{171}$$

$$= \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \tag{172}$$

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z\sim p(z)}[\nabla_\theta f(g_\theta(z))] \tag{173}$$

Integral Probability Metrics (IPMs) Given $\mathcal{F}$ a set of functions from $\mathcal{X}$ to $\mathbb{R}$ we can define :

$$d_\mathcal{F}(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \tag{174}$$

Some notations :

$$C_b(\mathcal{X}) = \{f : \mathcal{X} \mapsto \mathbb{R}, f \text{ continuous and bounded}\} \tag{175}$$

$$C_b(\mathcal{X})* = \{\phi : C_b(\mathcal{X}) \mapsto \mathbb{R}, \phi \text{ linear and continuous}\} \tag{176}$$

As $(C_b(\mathcal{X}), \|\|_\infty)$ is a normed vector space, $C_b(\mathcal{X})*$ is its dual. With the dual norm $\|\phi\| = sup_{f \in C_b(\mathcal{X}), \|f\|_\infty \leq 1} |\phi(f)|$.

- $W(\mathbb{P}_r, \mathbb{P}_\theta) = d_\mathcal{F}(\mathbb{P}_r, \mathbb{P}_\theta)$ if $\mathcal{F}$ is the set of 1-Lipschitz functions.
- $\delta(\mathbb{P}_r, \mathbb{P}_\theta) = d_\mathcal{F}(\mathbb{P}_r, \mathbb{P}_\theta)$ if $\mathcal{F}$ is the set of all measurable functions bounded between -1 and 1.

- Energey-based GANs (EBGANs) can be thought as the generatibe approach to the total variation distance. The discriminator will be trying to optimize the equation (174) with measurables functions bounded between 0 and m. Which in essence as an optimization problem is the same as -1 and 1.
- Maximum Mean Discrepancy (MMD) special case of IPMs where $\mathcal{F} = \{f \in \mathcal{H} : \|f\|_\infty \leq 1\}$ with $\mathcal{H}$ some Reproducing Kernel Hilbert Space (RKHS) associated with a given kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. It's not only a pseudo-metric but a proper metric when the kernel is universal. If $\mathcal{H} = L^2(\mathcal{X}, m)$ for m the normalized Lebesque measure on $\mathcal{X}$, we know that $\{f \in C_b(\mathcal{X}), \|f\|_infty \leq 1\}$ will be contained in $\mathcal{F}$ and therefore $d_{\mathcal{F}}(\mathbb{P}_r, \mathbb{P}_\theta) \leq \delta(\mathbb{P}_r, \mathbb{P}_\theta)$ so the regularity of the MMD distance as a loss function is as bad as the one of the total variation. ie : with low bandwith kernels the distance might be close to a saturating regime similar as with total variation or the JS.
- Generative Moment Matching Networks (GMMNs) generative counterpart of MMD. Backproping through the kernelized formula you optimize $d_{MMD}(\mathbb{P}_r, \mathbb{P}_\theta)$

7 Machine Learning

7.1 Risk Decomposition

Action An action is what is produced by our system.

Input x, action a outcome y evaluation $l(a, y)$. Decision function $\mathcal{X} \mapsto \mathcal{A}$. Evaluation function $\mathcal{A} \times \mathcal{Y} \mapsto \mathbb{R}$.

Real life steps

- Define an action space
- Specify evaluation criterion

risk of decision function

$$R(f) = \mathbb{E}[l(f(x), y)] \tag{177}$$

Cannot be computed.

Bayes decision function or target function.

$$f^* = \underset{f}{\text{argmin}} R(f) \tag{178}$$

Mean square risk

$$l(a, y) = (a - y)^2 \tag{179}$$
$$R(f) = \mathbb{E}[(f(x) - y)^2] = \mathbb{E}[(f(x) - E[y|x])^2] + \mathbb{E}[(E[y|x] - y)^2] \tag{180}$$
$$f^* = E[y|x] \tag{181}$$

Multi-class classification

$$\mathcal{A} = \mathcal{Y} = \{0, .., k\} \tag{182}$$
$$l(a, y) = 1_{a \neq y} \tag{183}$$
$$R(f) = P(f(x) \neq y) \tag{184}$$
$$f^*(x) = \underset{k}{\text{argmax}} P(y = k|x) \tag{185}$$

Empirical risk function $((x_1, y_1), ..., (x_n, y_n))$ drawn i.i.d.

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i \in [1..n]} l(f(x_i), y_i) \tag{186}$$

$$\hat{R}_n(f) \underset{n \to \infty}{\mapsto} R(f) \tag{187}$$

Empirical risk minimizer

$$\hat{f} = \underset{f}{\text{argmin}} \hat{R}_n(f) \tag{188}$$

Constrained Empirical risk minimizer We can fit perfectly to the data but not at all overall. We need to smooth things up so hypothethis space.

$$\hat{f} = \underset{f \in \mathcal{F}}{\text{argmin}} \hat{R}_n(f) \tag{189}$$

Risk minimizer :

$$f^*_{\mathcal{F}} = \underset{f \in \mathcal{F}}{\text{argmin}} R(f) \tag{190}$$

7.2 Gradients

Unconstrained optimization problem

$$f : \mathcal{R}^d \to \mathcal{R} \tag{191}$$

$$x^* = \underset{x}{\text{argmin}} f(x) \tag{192}$$

The gradient is the direction of the steepest ascent.

Gradient descent Stoping criterion : when $\|\nabla f(x)\|_2 \leq \varepsilon$

Linear least square regression

$$\mathcal{X} = \mathbb{R}^d \tag{193}$$

$$\mathcal{Y} = \mathbb{R} \tag{194}$$

$$\mathcal{A} = \mathbb{R} \tag{195}$$

$$l(\hat{y}, y) = 1/2(\hat{y} - y)^2 \tag{196}$$

$$\mathcal{F} = \{f : \mathcal{R}^d \to \mathcal{R} | f(x) = w^T x, w \in \mathcal{R}^d\} \tag{197}$$

Minimize :

$$\hat{R}_n(w) = 1/n \sum l(f_w(x_i), y_i) \tag{198}$$

The gradient takes all data for one step. But if we take minibatches we have an unbiased estimator of the gradient, even with n=1. So let's do that.

Step size For SGD we want a decreasing step size to dampen noise in step direction. η_t step size at time t.

Robbins-Monro Conditions : Many classical convergence results depends on the following conditions :

$$\sum_{t\in\mathbb{N}} \eta_t^2 < \infty \tag{199}$$

$$\sum_{t\in\mathbb{N}} \eta_t = \infty \tag{200}$$

7.3 Directional derivatives and Optimality

Convex function A function is convex if its domain is convex and :

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \tag{201}$$

$$\theta \in [0,1] \tag{202}$$

If f is convex finite near x then all directional derivatives exists.

Descent direction v is a descent direction if $f'(x,v) < 0$. If $\nabla f(x) \neq 0$ then $-\nabla f(x)$ is a descent direction.

Theorem 7.1 (Characterization of mimum of convex function)**.** *If f convex finite near x*

- *x minimize f or*
- *There is a descent direction for f at x.*

$\lambda_{\max}$ for lasso ? Suppose there is $\lambda_{\max}$ such as :

$$J_\lambda(w) = \sum_{i\in[1..n]} (w^T x_i - y_i)^2 + \lambda |w|_1 \tag{203}$$

$$\lambda \geq \lambda_{\max} \implies \text{argmin} J_\lambda(w) = 0 \tag{204}$$

$$\text{argmin} J_\lambda(w) = 0 \implies J'_\lambda(0,v) \geq 0 \tag{205}$$

$\lambda_{\max}$ found following this condition.

7.4 Excess risk decomposition

$$f^* = \underset{f}{\text{argmin}} R(f) \tag{206}$$

$$f_{\mathcal{F}} = \underset{f\in\mathcal{F}}{\text{argmin}} R(f) \tag{207}$$

$$\hat{f}_n = \underset{f\in\mathcal{F}}{\text{argmin}} \hat{R}_n(f) = \underset{f\in\mathcal{F}}{\text{argmin}} \frac{1}{n}\sum_i l(f(x_i), y_i) \tag{208}$$

Approximation error : $R(f_{\mathcal{F}}) - R(f^*)$ Estimation error : $R(\hat{f}_n) - R(f_{\mathcal{F}})$

Excess risk

$$\text{Excess Risk}(f) = R(f) - R(f^*) \tag{209}$$

$$\text{Excess Risk}(\hat{f}_n) = R(\hat{f}_n) - R(f_{\mathcal{F}}) + R(f_{\mathcal{F}}) - R(f^*) \tag{210}$$

$$\text{Excess Risk}(\hat{f}_n) = \text{Estimation error} + \text{Approximation error} \tag{211}$$

Bigger the $\mathcal{F}$ smaller the approximation error. Smaller the $\mathcal{F}$ smaller the estimation error.

Optimization error In practice you can't find the ERM, but an approximation of it.

$$\text{Optimization error} = R(\tilde{f}_n) - R(\hat{f}_n) \tag{212}$$

It can be positive or negative.

$$\hat{R}(\tilde{f}_n) - \hat{R}(\hat{f}_n) \geq 0 \tag{213}$$

$$\text{Excess risk } (\tilde{f}_n) = R(\tilde{f}_n) - R(\hat{f}_n) + R(\hat{f}_n) - R(f_{\mathcal{F}}) + R(f_{\mathcal{F}}) - R(f^*) \tag{214}$$

$$\text{Excess risk } (\tilde{f}_n) = \text{Optimization error} + \text{Estimation error} + \text{Approximation error} \tag{215}$$

$$\tag{216}$$

7.5 L^1 and L^2 regularization

Tikhonov and Ivanov regularization We want a nested sequence of hypothesis space. Like all polynoms of degree lower than d. For linear models :

$$\Omega(f) \text{ complexity measure} \tag{217}$$

$$\mathcal{F}_r = \{f \in \mathcal{F} | \Omega(f) \leq r\} \tag{218}$$

Increasing complexity gives nested spaces.

Ivanov regularization or Constrained ERM

$$\min_{f \in \mathcal{F}} 1/n \sum_i l(f(x_i), y_i) \tag{219}$$

$$\text{subject to : } \Omega(f) \leq r \tag{220}$$

Cross-validate r.

Penalized ERM or Tikhonov regularization

$$\Omega(f) \geq 0 \tag{221}$$

$$\min_{f \in \mathcal{F}} 1/n \sum_i l(f(x_i), y_i) + \lambda \Omega(f) \tag{222}$$

Equivalence They are often equivalent. Which mean $\forall r, \exists \lambda$ that yields the same solution and conversly.

Example Linear least square regression can overfit if number of features d is big compared to n number of examples, for instance in text classification.

Ridgre regression

$$\text{Tikhonov regularization :} \tag{223}$$

$$\hat{w} = \underset{w \in \mathcal{R}^d}{\text{argmin}} \frac{1}{n} \sum_i (w^T x_i - y_i)^2 + \lambda \|w\|_2^2 \tag{224}$$

$$\text{Ivanov regularization :} \tag{225}$$

$$\hat{w} = \underset{w \in \mathcal{R}^d}{\text{argmin}} \frac{1}{n} \sum_i (w^T x_i - y_i)^2 \tag{226}$$

$$\text{Subject to : } \|w\|_2^2 \leq r^2 \tag{227}$$

Lipschitz If $\hat{w}$ solution of Ivanov ridge regression, then $\hat{f}$ is $\|w\|_2 \leq r$ lipschitz.

Lasso regression

$$\text{Tikhonov regularization :} \tag{228}$$

$$\hat{w} = \underset{w \in \mathcal{R}^d}{\text{argmin}} \frac{1}{n} \sum_i (w^T x_i - y_i)^2 + \lambda \|w\|_1 \tag{229}$$

$$\text{Ivanov regularization :} \tag{230}$$

$$\hat{w} = \underset{w \in \mathcal{R}^d}{\text{argmin}} \frac{1}{n} \sum_i (w^T x_i - y_i)^2 \tag{231}$$

$$\text{Subject to : } \|w\|_1 \leq r \tag{232}$$

Feature sparsity Lasso gives feature sparsity. Why is that good ?

- Time/expense to compute/buy features
- memory to store features
- Identifies important features
- Better prediction, sometimes
- Feature-selection step for training slower non-linear models.

Sparsity understanding For least square :

$$\text{Empirical risk : } \hat{R}_n(w) = 1/n \|Xw - y\|^2 \tag{233}$$

$$X \text{ design matrix} \tag{234}$$

$$\hat{R}_n(w) \text{ minimized by : } (X^T X)^{-1} X^T y \text{ OLS sol.} \tag{235}$$

$$\hat{R}_n(w) = \frac{1}{n}(w - \hat{w})^T X^T X (w - \hat{w}) + \hat{R}_n(\hat{w}) \tag{236}$$

$$\text{It's an elipsoïde : } \{w | \hat{R}_n(w) = \hat{R}_n(\hat{w}) + c\} = \{w | (w - \hat{w})^T X^T X (w - \hat{w}) = nc\} \tag{237}$$

If design matrix is orthogonal, then we have circles not elipsoïdes. If the OLS solution lies in a very big space starting from corners of L^1 ball then solution will be on the corner of L^1 ball. See class picture. For L^q norm $0 \leq q \leq 1$ we have even better space where solution will be projected on angles. But harder to optimize.

Solving Lasso Split L^1 in positive and negative parts :

$$\min_{w^+, w^-} 1/n \sum_i ((w^+ - w^-)^T x_i - y_i)^2 + \lambda 1^T (w^+ + w^-) \tag{238}$$

$$\text{Subject to : } w^+ \geq 0, W^- \geq 0 \tag{239}$$

Projected SGD on constrained problem Just like SGD, but after each step, you look at any composant w^+, w^-, if negative set it back to 0.

Coordinate descent solving Goal : minimize L(w). To do so at each of coordinate descent you adjust only one w $w_i^{new} = \text{argmin} L(w_1, w_{i-1}, w_i, ..., w_n)$. Coordinate descent is great when it's easy to minimize w.r.t. one coordinate at a time. If the coordinate is randomly selected : Stochastic coordinate descent. If it's cyclic : cyclic coordinate descent.

Closed form coordinate descent for Lasso

$$\hat{w}_j = \underset{w_j \in \mathcal{R}}{\operatorname{argmin}} \sum_i (w^T x_i - y_i) + \lambda \|w\|_1 \tag{240}$$

$$\hat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases} \tag{241}$$

$$a_j = 2 \sum_i x_{i,j}^2 \tag{242}$$

$$c_j = 2 \sum_i x_{i,j} (y_i - w^T x_i + w_j x_{i,j}) \tag{243}$$

Coordinate descent work if f $\mathcal{C}^1$ stricly convex in each of its coordinates. Lasso not $\mathcal{C}^1$. But there is a theorem that makes it work under conditions that lasso meets.

Remark : a single projected gradient step is sufficient for l_1 regularization.

7.6 Elastic-net

Equal features If we have two equal features, then Lasso will put random weight on both as long as they sum up to the best w_i for this feature. Ridge regression actually put half the best weight on each feature.

Linearly related features $x_2 = 2x_1$. For l^2 we have $w_2 = 2w_1$. For l^1 we have a sparse solution (end up in a corner) and choose the variable with larger scale.

Correlated features The more features are correlated the more the ellipsoid turns into line. Linearly dependent equals lines. So the l^1 solution is unstable as a minor perturbation of data points can change drastically the solution. When features are highly correlated we want to give same equal weight, to let their errors cancel out. Elastic-net does that.

Elastic-net The sparse region of elastic-net is smaller than l^1

7.7 Loss functions

Distance based regression often based on residuals $r = \hat{y} - y$. Square loss very affected by outliers.

Robustness how affected the learning algorithm is by outliers. Square loss not robust, Laplace or absolute loss robust but not differentiable. Huber loss, mix of the two.

Classification problem If $f : \mathcal{X} \to \{0, 1\}$ 0-1 loss is good. If $f : \mathcal{X} \to \mathcal{R}$ you can see the score as a confidence indicator. The margin is $y\hat{y}$.

Minimizing the 0-1 loss is not computationally feasible. Hinge loss is better (convex) not differentiable though.

7.8 Geometric Derivation of SVMs

Soft margin SVMs when points are not linearly separable. Versus hard margin SVMs. Hard margin not always the best when data is separable, because there can be almost no margin vs a good margin for soft margin SVMs.

7.9 Convex optimization

General Optimization Problem

$$\text{Minimizer : } f(x) \tag{244}$$

$$\text{Subject to : } f_i(x) \leq 0 \tag{245}$$

$$h_i(x) = 0 \tag{246}$$

Feasable set set of points satisfying the constraints. A feasible point is a point satisfying the constraints.

Optimal value $p^* = \inf(f_0(x)|x \text{ satifies constraints})$. $f(x^*) = p^*$ optimal point. An active constraint is when $f_i(x) = 0$. We can drop the h_i.

Lagrangian

$$L(x, \lambda) = f_0(x) + \sum_i \lambda_i f_i(x) \tag{247}$$

λ_i lagrangian multipliers or dual variables.

$$\sup_{\lambda > 0} L(x, \lambda) = \begin{cases} f_0(x) & \text{if } x \text{ constrained} \\ +\infty & \text{otherwise} \end{cases} \tag{248}$$

$$p^* = \inf_x \sup_\lambda L(x, \lambda) \tag{249}$$

Lagrangian dual problem

$$d^* = \sup_\lambda \inf_x L(x, \lambda) \tag{250}$$

$$p^* \geq d^* \tag{251}$$

It's called weak duality for any optimization problem (still no convexity here !). $p^* - d^*$ the duality gap. We often have strong duality for convex problems.

Lagrangian dual function

$$g(\lambda) = \inf_x L(x, \lambda) \tag{252}$$

$$p^* \geq g(\lambda) \tag{253}$$

It can have $-\infty$ values. But it's always concave (pointwise min of affine functions !). Why ?!?

Lagrange dual problem

$$\text{maximize } g(\lambda) \tag{254}$$

$$\text{subject to : } \lambda \geq 0 \tag{255}$$

λ is a dual feasible point if $g(\lambda) > -\infty$. λ^* dual optimal or optimal lagrangian multipliers if optimal for lagrange dual problem.

- Often easier to solve.
- d^* can be a stopping criterion.
- dual can reveal structure in solution.

Slater's constraint qualification for strong duality In a convex problem. Roughly the problem must be stricly feasible : $\exists x$ who stricly satisfies all constraints. For affine constraints $\leq$ is sufficient.

Complementary slackness If strong duality :

$$\lambda_i^* f_i(x^*) = 0 \tag{256}$$

Either constraint is active or lagrange multiplier is zero or both.

$$f_0(x^*) = g(\lambda^*) = \inf_x L(x, \lambda^*) \tag{257}$$
$$\leq L(x^*, \lambda^*) \tag{258}$$
$$= f_0(x^*) + \sum_i \lambda_i^* f_i(x^*) \tag{259}$$
$$\leq f_0(x^*) \tag{260}$$

Because $\lambda_i^* \geq 0$ and $f_i(x^*) \leq 0$. So by sandwich proof every $\lambda_i^* f_i(x^*)$ must be 0.

Consequences If we have strong duality :

$$p^* = d^* = f_0(x^*) = g(\lambda^*) = L(x^*, \lambda^*) \tag{261}$$
$$\lambda_i^* f_i(x^*) = 0 \tag{262}$$
$$L(x^*, \lambda^*) = \inf_x L(x, \lambda^*) \tag{263}$$
$$\nabla L(x^*, \lambda^*) = 0 \text{ First order condition} \tag{264}$$

KKT conditions : If we have strong duality, with all f_i differentiable but not necessarily convex :

- Primal and dual feasible $f_i(x^*) \leq 0$ and $\lambda_i^* \geq 0$.
- Complementary slackness.
- First order condition.

KKT for convex problems If f_i differentiable and convex, and $(\tilde{x}, \tilde{\lambda})$ satisfies KKT conditions then we have strong duality and $(\tilde{x}, \tilde{\lambda})$ are primal and dual optimal.

7.10 SVMs

We want to maximize the margin, margin measure of how correct we are. Hinge loss, convex upper bound on the 0-1 loss. The svm is the solution to :

$$\min_x 1/2\|w\|_2^2 + \frac{c}{n}\sum_i (1 - y_i(w^T x_i + b))_+ \tag{265}$$

It's an unconstrained optimization problem, not differentiable. Can we reformulate it ?

$$\text{Minimize : } 1/2\|w\|_2^2 + \frac{c}{n}\sum_i \xi_i \tag{266}$$
$$\text{Subject to : } -\xi_i \leq 0 \text{ and } (1 - y_i(w^T x_i + b)) - \xi_i \leq 0 \tag{267}$$

n + d + 1 unknowns and 2n affine constraints.

Dual optimization problem :

$$L(w, b, \xi, \alpha, \lambda) = 1/2\|w\|_2^2 + \frac{c}{n}\sum_i \xi_i + \sum_i \alpha_i((1 - y_i(w^T x_i + b)) - \xi_i) + \sum_i \lambda_i(-\xi_i) \tag{268}$$
$$= 1/2 w^T w + \sum_i \xi_i(\frac{c}{n} - \alpha_i - \lambda_i) + \sum_i \alpha_i(1 - y_i(w^T x_i + b)) \tag{269}$$

Do we have strong duality ? The problem is convex differentiable with affine constraints, strong duality i.i.f. there is a feasible point. $w = b = 0$ and $\xi_i = 1$ gives a feasible point. So strong duality. Dual optimization problem ?

$$g(\alpha,\lambda) = \inf_{w,b,\xi}\left[1/2w^T w + \sum_i \xi_i(\frac{c}{n} - \alpha_i - \lambda_i) + \sum_i \alpha_i(1 - y_i(w^T x_i + b))\right] \tag{270}$$

Optimal point if $\partial_w L = 0$, $\partial_b L = 0$ and $\partial_\xi L = 0$.
Note : $g = -\infty$ if $\frac{c}{n} - \alpha_i - \lambda_i \neq 0$.

$$\partial_w L = 0 \implies w = \sum_i \alpha_i y_i x_i \tag{271}$$

$$\partial_b L = 0 \implies \sum_i \alpha_i y_i = 0 \tag{272}$$

$$\partial_i L = 0 \implies \alpha_i + \lambda_i = \frac{c}{n} \tag{273}$$

So we have the dual optimization function :

$$g(\alpha,\lambda) = \begin{cases} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_j^T x_i & \text{If : } \sum_i \alpha_i y_i = 0 \text{ and : } \frac{c}{n} - \alpha_i - \lambda_i = 0 \\ -\infty & \text{Otherwise} \end{cases} \tag{274}$$

dual problem :

$$\sup_{\alpha,\lambda} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_j^T x_i \tag{275}$$

$$\text{Subject to : } \sum_i \alpha_i y_i = 0 \tag{276}$$

$$\forall i \quad : \quad \frac{c}{n} - \alpha_i - \lambda_i = 0 \tag{277}$$

$$\alpha_i \geq 0 \;\; \lambda_i \geq 0 \tag{278}$$

Or easier :

$$\sup_{\alpha,\lambda} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_j^T x_i \tag{279}$$

$$\text{Subject to : } \sum_i \alpha_i y_i = 0 \tag{280}$$

$$\alpha_i \in [0, \frac{c}{n}] \tag{281}$$

7.11 Sub-gradient descent

Theorem if $f : \mathcal{R}^d \mapsto \mathcal{R}$ is convex, sublevel sets are convex.

Theorem f convex i.i.f. :

$$f(x+v) \geq f(x) + \nabla f(x)^T v \tag{282}$$

Subgradients g is a subgradient of f at x if :

$$f(x+v) \geq f(x) + g^T v \tag{283}$$

$$\partial f(x) \text{ set of all subgradients of } f \text{ at } x \tag{284}$$

f subdifferentiable if $\partial f(x) \neq \emptyset$.

- If f convex differentiable $\partial f(x) = \{\nabla f(x)\}$.
- If f convex $\partial f(x) \neq \emptyset$.
- $\partial f(x)$ is a convex set.
- If 0 subgradient at x then minimum.
- If g subgradient of f at x, $(g, -1)$ orthogonal to the underestimating hyperplane.

Gradients lies normal to contours. $f : \mathcal{R}^d \mapsto \mathcal{R}$ with $\nabla f(x_0) \neq 0$ normal to $S = \{x \in \mathcal{R}^d | f(x) = f(x_0)\}$.

Subgradient descent : Same as gradient descent except at the end take $f_{best} = min f(x^i)$ because not a descent method. Given a step size that follows Robbins monroe conditions it converges. Fixed t with G lipschitz it gets near.

7.12 Kernels

7.12.1 Kernel methods

Feature extraction Mapping an input from $\mathcal{X}$ to $\mathbb{R}^d$ is called feature extraction. Feature map $\Phi : \mathcal{X} \mapsto \mathbb{R}^d$.

Kernelized A method is kernelized if inputs only appears inside products. $\rangle\Phi(x), \Phi(y)\langle$. Kernel function $k(x, y) = \rangle\Phi(x), \Phi(y)\langle$. Kernels are a bit similiarity score, escpecially RBF.

Gram matrix or kernel matrix, the matrix of inner product on the set. $K = XX^T$. We can change the kernel just by changing the matrix.

Some kernels

- Linear kernel $\Phi(x) = x$, $k(w, x) = w^T x$.
- Quadratic kernel $\Phi(x_1, x_2) = (x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$, $k(w, x) = w^T x + (w^T x)^2$.
- Polynomial kernel $k(w, x) = (1 + w^T x)^m$.
- Radial Basis Kernel / Gaussian kernel $k(w, x) = \exp(-\frac{\|w-x\|^2}{2\sigma^2})$.

7.12.2 Kernels second

If d features + bias, and do all monomials up to degree M : $M + dM$ terms total. Doesn't scale well.

Kernel ridge regression Recall ridge regression loss :

$$J(w) = \frac{1}{n}\sum_i (w^T x_i - y_i)^2 + \lambda \|w\|_2^2 \tag{285}$$

$$\phi : \mathbb{R}^d \mapsto \mathbb{R}^D \tag{286}$$

$$J(\tilde{w}) = \frac{1}{n}\sum_i (\tilde{w}^T \phi(x_i) - y_i)^2 + \lambda \|\tilde{w}\|_2^2 \tag{287}$$

$$J(\tilde{w}) = \frac{1}{n}\sum_i (k(w, x_i) - y_i)^2 + \lambda k(w, w) \tag{288}$$

That's false because in this case $\tilde{w}$ follows all the space but $k(w, x_i) = \phi(w)^T \phi(x_i)$.

Representer theorem - baby version Suppose we have loss function :

$$J(w) = L(w^T\phi(x_1), ..., w^T\phi(x_n)) + R(\|w\|_2) \tag{289}$$

- $w, \phi(x_i) \in \mathbb{R}^d$.
- L arbitrary function, loss term.
- R increasing, regularization term.

If J as a minimizer, then it has a minizer of the form $\sum_i \alpha_i \phi(x_i)$. If R stricly increasing then all have this form.

Representer theorem, ridgre regression $\tilde{w} = \sum_i \alpha_i \phi(x_i) = \Phi^T \alpha$ so :

$$J(\alpha) = \frac{1}{n}\|\Phi\Phi^T\alpha - y\|_2^2 + \lambda\alpha^T\Phi\Phi^T\alpha \tag{290}$$

Prediction function is then : $f_\alpha(x) = \sum_i \alpha_i k(x_i, x)$.

Generally we have :

$$J(w) = L(\Phi w) + R(\|w\|) \tag{291}$$
$$J(\alpha) = L(K\alpha) + R(\sqrt{\alpha^T K\alpha}) \tag{292}$$

example : SVM

$$J(w) = \frac{c}{n}\sum_i (1 - y_i(\phi(x_i)^T w))_+ + \|w\|_2^2 \tag{293}$$
$$J(\alpha) = \frac{c}{n}\sum_i (1 - y_i(K\alpha)_i)_+ + \alpha^T K\alpha \tag{294}$$

Inner product symmetric bi-linearity definite positive. A norm as an associated inner product if it follows the parallelogram law.

Hilbert space inner product vector space and complete respect to the norm.

Representer theorem - Adult version Suppose we have loss function :

$$J(w) = L(\rangle w, \phi(x_1)\langle, ..., \rangle w, \phi(x_n)\langle) + R(\|w\|) \tag{295}$$

- $w, \phi(x_i) \in \mathbb{H}^d$ hilbert.
- L arbitrary function, loss term.
- R increasing, regularization term.

Then representation ...

Mercer's theorem Fix kernel k. There is H and k if $k(x,y) = \rangle\phi(x), \phi(y)\langle$ i.i.f. $K = (k(x_i, x_j))$ positive semi-definite.

7.13 Features

We started with $\mathcal{X} = \mathbb{R}^d$, but it's not always the case. For instance NLP we have words.

7.13.1 Featurization

A Feature extraction method is an application $\mathcal{X} \mapsto \mathbb{R}^d$.

Feature template : A group of features all computed in a similar way. Like LengthString ≥ 10, last three characters ...

One-hot encoding : A set of features where exactly one has non-zero value.

7.13.2 Representation

Array representation for dense features $[1.0, 0.85, 0.2]$.

Map representation for sparse features $'cat' : 1$.

7.13.3 How to handle non-linearity with linear methods

Non-monotonicity If try to classify a circle boundary : Features = $[1, x, x^2]$.

Saturation Recommendation system based on number of time article bought. Is an article bought 1000 times 10 times better than an article bought 100 times ? Not really. Features $[log(1 + N(x))]$ for instance.

Saturation with discretization : $\Phi(x) = [1_{0 \leq N(x) 10}, 1_{10 \leq N(x) 100}]$. Buckets for flexible relationships.

Interaction between features Features are Height and Weight of a person. Predict health score.

Approach 1 : Google things up, $Score = (idealWeight(height(x)) - weight(x))^2$.

Approach 2 : $[1, h, w, h^2, w^2, hw]$: let your model figure it out by itself.

7.13.4 Predicate Features

A predicate on the input $\mathcal{X}$ is a function : $\mathcal{X} \mapsto \{True, False\}$.

7.14 Multiclass Classification

$\mathcal{X}, \mathcal{Y} = \{1, ..., k\}$.

7.14.1 One vs All

Train k classifiers h_i classifier distinguish i from others. At the end $h_1, ...h_k : \mathcal{X} \mapsto \mathbb{R}$. Final prediction : $h(x) = \underset{i \in \{1,...,k\}}{\text{argmax}} h_i(x)$. Ties can be arbitrarly broken.

7.15 Linear classifiers

$f(x) = \langle w, x \rangle$, prediction is the sign of $f(x)$. Example with 3 clouds of points. The best classifier misclassify all class 2, but it's the best one. Is it a problem of Hypothesis set not good ? No, we can find a way better classifier. How to get this good solution then ?

7.15.1 Multiclass Predictiors

Base hypothesis $\mathcal{H} = \{h : \mathcal{X} \mapsto \mathbb{R}\}$.

Multiclass hypothesis set : $\mathcal{F} = \{x \mapsto \underset{i \in \{1,...,k\}}{\text{argmax}} h_i(x) | h_1, ..., h_k \in \mathcal{H}\}$.

Reframing Base hypothesis $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}\}$.

Multiclass hypothesis set : $\mathcal{F} = \{x \mapsto \underset{y \in \mathcal{Y}}{\text{argmax}} h(x, y)\}$.

We then want : $h(x_i, y_i) > h(x_i, y) \forall y \neq y_i$. One loss function we could use :

$$\min_{h\in\mathcal{H}} \sum_{i=1}^{n} l[h(x_i, y_i) - \max_{y\neq y_i} h(x_i, y)] \tag{296}$$

7.15.2 Linear multi-class prediction function

$$h(x, y) = \langle w, \Psi(x, y)\rangle \tag{297}$$

Φ class sensitive feature map.

Example : 3 classes points in $\mathbb{R}^2$ The 3 classes before : $w_1 = (-\sqrt{2}/2, \sqrt{2}/2), w_2 = (0, 1), w_3 = (\sqrt{2}/2, \sqrt{2}/2), w = (w_1, w_2, w_3)$. And $\Psi(x, 1) = (x_1, x_2, 0, 0, 0, 0)$, $\Psi(x, 2) = (0, 0, x_1, x_2, 0, 0)$.

Example : NLP Predict NOUN, VERB, ADVERB ...

$\Psi_1(x, y) = 1_{x=APPLE and y=NOUN}$. A prediction is made by computing $\langle w, \Psi(apple, NOUN)\rangle$ for each class.

7.15.3 Label Features

Very large number of classes. For instance advertising banner click prediction. $\mathcal{X}$ User and user context. $\mathcal{Y}$ a large set of banners.

$$\Psi_1(x, y) = 1_{x \text{ is in target demographic of } y} \tag{298}$$
$$\Psi_2(x, y) = 1_{x \text{ previously clicked on add of company } y} \tag{299}$$

7.15.4 TF-IDF

$\mathcal{X}$ news articles. $\mathcal{Y}$ subject (politics, sports, ...).

Term Frequency : $TF(w, x)$ term frequency of word w in doc x. Number of times the word occurs.

Document Frequency : $DF(w, y)$ number of times w DO NOT appears in a topic y.

TF-IDF : $\Psi_w(x, y) = TF(w, x) \log(\frac{m}{DF(w,y)})$. m total number of doc in training set.

If you have d words, k classes you have d features in your vector.

7.15.5 Linear Multiclass SVM

$h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ class-sensitive score function.

Margin of h on i-th example (x_i, y_i) for class y :

$$m_{i,y}(h) = h(x_i, y_i) - h(x_i, y) \tag{300}$$

We want this margin to be positive large when $y \neq y_i$.

Linear case margin

$$m_{i,y}(h) = \langle w, \Psi(x_i, y_i)\rangle - \langle w, \Psi(x_i, y)\rangle \tag{301}$$

SVM

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max_{y\neq y_i}(1 - m_{i,y}(w))_+ \tag{302}$$

Class-sensitive loss It can happens that some mis-classifications are worse than others.

$$\Delta : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}^+ \tag{303}$$

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max_{y \neq y_i} (\Delta(y, y_i) - m_{i,y}(w))_+ \tag{304}$$

$\Delta(y, y_i)$ the target margin.

Geometric interpretation $\langle w, \Psi(x, y)\rangle$ is the projection of Ψ on w. We want :

$$\langle w, \Psi(x, y)\rangle - \langle w, \Psi(x, y')\rangle \geq \Delta(y, y') \tag{305}$$

7.15.6 Is this worth it compared to One-Vs-All ?

- One-vs-all : k classifiers, can fail drastically on simple examples.
- Multiclass-SVM : involve solving an argmax.

In practice, all methods are more or less the same. But Multiclass can generalize where one-vs-all is computationally intractable.

7.15.7 Structured Prediction

Part-of-Speech tagging (POS).

$$\mathcal{V} = \{\text{All words in english voc.}\} \cup \{[START], "."\} \tag{306}$$

$$\mathcal{P} = \{START, Pronoun, ...\} \tag{307}$$

$$\mathcal{X} = \mathcal{V}^n \tag{308}$$

$$\mathcal{Y} = \mathcal{P}^n \tag{309}$$

$\mathcal{Y}$ very large but with certain structure. Structure assumption : Markov chain $y_{n+1}|y_n, ... = y_{n+1}|y_n$.

Type 1 local features A single position of labels, x at any position :

$$\Phi_1(i, x, y_i) = 1_{x_i = APPLE} 1_{y_i = NOUN} \tag{310}$$

Type 2 local features Label at 2 consecutive position, x at any point.

$$\theta_1(i, x, y_{i-1}, y_i) = 1_{x_i = APPLE} 1_{y_i = NOUN} 1_{y_{i-1} = VERB} \tag{311}$$

Local Feature vector

$$\Psi_i(x, y_{i-1}, y_i) = (\Phi_1(i, x, y_1), ..., \theta_1(i, x, y_{i-1}, y_i), ...) \tag{312}$$

Local Compatibility score

$$\langle w, \Psi_i(x, y_{i-1}, y_i)\rangle \tag{313}$$

Sequence Compatibility score

$$\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i)\rangle = \langle w, \sum_i \Psi_i(x, y_{i-1}, y_i)\rangle \tag{314}$$

$$= \langle w, \Psi(x, y)\rangle \tag{315}$$

$\Psi(x, y) = \sum_i \Psi_i(x, y_{i-1}, y_i)$ sequence feature vector.

Sequence Target Loss How to compute a loss for predicting sequence y' ?

$$\text{Hamming Loss} \tag{316}$$

$$\Delta(y, y') = \frac{1}{|y|} \sum_{i=1}^{|y|} 1_{y_i \neq y'} \tag{317}$$

$$\text{Generalize : } \frac{1}{|y|} \sum_{i=1}^{|y|} \delta(y_i \neq y') \tag{318}$$

7.15.8 Left to do ?

Compute argmax can be quite complicated if big $\mathcal{Y}$, because of structure of Ψ with markov assumption :

- Dynamic Programming.
- Viterbi algorithm.

7.16 Trees

7.16.1 Introduction

Root node, internal node (split node), terminal (leaf) node.

Binary tree : Each node as 2 children or 0 children.

The decision function on $\mathbb{R}^2$ is rectangles.

We will consider :

- Binary trees, not multiway trees.
- Decision at each node on a single feature.
- splits only of the form $x - i \leq t$. Not oblique decision trees or Binary Space Partition trees (Linear split at each internal node) or Sphere trees.

7.16.2 Binary Regression Trees

Partition of $\mathcal{X} = \cup_i R_i$ with $R_i \cap R_j = \emptyset$. You have $f(x) = \sum_m c_m 1_{x \in R_m}$. Usually :

$$\text{If : } l(\hat{y}, y) = (\hat{y} - y)^2 \tag{319}$$

$$\hat{c}_m = E[y_i | x_i \in R_m] \tag{320}$$

With enough splits, every unique x as its own partition. It overfits a lot !

Control complexity of $\mathcal{H}$:

- Tree depth.
- Number of terminal nodes $|T| = M$ chosen metric.

For a given complexity we want to minimize square error on training set. Finding the optimal tree can be computationally intractable.

7.16.3 Greedy Algorithm

When we build a node, don't plan ahead. If continuous features $x = (x_1, ..., x_d) \in \mathbb{R}^d$ split choose $j \in \{1..d\}$ and $s \in \mathbb{R}$.

$$R_1(j,s) = \{x | x_j \leq s\} \tag{321}$$
$$R_2(j,s) = \{x | x_j > s\} \tag{322}$$
$$\hat{c}_1(j,s) = E[y_i | x_i \in R_1(j,s)] \tag{323}$$
$$\hat{c}_2(j,s) = E[y_i | x_i \in R_2(j,s)] \tag{324}$$
$$\text{Minimize : } \sum_{i, x_i \in R_1} (y_i - \hat{c}_1(j,s))^2 + \sum_{i, x_i \in R_2} (y_i - \hat{c}_2(j,s))^2 \tag{325}$$

Finding the split point : On j-th feature, as we change split points, change at most n-1. If $x_{j(1)}, ..x_{j(n)}$ are the sorted values of j-th feature, we only need to check spaces in-between. So only perform n-1 splits (why not n+1). Usually the middle points between sorted values.

Algorithm is good now : Look at all possible splits you can add to a tree and take the one which improves the best performances (reduce the most the loss).

7.16.4 Complexity control strategy : CART

If tree to big, over fit. Too small, miss patterns.

CART :

- Build a very big tree, stop when less than x points per region, or region no more mistakes.
- PRUNE tree.

Pruning : Consider an internal node n. To prune the sub-tree rooted at n, eliminate all descendants of n, n terminal node.

$\hat{R}(T)$ empirical risk, $T \subset T_0 \hat{R}(T) \geq \hat{R}(T_0)$.

Cost complexity criterion

$$C_\alpha(T) = \hat{R}(T) + \alpha |T| \tag{326}$$

When pruning following this strategy : Cost complexity or weakest link pruning. You need to cross-validate α.

Do this efficiently Do we need to search over all subtrees ? 2^n trees... No : Greedy Prunning Algorithm.

Find $T_1 \subset T_0$ minimizing $\hat{R}(T_1) - \hat{R}(T_0)$ with T_1 one fewer node than T_0.

At the end : $\mathcal{T} = \{T_0, ..., T_{|N|-1}\}$.

Result :

$$\{\underset{T \subset T_0}{\text{argmin}} C_\alpha(T) | \alpha \geq 0\} \subset \mathcal{T} \tag{327}$$

7.16.5 Classification Trees

$\mathcal{Y} = \{1, ..., k\}$. Modify criterion for splitting node. Method for prunning trees.

$$\hat{p}_{nk} = \frac{1}{N_n} \sum_{i, x_i \in R_n} 1_{y_i = k} \tag{328}$$
$$N_n = |R_n| \tag{329}$$

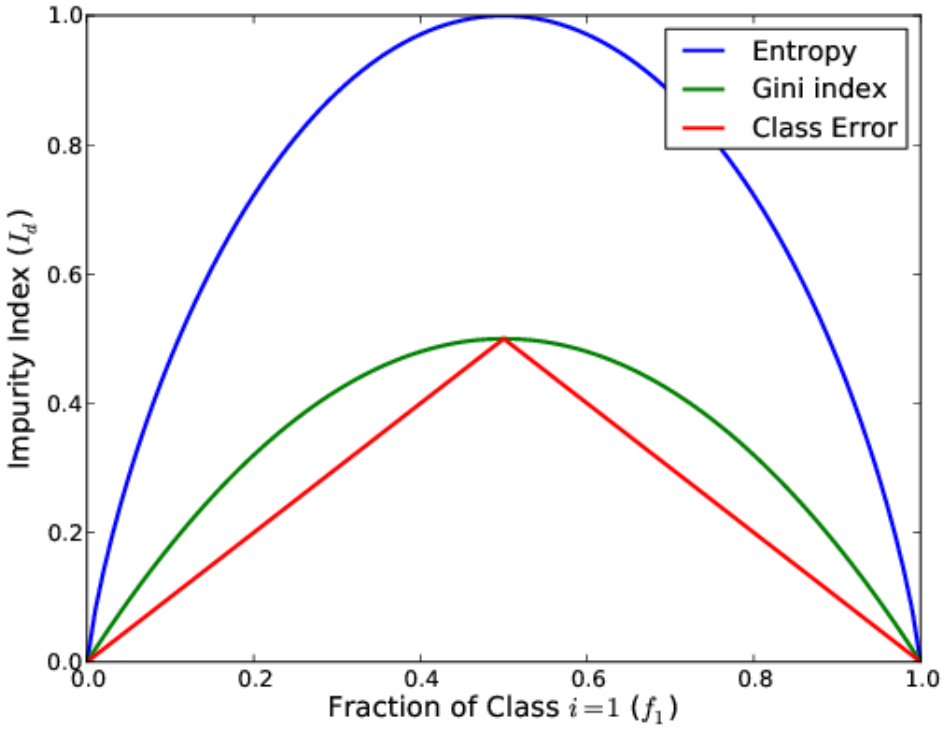
p_{nk} probability of being in k when in region n. We can have predicted class probabilioty and predicted classification node :

$$k(m) = \underset{k}{\text{argmax}} \hat{p}_{mk} \tag{330}$$

Miss-classification rate : $1 - \hat{p}_{nk(m)}$.

We want pure leaf nodes. Different measures of node impurity.

Figure 7: Impurity functions



- Misclassification error :

$$1 - \hat{p}_{m,k(m)} \tag{331}$$

- Gini Index :

$$\sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) \tag{332}$$

- Entropy :

$$-\sum_{k=1}^{K} \hat{p}_{m,k} \log(\hat{p}_{m,k}) \tag{333}$$

  Minimizing entropy = Maximazing information gain.

Let's assume we have a split with two regions R_l, R_r and N_l, N_r points in each of them.

$$\text{Minimize} : N_L Q(R_L) + N_R Q(R_R) \tag{334}$$

Gini and entropy are more effective for building trees, it leads to purer nodes, not just misclassification rate.

Example : Split 1 : $(3,1),)(1,3)$ Split 2 : $(2,4),(2,0)$ Gini/Entropy choose split 2 purer split.

7.16.6 Trees in General

Missing Features or predictors or covariates : The usual way (dummy variable, removing them ...).

Trees surrogate splits For every internal nodes, form a list of surrogate features and split points. Approximate original split as well as possible, Measure how well they approximate the original split.

Trees are easy to interpret

Finding Non-linear features If you fit a tree, you have regions. You can create features out of those regions and fit a linear classifier on it $1_{x \in R}$. Rule-fit by Friedman.

7.17 Boostraps

7.17.1 Why boostrap

P probability distribution. Expectation, variance, ... is a parameter of P, which is not a random variable.

If you have a sample $\mathcal{D}_n = (x_1, ..., x_n)$ i.i.d. samples of D, you can compute Statistics of the data (mean, variance on the data). Statistics are random variables.

The distribution of statistics is called a sampling distribution. Parameters of the sampling distribution : Not a random variable, can be useful like standard error (stdev of sampling distribution) gives confidence interval.

Bias

$$E[\hat{\mu}] - \mu \tag{335}$$

Estimator unbiased if above is 0.

Variance

$$E[\hat{\mu}^2] - E[\hat{\mu}]^2 \tag{336}$$

Estimating the variance of an estimator B independent samples of size n : $\mathcal{D}_n^1, ..., \mathcal{D}_n^B$.

$$E[\hat{\mu}] \approx \frac{1}{B} \sum_{i=1}^{B} \hat{\mu}(\mathcal{D}_n^i) \tag{337}$$

$$E[\hat{\mu}^2] \approx \frac{1}{B} \sum_{i=1}^{B} \hat{\mu}(\mathcal{D}_n^i)^2 \tag{338}$$

$$var(\hat{\mu}) = E[\hat{\mu}^2] - E[\hat{\mu}]^2 \tag{339}$$

Why variance ? Confidence interval $\hat{\mu} + / - \sqrt{\hat{var}(\hat{\mu})}$. Practically we have only on sample, we don't want to divide it into B groups.

7.17.2 Boostrap

Boostrap a boostrap sample of $\mathcal{D}_n$ is a sample of size m drawn with replacement from $\mathcal{D}_n$.

X_i has $(1 - \frac{1}{n})^n \approx e^{-1} = 0.368$ probability of being in the sample.

Boostrap Method when you simulate having B independent samples from P using B boostrap samples from the sample $\mathcal{D}_n$. For instance you can compute $\mu(\hat{\mathcal{D}}_n) + / - \sqrt{\text{Bootstrap Variance}}$.

7.18 Bagging and Random Forest

Averaging : Z and n i.i.d. samples $Z_1, ..., Z_n$. $E[Z] = \mu$ and $Var(Z) = \sigma^2$. Every single Z_i is an unbiased estimator of μ with standard deviation σ but $\bar{Z} = \frac{1}{n}\sum_i Z_i$ is also unbiased with variance $\frac{\sigma^2}{n}$.

If you have B independent prediction functions : $\hat{f}_1(x), ..., \hat{f}_B(x)$ and $\hat{f}_{AVG}(x) = \frac{1}{B}\sum_i \hat{f}_i(x)$ then $\hat{f}_{AVG}$ has same expected value as each $\hat{f}_i$ but lower variance. In real world you don't have B independent training set but Bootstrapping.

7.18.1 Bagging

B bootstrap samples $D^1, ..., D^B$ from D with $\hat{f}_1(x), ..., \hat{f}_B(x)$ decision functions and $\hat{f}_{AVG}(x) = \text{Combine}(\hat{f}_i(x))$

7.18.2 Bagging from regression

$\hat{f}_{AVG}(x) = \frac{1}{B}\sum_i \hat{f}_i(x)$. Empirically, $\hat{f}_{AVG}$ performs similarly to training on B independent samples with small variance.

7.18.3 Out-of-bag error estimation

Each bagged prediction is trained on approx. 63% of data points, so 37% remains. Out-of-bag observations (OOB) :

$$S_i = \{b | D^b \text{ does not contain i-th point}\} \tag{340}$$

OOB prediction on x_i is :

$$\hat{f}_{OOB}(x_i) = \frac{1}{|S_i|}\sum_{b \in S_i} \hat{f}_b(x_i) \tag{341}$$

It's a good estimate of the test error. OOB similar to cross validation error - both computed on training set.

7.18.4 Bagging for classification

2 ways to combine classifications :

- Consensus class : Majority vote.
- Average probabilities

7.18.5 Bias and Variance in Casual Usage

Restricting $\mathcal{F}$ biases the space :

- Toward a simpler model.
- Away from the best possible fit on the training data.

Full, un-prunned decision trees have little bias.

Prunning introduces bias

Variance is approx. how much the fit changes accross different training sets.

If different training sets yields similar fit, Algorithm has high stability.

Decision trees have high variance - not stable.

Bagging reduces variance without making bias worse.

Bagging helps most with :

- Unbiased base prediction functions.
- High variance low stability learning algorithms.

7.18.6 Random Forest

Motivations : Average $\hat{f}_1, ...\hat{f}_B$ reduces variances the most when i.i.d. samples :

Suppose $\forall i \neq j, cor(Z_i, Z_j) = \rho$.

$$Var(\bar{Z}) = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2 \tag{342}$$

Bootstrap samples are :

- Independent from training set.
- Not independent samples from $P_{\mathcal{X}\times\mathcal{Y}}$.

Limits amount of variances we can diminish.

Random Forest Use bagged decision trees but modify tree-growing procedure to reduce correlation between trees. Key step is to restrict each tree choice of splitting variable to a randomly chosen subset of features $m \approx \sqrt{p}$, p number of features.

7.19 Boosting

Ensemble methods combine multiple models :

- Parallel ensembles : Each model is built independently (Baggingm Random Forest).
- Sequential Ensembles : Models generated sequentially, add new models that do well where previous models lack.

7.19.1 Adaboost

Weighted train-set and classification error, minimize training error. Resistant to overfitting empirically. Minimize exponential loss function.

Weak learner A classifier which does slightly better than random. Can we combine weak learners to create a strong learner (accurate predictions) ?

$\mathcal{Y} = \{-1, 1\}$ and $\mathcal{H} = \{h : \mathcal{X} \to \{-1, 1\}\}$ Base hypothesis space. Base learner output class labels.

Typically $\mathcal{H}$ is :

- Decision stumps.
- Trees with few terminal nodes.
- Linear decision functions

Training set : $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ and weights $w_1, ..., w_n$.

Weighted empirical risk :

$$\hat{R}_n^w(f) = \frac{1}{W}\sum_i w_i l(y_i, f(x_i)) \tag{343}$$

$$W = \sum_i w_i \tag{344}$$

We can try to minimize weighted empirical risk. If our model can't be trained on weighted data, can sample a new data-set D with probability $w_1/W, ..., w_n/W$.

Algorithm 7 Adaboost skeleton

$\boldsymbol{D} \leftarrow \{(x_1, y_1), ..., (x_n, y_n)\}$ *Arbitrary value*
$w_i \leftarrow 1$
while $m = 1, ..., M$ **do**
    Find Base classifier $G_m(x)$ tries to fit weighted data.
    Increase weight on points. $G_m(x)$ misclassified.
return $G(x) = sign[\sum_m \alpha_m G_m(x)]$

With $\alpha_m \geq 0$ and $G_m(x) \in \{-1, 1\}$.

$$err_m = \frac{1}{W} \sum_i w_i 1_{y_i \neq G_m(x_i)} \in [0, 1] \tag{345}$$

$$\alpha_m = \log(\frac{1 - err_m}{err_m}) \in \mathbb{R}^+ \tag{346}$$

If G_m classifies x_i correctly, w_i changes, otherwise :

$$w_i \leftarrow w_i \exp(\alpha_m) \tag{347}$$

$$\leftarrow w_i \frac{1 - err_m}{err_m} \tag{348}$$

The smaller the err_m the more we increase weights on the misclassified.

Algorithm 8 Adaboost

$\boldsymbol{D} \leftarrow \{(x_1, y_1), ..., (x_n, y_n)\}$ *Arbitrary value*
$w_i \leftarrow 1$
while $m = 1, ..., M$ **do**
    Find Base classifier $G_m(x)$ tries to fit weighted data.
    $err_m = \frac{1}{W} \sum_i w_i 1_{y_i \neq G_m(x_i)}$
    $\alpha_m = \log(\frac{1 - err_m}{err_m})$
    $w_i \leftarrow w_i \exp(\alpha_m 1_{y_i \neq G_m(x_i)})$
return $G(x) = sign[\sum_m \alpha_m G_m(x)]$

7.19.2 Does Adaboost minimize training error ?

We've seen regularized risk minimization and Trees. Adaboost minimize training error if weak classifier have an edge over random. $\gamma_m = \frac{1}{2} - err_m$ the edge of the classifier. How much better than random G_m performs.

Theorem :

$$\frac{1}{n} \sum_i 1_{y_i \neq G(x_i)} \leq \prod_m \sqrt{1 - 4\gamma_m^2} \tag{349}$$

7.19.3 Boosting fits an additive model

Base hypothesis $\mathcal{H}$, an adaptive function expansion over $\mathcal{H}$ is $f(x) = \sum_m \nu_m h_m(x)$ with $\nu_m \in \mathbb{R}$ expansion coefficient and adaptative function $h_m \in \mathcal{H}$. We want to find $f(x)$ which minimize empirical risk, step by step starting with $f_0 = 0$. We need then step direction (h_m) and step size.

Algorithm 9 Forward Stagewise Additive Modeling

$f_0 = 0$
while $m = 1, ..., M$ **do**
    $(\nu_m, h_m) = \underset{\nu \in \mathbb{R}, h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} l(y_i, f_{m-1}(x_i) + \nu h(x_i))$
    $f_m = f_{m-1} + \nu_m h_m$
return $f_M(x)$

7.19.4 Forward Stagewise Additive Modeling (FSAM)

If $l(y, f(x)) = \exp(-yf(x))$ $h : \mathcal{X} \to \{-1, 1\}$ then we have adaboost. The only difference is that adaboost gets base learner slighly better than average is sufficient whereas here we explicitly ask for the best hypothesis.

7.19.5 Adaboost and robustness

Because of exponential loss, adaboost puts a lot of weight on outliers. If Bayes error is $.25$ for instance, the best we can do is predict the majority class for each x and some training examples should be wrong. Adaboost puts a lot of weights on those.

Degraded performances on :

- High label noise.
- High bayes error rate.

Logistic loss ($l(x) = \log(1 + \exp(-x))$) instead of exponential one is better in settings with high bayes error.

7.19.6 Population minimizer

The population refers to the full population of a group rather than a sample. In ML, population case is the hypothetical case of infinite training data. A population minimizer for a loss function is the risk minimizer in ML. For exponential loss, the population minimizer is :

$$f(x) = \frac{1}{2} \log \frac{P(Y = 1|X = x)}{P(Y = -1|X = x)} \tag{350}$$

By solving $P(Y = 1|X = x)$ we can give probabilistic prediction from adaboost as well.

Population minimizer of SVM higne loss is :

$$f^*(x) = sign[P(Y = 1|X = x) - \frac{1}{2}] \tag{351}$$

7.20 Gradient Boosting

Introduction We have 10 meteorologist which gives predictions from dataset. How to create the best predictor ? Stack $\hat{x}_i = (x_i, f_1(x_i), ..., f_{10}(x_i))$.

For FSAM we need to find the argmin at each step. As we have finite training data, each function h can be represented as its vector $(h(x_1), ..., h(x_n)$ to find the best one.

Note : The step $h_m = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i l(r_{m,i}, h(x_i))$ is a standard fitting like least square regression on residuals. It's called Anyboost or Functional Gradient Descent also.

We can call $\mathcal{F}_m = \{\sum_{i=1}^{m} \nu_i h_i | \nu \in \mathbb{R}, h \in \mathcal{H}\}$.

Algorithm 10 Gradient Boosting Machine

$f_0 = 0$
while $m = 1, ..., M$ **do**
  $r_m = -(\partial_2 l(y_1, f_{m-1}(x_1)), ..., \partial_2 l(y_n, f_{m-1}(x_n)))$
  $h_m = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i l(r_{m,i}, h(x_i))$
  $\nu_m = \underset{\nu \geq 0}{\operatorname{argmin}} \sum_i l(y_i, f_{m-1}(x_i) + \nu h_m(x_i))$
  $f_m = f_{m-1} + \nu_m h_m$
return $f_M(x)$

7.20.1 Options for step size

- Option 1 : Line search
- Option 2 : Shrinkage : We consider $\nu = 1$ to be the full gradient, so we can choose a fixed $\nu \in]0, 1]$, .1 in general (can be tuned).

7.20.2 Gradient Tree boosting

$\mathcal{H} = \{$Regression trees of size J$\}$. $J = 2$ is boosting stumps. $4 \leq J \leq 8$ usual values.

The smaller the step size the longuer you need, but gives often better results. So put the step size as small as your patience allows.

7.20.3 Stochastic Gradient Boosting

Choose Randomly chosen subset of data to compute boosting step. Usually 50%. This fraction is called the bag fraction. Subsample is an additional regularization parameter and it's much faster. This is a lot like mini-batch method, estimation of the true step with smaller batch.

Why 50 % ? Because in bagging, 50 % without replacement gives very similar results to full boostrap samples. Subsampling is inspired by bagging, it makes sense.

If we think about it in term of mini batch methods, makes little sense to choose batch size as a fixed size of the data set, especially for large data sets.

Bag as minibatch : Minibatch size should depend on complexity of base hypothesis space and complexity of target function.

Column feature subsampling : Similiar to random forest, choose a subset of feature for each round. Seems to yield better resutls 20% to 100%.

7.20.4 Newton step direction

GBM is a first order method, we approximate the gradient by fitting a function to it. Newton method is a second order method, required computing the hessian and gradient of J. Newton points toward minimizer of the quadratic, which is easy to find in closed form. Boosting methods with projected Newton step : LogitBoost, XGBoost.

7.20.5 XGBoost

Add explicit penalty term on tree complexity.

$$\Gamma(h) = \gamma T + \lambda 1/2 \sum_i w_i^2 \tag{352}$$

h regression tree, T number of leaf nodes, w_i is the prediction is the j-th node. And $J(h) = \Gamma(h) + ...$ second order approximation to empirical risk.

7.21 Probability models

$\mathcal{A} = \{\text{Probability distribution on space } \mathcal{Y}\}$. If we know $P(y|x)$ we can find $\hat{y}$ that minimize any other loss functions :

- least square regression : mean of $p(y|x)$.
- l_1 norm : median of $p(y|x)$.

7.21.1 Probability distribution of CitySense

Predict number of pick-ups is a Poisson 40. Is 100 a anomaly ?

$$p(y \geq 100|x) = \sum_{i \geq 100} P(y = i|x) \tag{353}$$

It's straitforward to give prediction intervals :

$$p(y \in [a, b]|x) = .95 \tag{354}$$

We need to do that the 2.5% and 97.5% quantiles.

7.21.2 Grid cells

Raw input is space (lat/long) and time of pickups. We do a spatial grid and consider only time as hours. Aggreate taxi pickups as Number of pickups by grid cells by hour.

We have $\mathcal{A} = \{\text{Probability distribution on space } \mathcal{Y}\}$ and $\mathcal{Y} = \{0, 1, ...\}$ actual number of taxi pickups.

7.21.3 Stratifying versus Bucketting

Stratifying If we partition our input space into groups and treat each groups separately. We would build a model for each group for instance, without information sharing between groups.

Bucketing or Binning : If we are combining natural groups in the data into a single group rather than building separate models for each groups. Combining each weakday together would be bucketing.

With a separate model for every grid cell and week-hour pair, model is highly specific and we could capture idiosyncrasy that we wouldn't catch otherwise. That is minimizing the bias. With relatively little data per stratum we have a big variance.

Bucketing will allow us to reduce variance at the cost of bias. Bucketing smartly we could minimize the bias increase.

You can trade-off bias variance by varying stratification and bucketing.

7.22 Maximum Likelihood Estimation

$p(y)$ probability distribution over $\mathcal{Y}$ unknown and want to estimate it. Here $\mathcal{Y}$ can be real numbers, 2 labels, k labels or infinite labels.

7.22.1 Evaluation

We have $\hat{p}(y)$ and $\mathcal{D} = \{y_1, ..., y_n\}$, likelihood :

$$\hat{p}(\mathcal{D}) = \prod_i \hat{p}(y_i) \tag{355}$$

Parametric Model set of probability distribution indexed by $\theta \in \Theta$. $\{p(y;\theta)|\theta \in \Theta\}$.

In probabilistic modeling, analysing begins with : suppose the data are generated by a distribution in parametric family ...

Parametric model as hypotethis space.

- Poisson : $p(k,\lambda) = \frac{\lambda^k}{k!}e^{-\lambda}$.
- Beta : $p(y,a,b) = \frac{y^{\alpha-1}(1-y)^{\beta-1}}{B(\alpha,\beta)}$.
- Gamma family : $p(y;k;\theta) = \frac{1}{(k)\theta^k}y^{k-1}\exp(-\frac{y}{\theta})$.

7.22.2 Maximium likelihood estimator (MLE)

$$\hat{\theta} = \underset{\theta\in\Theta}{\operatorname{argmax}} \prod_i p(y_i,\theta) \tag{356}$$

Sometimes MLE does not exists (can be infinite for pdf).

7.22.3 Poisson example

$$\mathcal{D} = \{k_1,...,k_n\} \tag{357}$$
$$k_i \in \{0,...,\infty\} \tag{358}$$
$$\mathcal{L} = \sum_i \log(\frac{\lambda^k}{k!}e^{-\lambda}) \tag{359}$$
$$\lambda = \frac{1}{n}\sum_i k_i \tag{360}$$

7.22.4 Statistical Learning Formulation

Output space $\mathcal{Y}$ Action space $\mathcal{A} = \{p(y)|p$ probability density function on $\mathcal{Y}\}$. Loss :

$$l : (p,y) \mapsto -\log p(y) \tag{361}$$
$$R(p) = \underset{y\sim q}{E}[-\log p(y)] \tag{362}$$

And we also have an empirical risk with sums. So MLE is an empricial risk minizer.

Probability models can be Poisson distribution, Negative binomial distributions, histogram with 10 bins, Decision tree with histogram as leafs. Choose model with highest likelihood (histogram with bins for every y big overfit !).

7.22.5 Generalized Regression

How to represent probability distribution ? Parametric families. Logistic/Probit regression is a Bernoulli, Poisson regression is a poisson distribution. Linear regression is a normal distribution with fixed variance ....

Input space $\mathcal{X}$ output space $\mathcal{Y}$ (x,y) independent with distribution $P_{\mathcal{X}\times\mathcal{Y}}$.

Given a x predict a probabilistic distribution : $f(x)(y)$.

$$R(f) = -\underset{\mathcal{X},\mathcal{Y}}{E}[-\log f(x)(y)] \tag{363}$$

With its empirical risk minimizer. It's called a conditionnal log-likelihood.

7.22.6 Bernoulli regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{0, 1\}$. Predict $\theta = P(y = 1|x)$.

$$x \in \mathbb{R}^d \to w^T x \in \mathbb{R} \to f(w^t x) \in [0, 1] \tag{364}$$

f transfer or inverse link function. Logistic function yields logistic regression. Normal CDF yields Probit regression.

Learning by finding w with maximum log likelihood.

$$p_w(\mathcal{D}) = \prod f(w^T x_i)^{y_i}(1 - f(w^T x_i))^{1-y_i} \tag{365}$$

$$\text{Minimize} : J(w) = -\log p_w(\mathcal{D}) \tag{366}$$

7.22.7 Multinomial Logistic Regression

$$\mathcal{X} = \mathbb{R}^d \tag{367}$$

$$\mathcal{Y} = \{1, ..., k\} \tag{368}$$

$$\theta_i \geq 0 \tag{369}$$

$$\sum_i \theta_i = 1 \tag{370}$$

$$p(y|x) = \theta_y \tag{371}$$

$$x \to (\langle w_1, x \rangle, ...) \tag{372}$$

$$\to (\frac{\exp(w_1^T x)}{\sum_i \exp(w_i^T x)}, ...) \tag{373}$$

Soft-max function. We can also flatten this as for multiclass classification $\Psi(x, y) \in \mathbb{R}^{d \times k}$ and $w \in \mathbb{R}^{d \times k}$.

7.22.8 Poisson Regression

$\mathcal{Y} = \{0, 1, ...\}$.

$$f : x \mapsto Poisson(\lambda(x)) \tag{374}$$

$$\lambda > 0 \tag{375}$$

$$\lambda(x) = \exp(w^T x) \tag{376}$$

$$\log p(\mathcal{D}, \lambda) = \sum_i y_i w^T x - \exp(w^T x) - \log y_i! \tag{377}$$

SGD to optimize this.

7.22.9 Conditionnal Gaussian Regression

$\mathcal{Y} = \mathbb{R}$.

$$f(x) = \mathcal{N}(w^T x, \sigma^2) \tag{378}$$

$$w^* = \underset{w}{\text{argmax}} \sum_i \log p_w(y_i|x_i) \tag{379}$$

$$= \sum_i \log(\frac{1}{\sigma\sqrt{(2\Pi)}} \exp(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}))w^* = \text{argmin} \sum_i (y_i - w^T x_i)^2 \tag{380}$$

7.22.10 Generalized Linear Models : LITE

Natural Exponential Family : $\{p_\theta(y)|\theta \in \Theta \subset \mathbb{R}^d\}$.

$$p_\theta(y) = \frac{1}{Z(\theta)} h(y) \exp(\theta^T y) \tag{381}$$

$$h(y) > 0 \text{ Base measure} \tag{382}$$

$$Z(\theta) = \int_{\mathcal{Y}} h(y) \exp(\theta^T y) \text{ Partition function} \tag{383}$$

$$\Theta = \{\theta | Z(\theta) < \infty\} \text{ Natural Parameter Space} \tag{384}$$

$$\theta \text{ Natural parameter} \tag{385}$$

Specifying a natural exponential family is given h. We can make normal distribution with known variance, Poisson, gamma, Bernoulli.

$$\text{Poisson} : p(y,) = \exp(y \log \lambda - \lambda - \log y!) \tag{386}$$

$$\theta = \log \lambda \tag{387}$$

Note : This works if θ real number otherwise you can generalize with $\Psi : \mathbb{R} \to \Theta$ and $\Psi(w^T x)$.

Note on Poisson regression : you can gradient boost if you have a function $f(x)$ instead of dot product.

7.23 Bayesian Methods

Frequentist : θ is ficex constant unknwon to us. A statistic is a point estimation of $\hat{\theta}$. Desirable properties of point estimator :

- Consistency : As data size grows toward infinity, estimator converges.
- Efficiency : $\hat{\theta}_n$ is as accurate as we can get from a sample of size n.

MLE is both under good conditions.

Bayes : θ random variable.

- 1) Define model, choose prior $p(\theta)$ distribution, choose probability model or likelihood model $\{p(\mathcal{D}|\theta|\theta \in \Theta)\}$.
- 2) Compute posterior $p(\theta|\mathcal{D})$
- 3) Choose action based on posterior. Like MAP, or extract credible set for θ : $P(\theta \in [a,b]|D) \geq 0.95$. Choose loss function, $E[\theta|D]$.

posterior = likelihood times prior divided by marginal likelihood :

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{388}$$

7.23.1 Bernouilli coin flip

$$\text{Prior} : p(\theta) = \theta^{h-1}(1-\theta)^{t-1} \tag{389}$$

$$\text{Likelihood} : p(D|\theta) = \theta^{n_h}(1-\theta)^{n_t} \tag{390}$$

$$\text{Posterior} : p(\theta|D) = \theta^{n_h+h-1}(1-\theta)^{n+t+t-1} \tag{391}$$

7.23.2 Maximum a posteriori MAP

$$\hat{\theta} = \text{argmax} p(\theta|D) \tag{392}$$

7.23.3 Gaussian Regression Model

Assume y_i are conditionally independent given x and w.

$$p(y|x,w) = \prod p(y_i|x_i,w) \text{ conditionnal independence} \tag{393}$$

$$= \prod \frac{1}{\sigma\sqrt{2\Pi}} \exp(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}) \tag{394}$$

$$w^*_{MLE} = \text{argmin} \sum_i (y_i - w^T x_i)^2 \tag{395}$$

Choose a Gaussian prior $w \sim \mathcal{N}(0, \Sigma_0)$ with Σ_0 symetric positive definite.

$$p(w|D) = \prod \frac{1}{\sigma\sqrt{2\Pi}} \exp(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}) \times |2\Pi\Sigma_0|^{-1/2} \exp(-\frac{1}{2} w^T \Sigma_0 w) \tag{396}$$

7.23.4 Predictive distributions

Likelihood model is $y|x, w \sim \mathcal{N}(w^T x, \sigma^2)$. Best prediction is then $\hat{y}(x) = w^T x$. In bayesian we can compute posterior distribution.

7.23.5 1D example

$$\mathcal{X} = [-1, 1], \mathcal{Y} = \mathbb{R} \tag{397}$$

$$y = w_0 + w_1 x + \varepsilon \tag{398}$$

$$\varepsilon \sim \mathcal{N}(0, 0.2^2) \tag{399}$$

$$y|x, w_0, w_1 \sim \mathcal{N}(w_0 + w_1 x, 0.2^2) \tag{400}$$

$$w \sim \mathcal{N}(0, \Sigma_0 = \frac{1}{2} I) \tag{401}$$

$$w|D \sim \mathcal{N}(\mu_p, \Sigma_p) \tag{402}$$

$$\mu_p = (X^T X + \sigma^2 \Sigma_0^{-1})^{-1} X^T y \tag{403}$$

$$\Sigma_p = (\sigma^2 X^T X + \Sigma_0^{-1})^{-1} \tag{404}$$

$$\text{If} : \Sigma_0 = \frac{\sigma^2}{} I, \mu_p = (X^T X + \lambda I)^{-1} X^T y \tag{405}$$

Which is the ridge regression solution. All those estimators are random variables as function of the data. The assumption of gaussian noise is classic. OLS estimator $\hat{w}$ has a gaussian sampling distribution :

$$Cov(\hat{w}) = (\sigma^{-2} X^T X)^{-1} \tag{406}$$

$$\tag{407}$$

Same as Σ_p if $\Sigma_0^{-1} = 0$ or infinite variance. If $\Sigma_0 = \frac{\sigma^2}{} I$:

$$p(w|D) = \prod \frac{1}{\sigma\sqrt{2\Pi}} \exp(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}) \times |2\Pi\Sigma_0|^{-1/2} \exp(-\frac{\lambda}{2\sigma^2} w^T w) \tag{408}$$

$$\hat{w}_{MAP} = \text{argmin} \sum_i (y_i - w^T x_i)^2 + \lambda \|w\|^2 \tag{409}$$

Ridge regression objective.

Predicting y_{new} from x_{new} :

$$p(y_{new}|x_{new}, D) = \int_{\theta} p(y_{new}|x_{new}, \theta, D)p(\theta|D)d\theta \tag{410}$$

$$y_{new}|x_{new}, D \sim \mathcal{N}(\eta_{new}, \sigma_{new}) \tag{411}$$

$$\eta_{new} = \mu_p^T x_{new} \tag{412}$$

$$\sigma_{new} = x_{new}^T \Sigma_p x_{new} + \sigma^2 \tag{413}$$

Huge advantage : Error bands.

8 Deep Learning

8.1 Introduction

8.1.1 Non-linearly separable problems

If we have P examples of N features, probability of model linearly separable decrease a lot with P growing.

How to deal with them with a linear classifier :

- Making N larger with cross-product terms. polynom of degree d we have N^d features so impractical for big N.
- Making N larger with gaussian bumps. Grow exponentially with N also.
- Kernels.

$$f(X, W) = \sum_{k=1}^{K} W_k K(X, X^k) \tag{414}$$

- Sparse Basis functions :
  - Idea 1 : Use unsupervised learning function such as k-means to create k centroids.
  - Idea 2 : have K random centroids and adjust them with gradient descent.

$$F(X, W, U_1, ..., U_k) = \sum_{k=1}^{K} W_k K(X, U_k) \tag{415}$$

- Random directions : Randomly split the space in lots of little domains, use many variables of type $q(W^k X)$, q threshold function. It's the original perceptron algorithm.

8.1.2 Feature extraction before

Speech recognition : MFCC + Mix of gaussians.

Object recognition : SIFT / HoG

8.2 Convolutionnal Neural Networks

Image recognition : Only way to go is convnets.

8.2.1 Parameter transform

Imagine parameters of a Linear layer, or every layers for that matter are W. You can parametrize $W = G(U)$ and learn G, as long as it is differential.

One particular transform which is useful is weight sharing or 'tied' weights. Same weights values in multiple places in a convnet. When you train, the gradient with respect of this weight is the sum of gradients everywhere this weight appears.

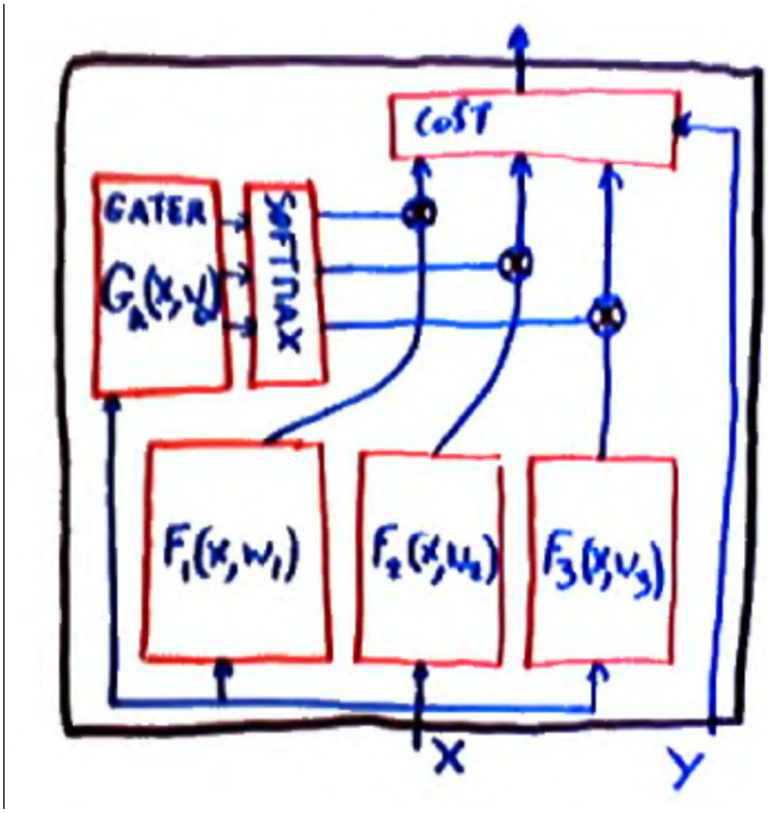


Figure 8: Attention or gating mechanism

8.2.2 Gating mechanism

Language models use attention mechanisms. You have multiple neural networks which are specialized in sub-tasks. For example : American english vs Brisith English speech recognition. One neural network has for task to judge which expert is the one to be used. "Mixture of experts". Used for language translation. Which combination of words in the input sentence does the word I have to predict have to translate.

8.2.3 Time-delayed neural networks

Predict x_t given $x_{t-1}, ...x_{t-k}$.

For instance old Modems used a 1 layer perceptron to recreate the given signal and be able to understand bits, thus the noise at the beginning which go through the whole usable spectrum and detect bits. Loss close to competitive learning idea. It's somewhat a clustering problem.

$$L = \sum_i \frac{1}{p}(\sigma(w^T x) - w^T x)^2 \tag{416}$$

$$w \leftarrow w - \eta(\sigma(w^T x) - w^T x) \tag{417}$$

8.2.4 Speech recognition

McGurk effect : If you hear ba and see pronouncing ga you think you hear ga. After this prepreocessing you have a neural network that tries to predict which phonem the middle of the time window of 0.5 seconds is over 3000 phonems approx.

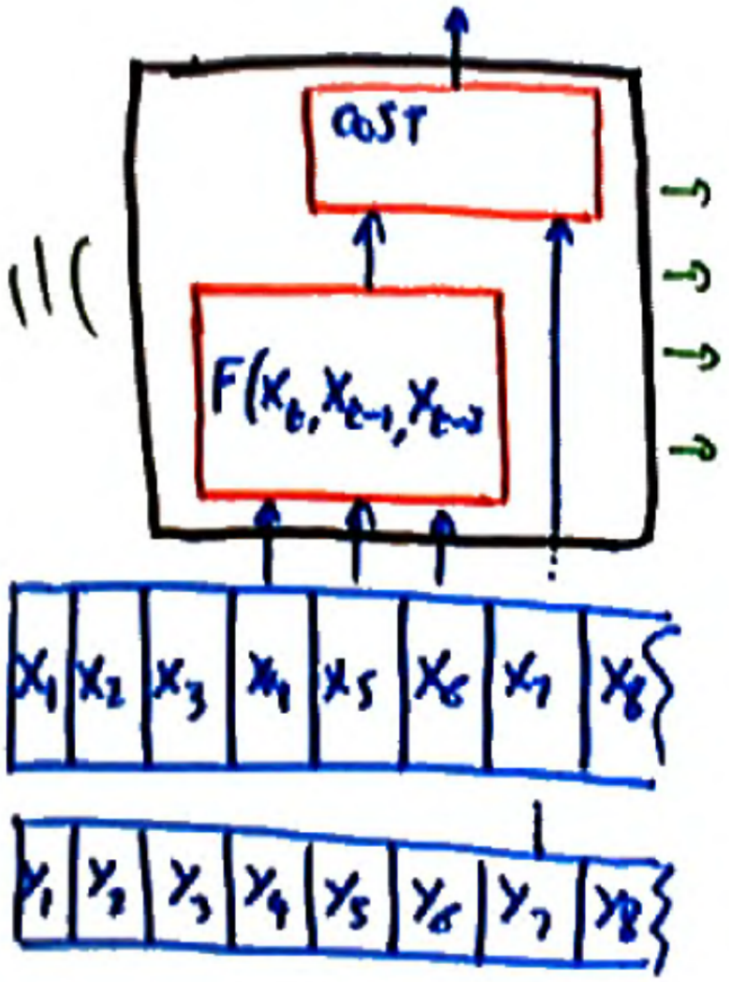


Figure 9: Time-delayed prediction

8.2.5 Pooling

Note : pooling is different of convolution in that it does not depend on the order of the input.

- Average Pooling.
- Max Pooling.
- L_p pooling.
- Log sum exponential pooling : $\frac{1}{b}\log(\sum \exp(bX_i))$. Same as max pooling when $b \to \infty$.
- Sort pooling.
- Variance pooling.

Important remark : You can apply ConvNet to larger images for a very cheap cost as if you slightly move the window most weights are somewhere already computed. So if you build Intelligently your neural network it should be almost free.

8.2.6 When to use convolutions

- Time series predictions.
- Speech recognition.

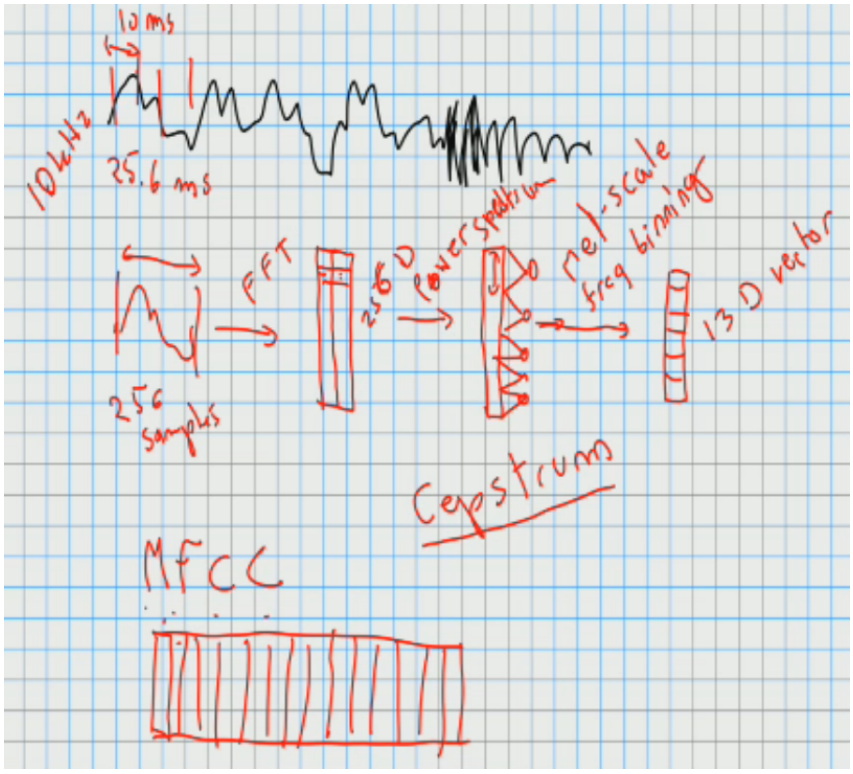


Figure 10: Speech preprocessing pipeline

- Prepossessing speech actually.
- Image, video, color videos, MRI, video MRI.

Convolution because neighbouring values are correlated, and far away values (pixels) aren't. If you take a $5x5$ patch of images and go through all possible natural images, you won't goo through the whole manifold of size 25 of possible pixels, but actually only a small part. It's because they are correlated.

You can represent the context of this $5x5$ patch not by the presence or absence of all possible values, but by the presence or abscence of a particular list of motives. It's actually a trick used for image compression : JPEG use $8x8$ patches with fourier transform to encode what's in the patch. JPEG 2000 use wavelet transform.

8.2.7 Object detection

Detect objets within a image. Here the big problem is scale. For instance face detection : they are various sizes in an image of faces. What you can do is take an image, run the face detector, if you don't detect faces zoom in (scale image), re-run ... Until you are at a scale where the image detector lights up and then you actually have information on the scale and position within an image of all faces.

2 class classifiction : NN in general are poor at 2 class classification. So pretrained on a lot of categories, like image net, then fine tune.

Faces are quite similar : A big issue is the fact that faces are quite similar but the amount of things that are not faces is huge, so if you train a neural net to detect faces or not, he will not have seen all cases things that are not faces (which is everything else than faces...). Solution : Negative Sampling. You train once your face detector on your dataset. Go through a dataset of things that are not faces, select the ones false positive, retrain on this augmented dataset, and keep on going this step until you have a good face detector. Also when you want to classify and locate object, multi-scale sliding

window approach is cheap and good. For instance bounding box detection. Skip connection instead of strides for more accurate prediction.

Deep Face : Detect key points of faces, fit average 3D model of face, unwrap and straighten the face, once frontal use ConvNet for classification.

8.2.8 Semantic Segmentation

Giving a label to every pixel in the image (street, building, sky ...). Very useful for self driving cars, knowing where you can drive or not. But number of samples in those datasets are very limited, as it is super painful to label. Coco has 80 categories for instance.

There is a multiple scale problematic to this problem. If you use $8x8$ filters, each region of $8x8$ has a label, which in terms of resolution is not optimal.

If you see a window of size $8x8$ which is totally grey, it could either be the sky or the road, you don't know. You need context to decide. What people do is you take the image feed it as if, as well as the reduced version by 2, 4 ... It gives better context to decisions.

8.2.9 Weak learning

You don't give the full information to you system. Example : predict digits but this time you have a couple of digits on a picture, and you say which digits they are but not the order of digits.

8.2.10 Graph transformer networks

See the example of digits recognition on checks. You can feed a graph to your neural network, nodes and edges. Each node and edge can be tensors.

8.2.11 Common architectures

VGG : Visual Geometry Group a bunch of $3x3$ or $5x5$ convolutions, good because you can have a lot of layers.

You have :

- Input n, convolution kernel k, output $n-k+1$.
- Input n, pooling stride s kernel k, output $(n-k+1)/s$. Useful for not having a huge network.
- Skips : Solution to a semantic segmentation problem : If an area has one label because of

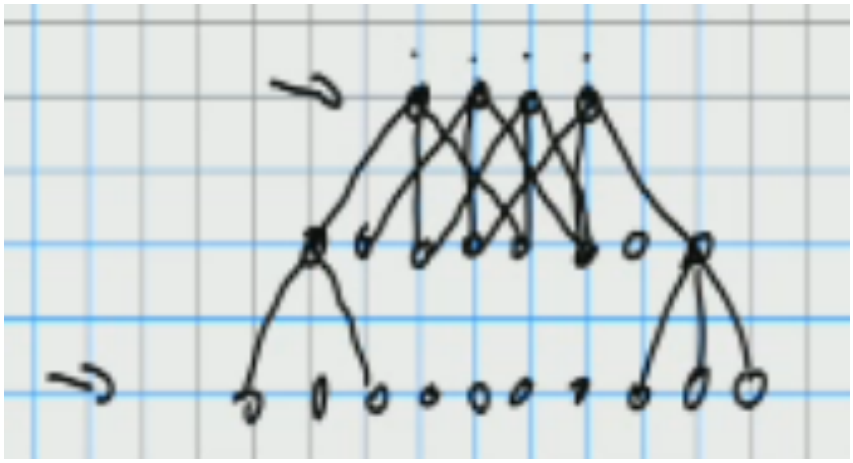


Figure 11: Skip connections

convolution, skip solve this problem. After a skip you see uniquely your input.

Inception Network Motivations : Neural networks tends to have too much weights, inceptions solve this issue. Also it makes the footprint lower so that Google can run it on the cloud efficiently.

ResNet If a layer as small weights, it dies and information do not get before it. Here the default signal is the identity by default, and the residual layer modify the identity by a residual. If the layer die it's okay. Now we can make super big networks without problems. It gets around the limitations of back-propagation.

8.2.12 Metric Learning

Siamese neural network. If you want to detect if two faces are from the same person. You can't see that as a multi-class classification with 6 billion classes. What you do is Siamese loss : Train a neural network to have close vectors when same face, and far-away vectors when faces are not from the same person. See DrLIM for dimensionality reduction by learning an invariant mapping.

$$L_{same} = \|G_w(x_1) - G_w(x_2)\|^2 \tag{418}$$

$$L_{diff} = [m - \|G_w(x_1) - G_w(x_2)\|^2]^+ \tag{419}$$

8.2.13 Convolution and fourier space

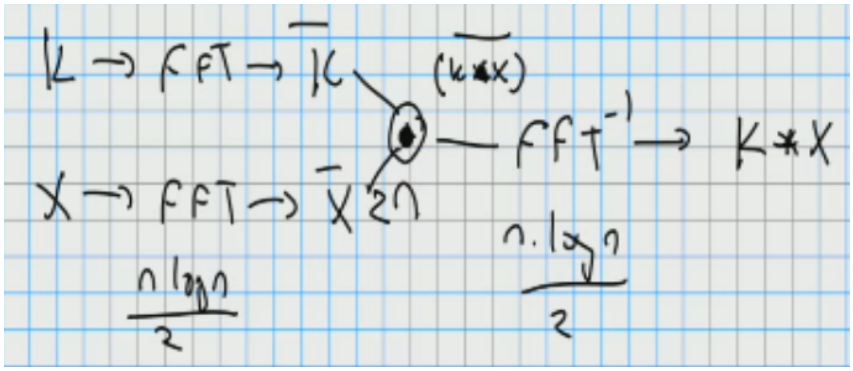


Figure 12: Fourier Transformation

8.2.14 Graph Laplacian

The graph laplacian is more or less a matrix with difference between value of a node and the sum of the values of all nodes it's connected to. The eigenvectors of that laplacian matrix is the fourier space. Fourier transformm = eigenspace of the graph laplacian.

$$L = \text{ Graph laplacian, } nxn \text{ matrix.} \tag{420}$$

$$L = Q^T \Delta Q \tag{421}$$

$$X \text{ Transformation on the graph, vector.} \tag{422}$$

$$QX : \text{ Fourier transform of } X. \tag{423}$$

$$K \star X = Q^T(QK \otimes QX) \tag{424}$$

8.3 Recurrent Neural Networks

Why :

- If you wnat a network to take into account things from the past, and don't know for how long to take it into account.
- Language models.

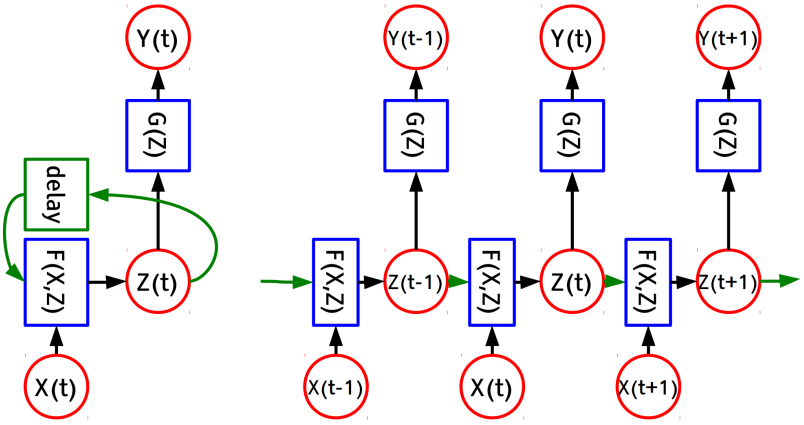


Figure 13: Recurrent Neural Network

There is still a finite horizon : How many steps you do before a back prop.

Usually $Z(t-1) \rightarrow Z(t)$ is a single layer and non-linearity. With mutilple layers you could do more complicated stuff though.

8.3.1 Stacking RNNs

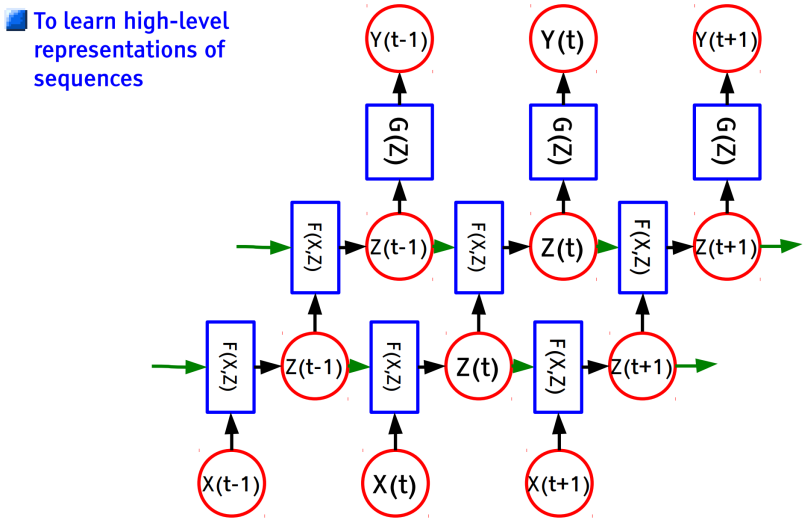


Figure 14: Stacked Recurrent Neural Network

8.3.2 RNN are incompatible with back prop

Vanishing gradient problem, exploding gradient problem :

$$f(t-1) = \nabla f(t) \times J_F \tag{425}$$

J_F jacobian, if its eigenvalues are not 1 then either the signal decrease a lot or increase a lot after a couple of steps backward. Huge problem.

Also, if RNN where to remember stuff, they should have stable memory states, it means that for several different initial states they should converge to the same state. What it means is that initial

state should not influence end state, thus you back prop no gradient. Stable memories implies back propagation would not work. This argument is not totally true as you could have stable orbits and not stable memory states.

8.3.3 RNN with some convolution behaviours

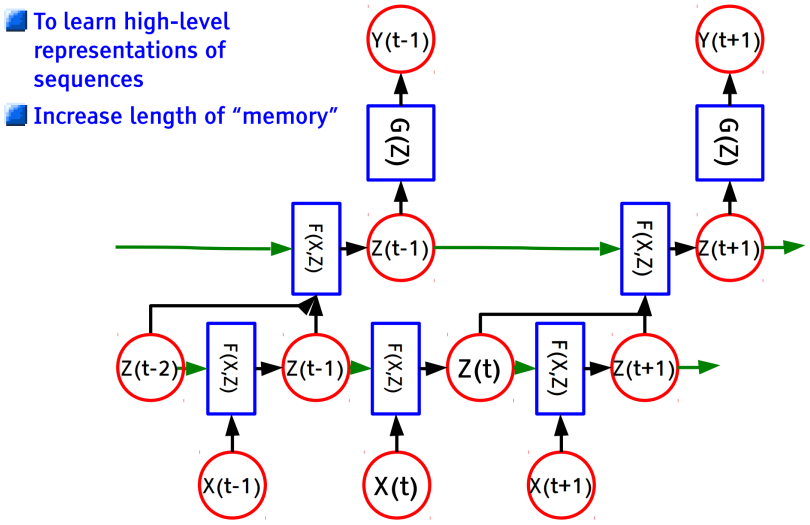


Figure 15: 2nd layer having a convolutional behaviour

It helps with memory.

On a basic architecture, you want by default that the memory state do not change, an identity function. What you can do is slow down evolution of hidden state by having two parts in the memory state : a fast-changing memory state and a slow changing one. You force the F function to be near the identity function for the slow changing part of the memory. The slow moving part of the memory state is the context feature.

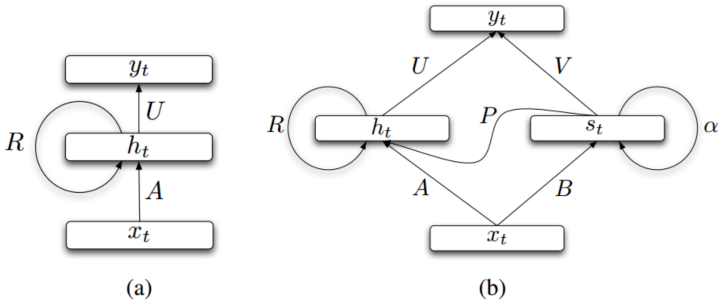


Figure 1: (a) Simple recurrent network. (b) Recurrent network with context features.

Figure 16: RNN with context features

8.3.4 Delay RNN

8.3.5 Tricks to make it work

- Gradient clipping for exploding gradient problem.
- Momentum.
- Careful Initialization.

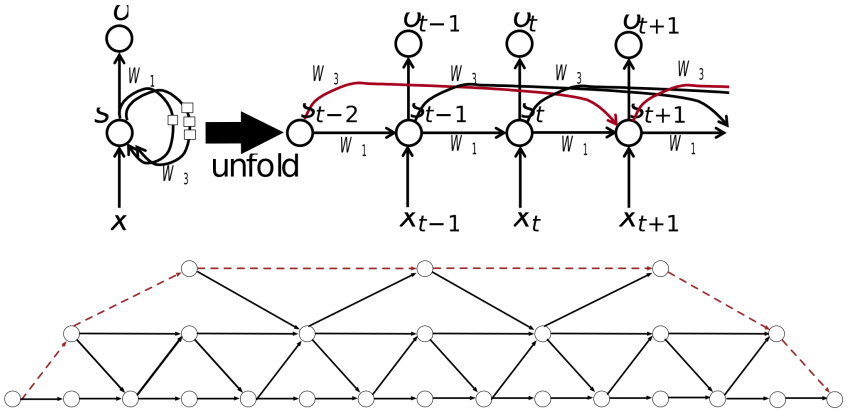


Figure 17: RNN with delay

- Sparse gradient.
- Leaky integration.
- Gradient regularizer for vanishing gradient problem.
- LSTM.

8.3.6 Long Short-Term Memory Cell

Used almost all the time. Simpler version which works great also : GRU unit.

Used for instance in sentence translation, with 4 layers in this case. You have one-hot encoded words, with a linear layer (an encoder, which is a trainable look-up table in this case). It can create sentences, but as you need to put as input the last created word, it become sequential thus damn long to run.

The best in sentence translation is attention based networks. Also note that there is multiple translation to a single sentence, thus we should use some context when translating sentences.

8.3.7 Language Modeling

Very useful for speech recognition for instance : you have more or less what the phonems of the current word, and by predicting what should be the next word you are better at predicting the actual said word. Also keep in mind that soft-max over 1 million word is damn expensive and there is a lot of litterature on how to make it computable.

Perplexity It's the measure of performance of language models. It's the average negative log probability my model give to the actual next word. If $\log_2$ it's in bits, how much information you should add for your system to actually produce the correct word.

Bidirectionnal LSTM It's a trick where, as you tend to forget after 20 words what the beginning of the sentence was about, you actually run your LSTM from beginning to end and from end to beginning.

Can look at :

- Viterbi decoding.
- Bin Search : You start with every possible words, then predict next word based on language model and last word : keep the best 1000 2 words. Then go on. It's kind of shortest path in a graph with dynamic coding.
- Orthogonal RNN : Making F orthogonal so that Jacobean eigenvalues are 1. Constraining that is not efficient, penalty term is better $\|A^T AX - X\| \to 0$.

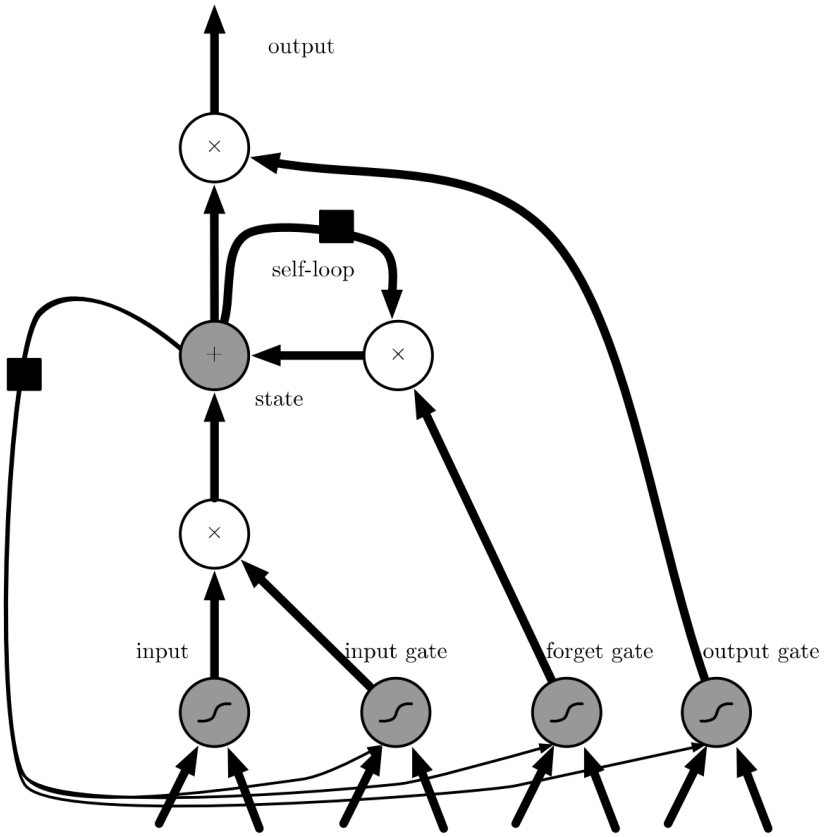


Figure 18: LSTM cell

8.3.8 Memory Augmented Networks

Augment Neural Network with differential memory. Also look at neural tree indexes, recursive convnets.

Idea of recursive convnets : After a point the input of a convolution and its output should be in the same space. For instance : the word "the" and "cat" can be represented in the same space as "the cat".

9 Causal Inference

9.1 Potential Outcome Model

9.1.1 Introduction

Potential outcomes are : Y^1 if treated, Y^0 if not treated. For each individual you can define $y_i^{\{0,1\}}$ and the individual treatment causal effect as :

$$\delta^i = y_i^1 - y_i^0 \tag{426}$$

But we observe only one of them : y_i^1 or the other ... What would happens if i was treated/not treated ?

9.1.2 Treatment

Random treatment $D \in \{0, 1\}$. The observed outcome is :

$$Y^* = DY^1 + (1-D)Y^0 \tag{427}$$

The average treatment effect (ATE) is :

$$ATE = E[\delta] = E[Y^1] - E[Y^0] \tag{428}$$

Which is the difference of probabilities if Y is binary. We can also look at the ratio (causal risk ratio).

9.1.3 SUTVA

STABLE UNIT TREATMENT VALUE ASSUMPTION.

Under SUTVA :

- There is no interference: the potential outcomes of an individual is unaffected by the treatment administered to other individuals.
- There is no hidden variation of treatment: treatment is deterministic and exists under only one form.

SUTVA is not always met in practice :

- Contagion : Vaccination ...
- Equilibrium : Too many high-skilled workers ...
- Classrooms, peer effect ...

9.1.4 Assignment mechanism

Distribution of D_i given potential outcome Y_i^0, Y_i^1 and X_i.

Some desirable properties :

- Individualistic : The treatment of unit i cannot depend on the potential outcome.
- Probabilistic : for every unit i, the probability of each state of the treatment is positive.
- Unconfounded : the assignment mechanism is independent on potential outcomes, conditional on the co-variates (more on this soon).

Some authors define :

- Classical randomized experiments: the three desirable properties are known, and the form of the assignment mechanism is known and chosen (RCTs).
- Regular assignment mechanisms: same as before, but the form of the assignment mechanism is not known or chosen (quasi-experimental setting in observational studies).
- Irregular assignment mechanisms: observational studies where one of the desirable properties, most typically unconfoundedness, fails.

Perfect randomization :

$$(Y^1, Y^0) \perp D \tag{429}$$

This is rarely the case in reality, and it does not imply that the observed outcome is independent from D.

We have then :

$$E[Y^*|D=1] = E[Y^1|D=1] = E[Y^1] \tag{430}$$
$$E[Y^*|D=0] = E[Y^0|D=0] = E[Y^0] \tag{431}$$

Example : If $U \sim \mu$ RV unobserved and $Y^i = f_i(U)$. Perfect randomization equals : $U \perp D$
Usually the weaker condition is sufficient :

$$Y^1 \perp D \ Y^0 \perp D \tag{432}$$

9.1.5 Average causal effect

$$ATT = E[\delta|D=1] = E[Y^1|D=1] - E[Y^0|D=1] \tag{433}$$
$$ATC = E[\delta|D=0] = E[Y^1|D=0] - E[Y^0|D=0] \tag{434}$$

ATT : Average Treatment effect of the Treated ATC : Average Treatment effect of the Control group.
Note that $E[Y^i|D=1-i]$ is not known, so naive estimator :

$$\delta_{naive} = E[Y^1|D=1] - E[Y^0|D=0] \tag{435}$$

Under randomized treatment, we have $ATT = ATC = \delta_{naive} = ATE$.
$E[Y^0|D=1]$ is : The outcome if you have not had the treatment, given that you had the treatment.
$\pi = P(D=1)$

$$ATE = E[\delta] = \pi E[Y^1|D=1] + (1-\pi)E[Y^1|D=0] + \pi E[Y^0|D=1] + (1-\pi)E[Y^0|D=0] \tag{436}$$
$$= \delta_{naive} - (E[Y^0|D=1] - E[Y^1|D=0]) - (1-\pi)(E[\delta|D=1] - E[\delta|D=0]) \tag{437}$$

With :

- $(E[Y^0|D=1] - E[Y^1|D=0])$ baseline bias.
- $(1-\pi)(E[\delta|D=1] - E[\delta|D=0])$ differential treatment effect bias.

9.1.6 Propensity score

X a vector of co-variates (age, sex ...)
The propensity score is the probability of being assigned to the treatment group given features.

$$e(x) = P[D=1|X=x] = E[D|X=x] \tag{438}$$

Overlap assumptions : we assume $e(x) \in]0,1[$.

9.1.7 Unconfoundedness

$$(Y^1, Y^0) \perp D|X \tag{439}$$

Unconfoundedness implies :

$$\pi_{Y,D|X}(y,d|x) = \pi_{Y|X}\pi_{D|X} \tag{440}$$
$$\pi_{YDX} = \pi_{XY}\pi_{D|X} \tag{441}$$
$$\pi_{D|X}(1|x) = e(x) \tag{442}$$
$$\pi_{D|X}(d|x) = 1 - e(x) + d(2e(x)-1) \tag{443}$$
$$\pi_{YDX} = \pi_{XY}(1 - e(x) + d(2e(x)-1)) \tag{444}$$
$$\pi_{YD|e}(y,d|e(X)=e) = f(e,d)\pi_{Y|e}(y|e(x)=e) \tag{445}$$

Under unconfoundedness we have :

$$E[Y^*|D=1,X] = E[Y^1|X] \tag{446}$$
$$E[Y^*|D=0,X] = E[Y^0|X] \tag{447}$$

because :

$$f_{XYD}(x,y,d) = f_X(x) f_{Y|X}(y|x) f_{D|X}(d|x) \tag{448}$$

The conditional ATE is obtained by :

$$ATE(x) = E[Y^1|X] - E[Y^0|X] = E[Y|D=1,X] - E[Y|D=0,X] \tag{449}$$
$$ATE = E[ATE(X)] \tag{450}$$

Under unconfoundedness we do not have :

$$ATE = E[Y^*|D=1] - E[Y^*|D=0] \tag{451}$$
$$ATE = P(X=0)ATE(0) + P(X=1)ATE(1) \tag{452}$$

Both quantities coincides if $D \perp X$ which is the same as perfect randomization.

9.1.8 Discrete case

$$\hat{ATE}(x^k) = \frac{1}{|\{x_i = x^k, d=1\}|} \sum_{x_i = x^k, d=1} y_i - \frac{1}{|\{x_i = x^k, d=0\}|} \sum_{x_i = x^k, d=0} y_i \tag{453}$$

$$\hat{ATE} = \sum_{x^k} \frac{|\{x_i = x^k\}|}{|\{x_i\}|} \hat{ATE}(x^k) \tag{454}$$

9.2 RCTs and field experiments

$Y^* = Y^D$ for observed outcome.

9.2.1 Regression

One way to estimate the ATE :

$$Y_i^* = \alpha + \beta D_i + \varepsilon_i \tag{455}$$

If random treatment, D_i and ε_i are independent.

Under unconfoundedness, we have :

$$Y^0 = \alpha_0 + \beta_0 X + \varepsilon_0 \tag{456}$$
$$Y^1 = \alpha_1 + \beta_1 X + \varepsilon_1 \tag{457}$$
$$\text{Unconfoundedness} : (\varepsilon_0, \varepsilon_1) \perp D|X \tag{458}$$
$$ATE(x) = \alpha_1 - \alpha_0 + (\beta_1 - \beta_0)x \tag{459}$$
$$ATE = \alpha_1 - \alpha_0 + (\beta_1 - \beta_0)E[X] \tag{460}$$

9.2.2 Field experiments :

Important factors :

- Pool of subject.
- Information brought to participants.
- Incentive mechanism.
- Task asked from participants.
- Stakes participants have in common.
- Environment.

9.2.3 Randomization

Various ways of randomization :

- Over-subscription (limited budget for treatment).
- Randomized phase-in. Control group : area waiting to get the treatment.
- Within-group randomization : Some subgroups treated in each targeted area, risk of spillover.
- Encouragement design : Announcement of the program/incentive to participate, randomly assigned.

Issues : Ethical / political. Internal / External validity of experiment (too low treatment group...). Imperfect compliance (people go out of treatment.). Spill-over. "Hawthrone effect", "John henry effect" people behaviour different is they know ther are watched.

9.2.4 Attrition

School class size : attrition is the fact that some students leave the sample. Students with better grades are more likely to quit as they have more options. One way to cope with this is to impute the score of students who leave. Prediction based on past results.

9.2.5 Charitable giving

Why do people give money ? Either they care about a cause or they feel socially pressured. In order to test this: 3 experiences. One people come to your house without notice. Two people leave a flyer saying they will be there tomorrow. Three people leave a flyer and you can mark whether or not you want them to come. Test with two hospitals one nearby one far away.

9.2.6 Absenteeism rate in India

India you are paid whether you come to class or not. Test a method were you are paid $1/3$ of your salary no matter what and if you come more than 10 days, then you start ganing money (1500 max instead of 1000).

9.3 AB-testing

9.3.1 Basic recipes

- Avoid spillovers.
- Feedback mechanism.
- Simpler is better.
- Care unanticipated difficulties (legal actions ...).

9.3.2 Bernoulli trials

Conditional probability of being in the treatment group given by the propensity score $e(x)$. Bernoulli : N units are assigned in an i.i.d. manner to control/treatment group.

$$D \in \{0,1\}^N \tag{461}$$

$$P(D|X,Y^0,Y^1) = \prod_{i=1}^{N} e(X_i)_i^D (1-e(X_i))^{1-D_i} \tag{462}$$

$$\text{When treatment probability uniform : } e(X) = q \tag{463}$$

$$P(D|X,Y^0,Y^1) = q_1^N (1-q)_0^N \tag{464}$$

$$N_1 = \sum_i D_i = N - N_0 \tag{465}$$

9.3.3 Completely randomized experiment

Size N_1 is fixed and drawn at random without replacement.

$$P(D|X,Y^0,Y^1) = \frac{1}{\binom{N}{N_1}} \tag{466}$$

9.3.4 Stratified experiment

Population divided in strata or blocks (eg. men vs women, number of children ...). Within each blocks $j \in J$ a completely randomized experiment is done. Each block has size $N(j) = N_0(j) + N_1(j)$.

$$P(D|X,Y^0,Y^1) = \prod_{j \in J} \binom{N(j)}{N_1(j)}^{-1} \tag{467}$$

9.3.5 Pairwise experiment

Stratified experiments where blocks are of size two. One is treated the other is not. N even, $j \in J = \{1,...,N/2\}$ has probability $1/2$ of being treated.

$$P(D|X,Y^0,Y^1) = \frac{1}{2^{N/2}} \tag{468}$$

9.4 Sample uncertainty

Example : Fundraising e-mail, treated have a photo of a child in it. Asks for $\$500/1000/2000$.

9.4.1 Tests for completely randomized experiments

Fisher's sharp null hypothesis : $Y_i^0 = Y_i^1$. We need a test statistic which given X, Y^*, X gives us how likely is our observation. First test is estimated ATE :

$$T^{diff} = |E[Y^*|D=1] - E[Y^*|D=0]| \tag{469}$$

Or classifcal t-statistics :

$$T^{t-stat} = \frac{T^{diff}}{\sqrt{\frac{s_0^2}{N_0} + \frac{s_1^2}{N_1}}} \tag{470}$$

$$s_d^2 = \sum_{i,D_i=d} \frac{(Y_i^* - \hat{\mathbb{E}}[Y^*|D=d])^2}{N_d - 1} \tag{471}$$

9.4.2 Fisher's p-value

We can look at the distribution of T^{diff} under H_0. Consider all possible draws where the treatment group is 4000 and control group 6000 ($\binom{10000}{6000}$). For each $\tilde{D}$ such distribution compute T^{diff}. Simulate T^{diff} and compute $P(T^{\tilde{diff}} > T^{diff})$. If inferior of confidence α this is unusual.

9.4.3 Repeated sampling, neymann estimator

ATE estimated under perfect randomization is :

$$\hat{ATE} = \frac{\sum_i Y_i^1 D_i}{N_1} - \frac{\sum_i Y_i^0 (1 - D_i)}{N_0} \tag{472}$$

Sometimes called Neyman estimator. Conditional on pontential outcome, if $\tilde{D}$ drawn from $\{0,1\}^N$ with $\sum_i D_i = N_1$ we can show :

$$var(\widehat{ATE}) = \frac{s_0^2}{N_0} + \frac{s_1^2}{N_1} - \frac{s_{01}^2}{N} \tag{473}$$

$$s_d^2 = \frac{1}{N-1} \sum_i (Y_i^d - \bar{Y}^d) \text{ can be computed.} \tag{474}$$

$$s_{01}^2 = \frac{1}{N-1} \sum_{i,D_i=d} (Y_i^1 - Y_i^0 - \bar{Y}^1 + \bar{Y}^0)^2 \text{ can't be computed.} \tag{475}$$

We only need an upper-bound on the variable to get confidence interval !

$$\hat{V}^{Neymann} = \frac{s_0^2}{N_0} + \frac{s_1^2}{N_1} \geq var(\widehat{ATE}) \tag{476}$$

$$90\ \%\text{ confidence interval : } [\widehat{ATE} - 1.645\sqrt{\hat{V}^{Neymann}}, \widehat{ATE} + 1.645\sqrt{\hat{V}^{Neymann}}] \tag{477}$$

9.4.4 Linear regression case

In a completely randomized experiment, the linear regression specification :

$$Y_i^* = \alpha + \tau D_i + \beta' X_i + \varepsilon_i \tag{478}$$

$$\text{OLS estimator : } \min_{\alpha,\tau,\beta'} \hat{\mathbb{E}}\left[(Y_i^* - (\alpha + \tau D_i + \beta' X_i))^2\right] \tag{479}$$

We get $(\alpha^*, \tau^*, \beta^*)$ and :

$$\tau^* = ATE = \mathbb{E}[Y^1 - Y^0] \tag{480}$$

$$\sqrt{N}(\tau - \tau^*) \to \mathcal{N}(0, V^{OLS}) \tag{481}$$

$$V^{OLS} = \frac{\mathbb{E}\left[(D_i - \mathbb{E}[D_i])^2 (Y_i^* - (\alpha^* + \tau^* D_i + \beta^{*\prime} X_i))^2\right]}{\mathbb{E}[D_i]^2 \mathbb{E}[1 - D_i]^2} \tag{482}$$

9.4.5 Model based imputation

Assume we have a model for the potential outcome. θ parameter assumed to be drawn from prior $p(\theta)$. (Y^1, Y^0) vector of potential outcome. D vector of treatment and $f(Y^1, Y^0|\theta)$ assumed to be known. Under complete randomization we have :

$$f(Y^0, Y^1, D|\theta) = f(Y^0, Y^1|\theta)f(D) \tag{483}$$

$$Y^- \text{ missing data : } Y_i^- = Y_i^{1-D_i} \tag{484}$$

9.4.6 Bayesian inference

- Step 1 : Compute $f(Y^-|Y^*, D, \theta)$.
- Step 2 : Compute $f(\theta|Y^*, D)$.
- Step 3 : Compute $f(Y^-|Y^*, D)$.
- Step 4 : If param. of interest is $\tau = \tau(Y^0, Y^1, D) = \tau(Y^-, Y^*, D)$ then the distribution can be inferred from $f(Y^-|Y^*, D)$.

See lecture 4 slide 20 for explenation of the method.

9.5 Observational studies: propensity score and matching estimators

In observational studies, assignment mechanism is unknown. Unconfoundedness is an assumption here. It is not testable !

Given :

$$(D, X, Y), D \in \{0, 1\}, X \in \mathbb{R}^d, Y^* \in \mathbb{R} \tag{485}$$

$$\tag{486}$$

We can write two potential outcomes models (D, X, Y^0, Y^1) compatible with experiment where unconfoundedness holds in one but not the other. So we need to assess whether unconfoundedness is plausible or not. Also the propensity score is unknown and needs to be estimated.

9.5.1 Estimating Propensity score

If X discrete, easy. But if continuous, it's a zero-probability event !

Overlap condition : $0 < e(X) < 1$ for all x in support of P_X.

Under unconfoundedness we have : $ATE = E[ATE(X)] = E[ATE(e)]$.

9.5.2 Efficiency Bound

Is it possible to compute the best possible accuracy that an estimator can achieve when sample size gets large ?

Semi-parametric efficiency bound x, for any estimator $\widehat{ATE}$:

$$\lim_{N\to\infty} var(\widehat{ATE} - ATE) \geq E[\frac{V^1(X)}{e(X)} + \frac{V^0(X)}{1 - e(X)} + (ATE(X) - ATE)^2] \tag{487}$$

$$V^d(X) = var(Y^d|X) \tag{488}$$

9.5.3 Parametric Regression

$$E[Y^d|D = d, X = x] = \alpha_d + \beta_d' x \tag{489}$$

So we regress Y_i^* on X and we have :

$$ATE(x) = \hat{\alpha}_1 - \hat{\alpha}_0 + (\hat{\beta}_1 - \hat{\beta}_1)'\bar{X} \tag{490}$$

Most popular approach. Nothing guarantees that it should be linear.

9.5.4 Blocking

Stratified approach :

$$\mathcal{X} = \mathcal{X}_1 \cup ... \cup \mathcal{X}_n \tag{491}$$

$$\sum_{k=1}^{K} Pr(X \in \mathcal{X}_k) ATE_k \tag{492}$$

If bins too large, we have naive estimator.

9.5.5 Kernel estimation

$$E[Y^d|D = d, X = x] = \frac{\sum_{i=1}^{n} K(\frac{X_i - x}{h}) Y_i 1_{D_i = d}}{\sum_{i=1}^{n} K(\frac{X_i - x}{h}) 1_{D_i = d}} \tag{493}$$

h should be decreased when N is large. If $K(z) = 1_{|z| \leq 1}$ it's approx. blocking. Typically gaussian kernel.

Horvitz-Thompson estimator.

$$\tau^{HT} = \frac{1}{N} \sum_{i=1}^{N} \frac{D_i Y_i^*}{e(X_i)} - \frac{1}{N} \sum_{i=1}^{N} \frac{(1 - D_i) Y_i^*}{1 - e(X_i)} \tag{494}$$

Because :

$$E[\frac{DY^*}{e(X)}] = E[\frac{E[DY^*|X]}{e(X)}] = E[\frac{E[D|X]E[Y^*|X]}{e(X)}] = E[Y^1] \tag{495}$$

$$E[\frac{(1 - D)Y^*}{1 - e(X)}] = E[Y^0] \tag{496}$$

In perfectly randomized experiment, $e(X) = N_1/N$ and we recover naive estimator. But often we don't know the propensity score.

9.6 Matching Estimators

For each treated unit i, look at the closest possible resemblance in untreated unit. Some match on similarity co variates, or propensity score ...

Typical estimator is :

$$\frac{1}{N_1} [\sum_{i \in I_1} (Y_i^* - \sum_{j \in I_0} \omega_{ij} Y_j^*)] \tag{497}$$

with ω_{ij} captures the similarity between propensity scores. Advantage : No need to have propensity score !

- k-NN
- Kernel method :

$$\omega_{ij} = \frac{K(\frac{x_i - x_j}{h}) 1_{D_i \neq D_j}}{\sum_{k \neq i} K(\frac{x_i - x_j}{h}) 1_{D_i \neq D_j}} \tag{498}$$

- propensity score matching : replace X by $e(X)$, good against curse of dimensionality. but $e(X)$ has to be estimated. Then get the support of $e(X)$, then use one of the previous methods.

9.7 Propensity score estimation

Logistic regression, with maximum likelihood loss function :

$$e_\Phi(x) = \frac{\exp(X'\Phi)}{1 + \exp(X'\Phi)} \tag{499}$$

Can be computer with gradient descent methods. If U follows a logistic distribution with c.d.f. :

$$F_U z = \frac{1}{1 + \exp(-z)} \tag{500}$$

Assignment mechanism can be interpreted as : $D = 1_{X'\Phi \geq U}$.

9.8 Regression Discontinuity

Sometimes a threshold determines eligibility to a program, thus regression discontinuity. Applicable when criterion is clear and explicit.

Example :

treatment = college scholarship, X = standardised test score, Y^1 earnings if eligible to scholarships, Y^0 earnings if not.

What is the effect of incumbency on electoral outcomes? i.e. do electors want change or continuity? In other words, what is the probability of a Democrat winning a House election given that a Democrat won the previous election?

Sharp regression discontinuity : $e_i(x) = 1x \geq \bar{x}$. Fuzzy one : $\lim_{x \to \bar{x}^-} e_i(x) \neq \lim_{x \to \bar{x}^+} e_i(x)$.

9.8.1 Treatment effect

Treatment effect locally at $\bar{x}$:

$$\delta_{SRD} = \lim_{x \to \bar{x}^+} E[Y^*|X = x] - \lim_{x \to \bar{x}^-} E[Y^*|X = x] \tag{501}$$

This is local treatment effect at $\bar{x}$.

9.8.2 Locally linear regression

$D_i = 1_{X_i \geq \bar{x}}$ and :

$$Y_i^1 = \alpha^1 + \beta^1 X_i + \varepsilon_i \tag{502}$$
$$Y_i^0 = \alpha^0 + \beta^0 X_i + \varepsilon_i \tag{503}$$
$$\tag{504}$$

Take kernel :

$$K(u) = \exp(-u^2/2)1_{u\geq 0} \tag{505}$$
$$\tag{506}$$

and :

$$\min_{\alpha,\beta} \sum_i K(\frac{X_i - \bar{x}}{h})(Y_i - \alpha - \beta X_i)^2 \tag{507}$$

Same for $\hat{\alpha}^0, \hat{\beta}^0$ and we have $\delta_{SRD} = \hat{\alpha}^1 - \hat{\alpha}^0$.

9.8.3 Fuzzy regression discontinuity

$$\delta_{FRD} = \frac{\lim_{x\to\bar{x}^+} E[Y^*|X=x] - \lim_{x\to\bar{x}^-} E[Y^*|X=x]}{\lim_{x\to\bar{x}^+} E[D|X=x] - \lim_{x\to\bar{x}^-} E[D|X=x]} \tag{508}$$

With unconfoundedness :

$$E[Y^d|D=1,X=x] = E[Y^d|D=0,X=x] = E[Y^d|X=x] \tag{509}$$
$$E[Y^1|X=x] - E[Y^0|X=x] = E[Y^*|D=1,X=x] - E[Y^*|D=0,X=x] \tag{510}$$

Unconfoundedness implies that similar units will have similar treatments. As for *SRD*, you do the same regression :

$$\min_{\alpha,\beta} \sum_i K(\frac{X_i - \bar{x}}{h})(Y_i - \alpha - \beta X_i)^2 \tag{511}$$
$$\min_{\alpha,\beta} \sum_i K(\frac{X_i - \bar{x}}{h})(D_i - \alpha - \beta X_i)^2 \tag{512}$$

9.9 Difference-in-differences

Idea : Heterogeneity is captured by a time-invariant additive factor. DID estimator :

$$DID = E[Y^1(t_1) - Y^1(t_0)|D=1] - E[Y^0(t_1) - Y^0(t_0)|D=0] \tag{513}$$

$$Y_i^d(t) = \alpha(t) + \rho(t)d + \varepsilon_i(t) \tag{514}$$
$$\alpha(t) = \alpha + \gamma t. \tag{515}$$
$$\rho(t) = \rho + \beta(t)t \tag{516}$$
$$E[\varepsilon_i(t)] = 0, cov(\varepsilon_i(t), D_i) = 0 \tag{517}$$

$$Y_i^d(t_1 = 1) - Y_i^d(t_0 = 0) = \gamma + \beta d + \varepsilon_i(t_1) - \varepsilon_i(t_0) \tag{518}$$

ATE estimated by β.

9.10 Linear Regression

9.10.1 Instrumental Variables

Causal effect of schooling on earnings. Y_i^* earnings, D_i schooling and A_i ability. If we have all 3 :

$$Y_i^* = \delta D_i + \gamma A_i + \eta_i \tag{519}$$
$$E[D_i \eta_i] = 0 \tag{520}$$
$$E[A_i \eta_i] = 0 \tag{521}$$

If ability not observed :

$$Y_i^* = \delta D_i + \varepsilon_i \tag{522}$$
$$\varepsilon_i = \gamma A_i + \eta_i \tag{523}$$
$$E[D_i \varepsilon_i] = \gamma E[D_i A_i] \tag{524}$$

Non-zero if ability and schooling are corelated.

If D_i and ε_i independent, one can estimate δ by OLS. The aim of IV : isolate the part of D_i not correlated with ε_i.

9.10.2 Two stage least-square

Interest :

$$Y_i^* = \delta D_i + \varepsilon_i \tag{525}$$

First Model how D_i depend on Z_i

$$D_i = \gamma Z_i + u_i \tag{526}$$
$$\text{Predictor : } \hat{D}_i = \hat{\gamma} \hat{Z}_i \tag{527}$$

Second stage regression :

$$Y_i^* = \delta \hat{D}_i + \varepsilon_i \tag{528}$$
$$= \delta(\gamma Z_i + u_i)_{\varepsilon i} \tag{529}$$
$$= \delta \gamma Z_i + (\delta u_i + \varepsilon_i) \tag{530}$$
$$\delta_{IV} = \frac{cov(Y_i^*, Z_i)}{cov(D_i, Z_i)} \tag{531}$$

Because :

$$cov(Z_i, \varepsilon_i) = 0 \tag{532}$$
$$cov(Y_i^* - \delta D_i, Z_i) = 0 \tag{533}$$
$$\delta = \frac{cov(Y_i^*, Z_i}{cov(D_i, Z_i)}) \tag{534}$$

Generally :

$$\delta_{IV} = \frac{cov(Y_i^*, Z_i}{cov(D_i, Z_i)}) - \frac{cov(\varepsilon_i, Z_i}{cov(D_i, Z_i)}) \tag{535}$$

If $cov(D_i, Z_i)$ small, instrument is said weaker, because bias term will be bigger.

9.11 Local Average Treatment Effect (LATE)

9.11.1 Framework

Effect of Catholic schools on academic performance. Assume that the outcome is a test score, and that D is an indicator for attending Catholic high school, and Z is an indicator for a student to be catholic.

$D_i = 1$ if school is catholic. $Z_i = 1$ if student is catholic. Potential treatments D_i^0 and D_i^1. We observe $D_i^{Z_i} = D_i^*$. Outcome $Y_i^* = Y_i^{D_i} = Y_i^{D_i^{Z_i}}$.

- $D^0 = 0, D^1 = 0$ never taker.
- $D^0 = 0, D^1 = 1$ complier.
- $D^0 = 1, D^1 = 0$ defier.
- $D^0 = 1, D^1 = 1$ alway taker.

Assume :

$$D_i^z = 1_{\pi_0 + \pi_1 z + \pi_2' X + \eta_i \geq 0} \tag{536}$$

$$\pi_1 > 0 \tag{537}$$

$$Y_i^d = \beta_0 + \beta_1 d + \beta_2' X + \varepsilon_i \tag{538}$$

And joint distribution (η_i, ε_i) known.

- Always taker : $-\pi_0 - \pi_2' X \leq \eta_i$.
- etc.

Assumptions :

- Independence : $Z_i \perp (Y_i^0, Y_i^1, D_i^0, D_i^1)$.
- Consider potential outcomes in all cases, $Z_i \perp (Y_i^{0,0}, Y_i^{1,0}, Y_i^{0,1}, Y_i^{1,1}, D_i^0, D_i^1)$.
- Exclusion restriction : $Y_i^{0,d} = Y_i^{1,d}, \forall d \in \{0,1\}, \forall i$. (In our case, religion impact the outcome only through school).
- Monotonicity : There are no defiers : $D_i^1 \geq D_i^0$.

Implication :

- $Z = 0, D = 0$ complier or never-taker.
- $Z = 0, D = 1$ always taker.
- $Z = 1, D = 0$ never-taker.
- $Z = 1, D = 1$ complier or always-taker.

9.11.2 Probability distribution compliance type

- $P_A = P(\text{Always taker}) = E[D_i^* | Z_i = 0]$.
  Because of monotonicity : $P_A = P(D_i^0 = 1, D_i^1 = 1) = P(D_i^0 = 1)$.
  Because of independence : $P(D_i^0 = 1) = P(D_i^0 = 1 | Z_i = 0) = E[D_i^* | Z_i = 0]$.
- $P_N = P(\text{Never taker}) = 1 - E[D_i^* | Z_i = 1]$.
  Because of monotonicity : $P_N = P(D_i^0 = 0, D_i^1 = 0) = P(D_i^1 = 0)$
  Because of independence : $P(D_i^1 = 0) = P(D_i^1 = 0 | Z_i = 1) = E[1 - D_i^* | Z_i = 1]$.
- $P_C = P(\text{Complier}) = E[D_i^* | Z_i = 1] - E[D_i^* | Z_i = 0] = 1 - P_A - P_N$ As there is no defiers.

Then :

$$E[Y_i^*|Z_i = 0, D_i^* = 0] = \frac{P_N E[Y_i^0|N]}{P_N + P_C} + \frac{P_C E[Y_i^0|C]}{P_N + P_C} \tag{539}$$

$$P(Y_i^0 = 1|Z_i = 0, D_i^0 = 0) = P(Y_i^0 = 1|D_i^0 = 0) \tag{540}$$

$$= \frac{P(Y_i^0 = 1, N) + P(Y_i^0 = 1, C)}{P_N + P_C} \tag{541}$$

$$P_N = P(D_i^0 = 0, D_i^1 = 0) \tag{542}$$

$$P_C = P(D_i^0 = 0, D_i^1 = 1) \tag{543}$$

Similarly :

$$E[Y_i^*|Z_i = 1, D_i^* = 1] = \frac{P_C E[Y_i^1|C]}{P_A + P_C} + \frac{P_A E[Y_i^1|A]}{P_A + P_C} \tag{544}$$

$$E[Y_i^*|Z_i = 0, D_i^* = 1] = E[Y_i^1|A] \tag{545}$$

$$E[Y_i^*|Z_i = 0, D_i^* = 0] = E[Y_i^1|N] \tag{546}$$

Thus :

$$E[Y_i^1|C] = E[Y_i^*|Z_i = 1, D_i^* = 1] + \frac{P_A}{P_C}(E[Y_i^*|Z_i = 1, D_i^* = 1] - E[Y_i^*|Z_i = 0, D_i^* = 1]) \tag{547}$$

$$E[Y_i^0|C] = E[Y_i^*|Z_i = 0, D_i^* = 0] + \frac{P_A}{P_C}(E[Y_i^*|Z_i = 0, D_i^* = 0] - E[Y_i^*|Z_i = 1, D_i^* = 0]) \tag{548}$$

And :

$$E[Y_i^*|Z_i = 1] = E[Y_i^1|C]P_C + E[Y_i^1|N]P_N + E[Y_i^0|A]P_A \tag{549}$$

$$E[Y_i^*|Z_i = 0] = E[Y_i^0|C]P_C + E[Y_i^1|N]P_N + E[Y_i^0|A]P_A \tag{550}$$

$$E[Y_i^*|Z_i = 1] - E[Y_i^*|Z_i = 0] = (E[Y_i^1|C] - E[Y_i^0|C])P_C \tag{551}$$

$$P_C = E[D_i^*|Z_i = 1] - E[D_i^*|Z_i = 0] \tag{552}$$

Thus :

$$\beta_{LATE} = E[Y_i^1|C] - E[Y_i^0|C] \tag{553}$$

$$= \frac{E[Y_i^*|Z_i = 1] - E[Y_i^*|Z_i = 0]}{E[D_i^*|Z_i = 1] - E[D_i^*|Z_i = 0]} \tag{554}$$

9.12 Control functions and panel data models

In control functions : Estimate variation of the treatment not due to the instrument.

Panel Data : we have several time observations about each unit, and individual effects, which affects the outcome in a time-independent way.

9.12.1 Control Functions

Recall :

$$y_i^* = d_i y_i^1 + (1 - d_i) y_i^0 \tag{555}$$

$$Y_i^* = \alpha + \delta D_i + \varepsilon_i \tag{556}$$

$$D_i = 1_{\gamma Z_i + u_i \geq 0} \tag{557}$$

Previously $D_i = \gamma Z_i + u_i$ continuous. And as before OLS of Y_i^* on D_i will not give δ as D_i might be correlated with ε_i.

Assumptions

- (A1) : $\varepsilon_i \perp (D_i, Z_i) | u_i$. It implies $epsilon_i \perp D_i | u_i$,
- (A2) : $\gamma_k > 0 \; \forall k$ and u has full support. It implies $e(z) = Pr(D_i = 1 | Z_i = z)$ increasing with respect to each component of x. Reminiscent of the monotonicity of LATE.
- (A3) :

$$(\varepsilon_i, u_i) \sim \mathcal{N}(0, \begin{bmatrix} \sigma_\varepsilon^2 & \rho\sigma_\varepsilon \\ \rho\sigma_\varepsilon & 1 \end{bmatrix}) \tag{558}$$

9.12.2 Heckman Corrections

Inverse Mills ratio formula :

$$g \sim \mathcal{N}(0,1) \tag{559}$$

$$E[g | g \geq a] = \frac{\phi(a)}{1 - \Phi(a)} = \frac{\phi(a)}{\Phi(-a)} \tag{560}$$

$$E[g | g \leq a] = -\frac{\phi(a)}{\Phi(a)} = \frac{\phi(a)}{1 - \Phi(-a)} \tag{561}$$

With ϕ the pdf of a Gaussian and Φ the CDF of a Gaussian.

Heckman Corrections

$$\varepsilon_i | D_i = 1, Z_i = z \sim \mathcal{N}(\rho\sigma_\varepsilon \frac{\phi(\gamma z)}{\Phi(\gamma z)}, \sigma_\varepsilon^2(1 - \rho^2)) \tag{562}$$

$$\varepsilon_i | D_i = 0, Z_i = z \sim \mathcal{N}(-\rho\sigma_\varepsilon \frac{\phi(\gamma z)}{1 - \Phi(\gamma z)}, \sigma_\varepsilon^2(1 - \rho^2)) \tag{563}$$

Indeed :

$$\varepsilon_i = \rho\sigma_\varepsilon u_i + \eta_i \tag{564}$$

$$\sigma_\eta^2 = \sigma_\varepsilon^2(1 - \rho^2) \tag{565}$$

$$D_i = 1 \text{ i.i.f} : u_i \geq -\gamma z \tag{566}$$

$$\text{Inverse Mills} : E[\varepsilon_i | D_i = 1, Z_i = z] = \rho\sigma_\varepsilon \frac{\phi(\gamma z)}{\Phi(\gamma z)} \tag{567}$$

Result :

$$Y_i^* = \alpha + \delta D_i + \rho\sigma_\varepsilon (D_i \frac{\phi(\gamma Z_i)}{\Phi(\gamma Z_i)} + (1 - D_i) \frac{\phi(\gamma Z_i)}{1 - \Phi(\gamma Z_i)}) + \eta_i \tag{568}$$

Can be estimated by non-linear least-square. Note that we can factor everything up and get :

$$Y_i^* = \alpha + \rho\sigma_\varepsilon \frac{\phi(\gamma Z_i)}{1 - \Phi(\gamma Z_i)} + (\delta + \rho\sigma_\varepsilon (\frac{\phi(\gamma Z_i)}{\Phi(\gamma Z_i)} - \frac{\phi(\gamma Z_i)}{1 - \Phi(\gamma Z_i)}))D_i + \eta_i \tag{569}$$

And we get the bias of running OLS of Y^* on D :

$$\rho\sigma_\varepsilon (\frac{\phi(\gamma Z_i)}{\Phi(\gamma Z_i)} - \frac{\phi(\gamma Z_i)}{1 - \Phi(\gamma Z_i)}) \tag{570}$$

9.12.3 Panel Data Models

In panel data we observe units over time and we assume there is an individual effect, which does not vary over time. The other regressors have sufficient time-variation to that this effect can be estimated.

- balanced panel: same periods observed for all units.
- unbalanced panel: potentially different periods observed for different units.
- fixed effect: the correlation between the individual effects and the other regressors is unknown.
- random effect: the individual effects are assumed to be independent from the other regressors.

9.12.4 Framework

$$y_{it} = x_{it}\beta + c_i + \varepsilon_{it} \tag{571}$$
$$\forall i, s, tE[x_{is}\varepsilon_{it}] = 0 \tag{572}$$
$$\forall i, sE[c_i\varepsilon_{it}] = 0 \tag{573}$$
$$\text{Assume :} \tag{574}$$
$$E[\varepsilon_{it}\varepsilon_{js}] = 0, E[c_i c_j] = 0 \forall t \neq s \tag{575}$$

It assumes that the outcome of i at time t is only affected by x_{it} and c does not depend on t.

9.12.5 Random Effect

Can we regress y on x ? It implies :

$$y_{it} = x_{it}\beta +_{it} \tag{576}$$
$$\varepsilon it = c_i + \varepsilon_{it} \tag{577}$$

This is Okay as long as $E[x_{it}\varepsilon is] = 0 i, s, t$ which implies :

$$E[x_{it}c_i] = 0 \tag{578}$$

In this case, the individual effect is orthogonal to the other regressors it is the Random Effect Model. In a Random Effect setting, one can do OLS but take into account :

$$cov(\varepsilon_{it}, \varepsilon_{is}) = var(c_i) \forall s \neq t \tag{579}$$

Errors are not i.i.d. !

Correlation Structure in the Random Effect model : With $\varepsilon it = c_i + \varepsilon_{it}$ we have $\Omega_i = (cor(\varepsilon it, \varepsilon is))_{t,s}$ matrix with : $\Omega_{Is,s} = \sigma_\varepsilon^2 + \sigma_c^2$ and $\Omega_I s, t = \sigma_c^2$.

Estimator of the Random Effect Model :

$$\tilde{y} = \Omega^{-1/2} y \tag{580}$$
$$\tilde{X} = \Omega^{-1/2} X \tag{581}$$
$$\tilde{\varepsilon} = \Omega^{-1/2} \varepsilon \tag{582}$$

OLS Gives us :

$$\hat{\beta}_{RE} = (\tilde{X}^T)^{-1}\tilde{X}^T\tilde{y} = (X^T\gamma^{-1}X)^{-1}X^T\Gamma^{-1}y \tag{583}$$

9.12.6 Fixed Effect

In fixed effect, we allow for possible correlation between x_{it} and c_i. We have :

$$y_{it} = x_{it}\beta + c_i + \varepsilon_{it} \tag{584}$$
$$\forall i, s, t E[x_{is}\varepsilon_{it}] = 0 \tag{585}$$
$$\forall i, s E[c_i\varepsilon_{it}] = 0 \tag{586}$$
$$\text{Assume :} \tag{587}$$
$$E[\varepsilon_{it}\varepsilon_{js}] = 0, E[c_i c_j] = 0 \forall t \neq s \tag{588}$$
$$E[X_{is}c_i] \neq 0 \tag{589}$$

Objective is to get rid of c_i.

First difference : Write :

$$y_{it} - y_{it-1} = (x_{it} - x_{it-1})\beta + \varepsilon_{it} - \varepsilon_{it-1} \tag{590}$$

The first difference estimator is obtained by OLS this : $\hat{\beta}_{FE-FD}$. However this estimator is not sufficient because errors $\varepsilon_{it} - \varepsilon_{it-1}$ are serially uncorrelated.

Within Estimator : Similar to previously, but instead of substracting the previous period, substract the time average :

$$\bar{y_i} = \frac{1}{T}\sum_{t=1}^{T} y_{it} \tag{591}$$
$$\bar{x_i} = \frac{1}{T}\sum_{t=1}^{T} x_{it} \tag{592}$$
$$\bar{\varepsilon_i} = \frac{1}{T}\sum_{t=1}^{T} \varepsilon_{it} \tag{593}$$
$$\bar{y_i} = \beta\bar{x_i} + \bar{\varepsilon}_i + c_i \tag{594}$$

And :

$$\delta y_{it} = y_{it} - \bar{y_i} \tag{595}$$
$$\delta y_{it} = \delta x_{it}\beta + \delta\varepsilon_{it} \tag{596}$$

OLS on this we get : $\hat{\beta}_{FE-W}$.

This estimator is unbiased and consistent, but not efficient because $cov(\delta\varepsilon_{it}, \delta\varepsilon_{is}) = \frac{1}{T}\sigma_\varepsilon^2 \neq 0$.

Dummy Variables :

$$g_{it}^j = 1_{i=j} \tag{597}$$
$$y_{it} = x_{it}\beta + gc + \varepsilon_{it} \tag{598}$$

This estimator is unbiased, consistent and efficient. The fixed effect is directly available using the OLS regressor of y on (x, g). The large dimensionality of the problem n big can be a problem.

9.13 Distributional Treatment Effect

We have focused on treatment effect, but we haven't answered the question : Does the treatment favor the worse off, the better off ? What is the effect on the worse / median of the distribution ?

How to regress quantiles on covariates.

9.13.1 Quantiles

Random variable Y c.d.f. F_Y continuous stricly increasing.

$$\forall \tau] in [0,1] \exists ! \bar{y}, F_Y(\bar{y}) = \tau \bar{y} = F_Y^{-1}(\tau) = Q_Y(\tau) \tag{599}$$

Q_Y the quantile function.

$$U = F_Y(Y) \sim \mathcal{U}(0,1) \tag{600}$$
$$Q_Y(U) \text{ has cdf } F_Y \tag{601}$$

9.13.2 Loss minimization

$$\rho_\tau(u) = \tau u^+ + (1-\tau)u^- \min_y \mathbb{E}[\rho_\tau(Y-y)] \tag{602}$$
$$\text{First order condition : } (1-\tau)F_Y(\bar{y}) - \tau(1-F_Y(\bar{y})) = 0 \tag{603}$$

The solution of the min. problem is the τ-th quantile. The median minimizes the absolute loss error : $\rho_{1/2}(u) = 1/2|u|$.

More generally, quantiles minimizes an expected asymmetric absolute loss error. If an underestimate is 3 times more costly than an over-estimate we select the 0.75 quantile.

Empirical quantile :

$$\hat{q}_y(\tau) = \underset{y}{\text{argmin}} \frac{1}{n} \sum_i \rho_\tau(Y_i - y) \tag{604}$$

9.13.3 Conditionnal quantile function

$F_{Y|X}(\bar{y}|x) = \tau$.

The conditionnal quantile function $Q_{Y|X}(.|x)$ is solution to :

$$Q_{Y|X}(.|x) = \underset{y}{\text{argmin}} \mathcal{E}[\rho_\tau(Y-\hat{y})|X=x] \tag{605}$$

9.13.4 Quantile regression

Suppose the true quantile function is linear in x.

$$Q_{Y|X}()\tau|x) = x^T\beta(\tau) \tag{606}$$
$$\text{Then : } \beta(\tau) = \underset{y}{\text{argmin}} \mathcal{E}[\rho_\tau(Y - x^T b)|X=x] \tag{607}$$
$$\text{Expectation over X : } \beta(\tau) = \underset{y}{\text{argmin}} \mathcal{E}[\rho_\tau(Y - X^T b)] \tag{608}$$

We can estime $\beta(\tau)$ by quantile regression using :

$$\hat{\beta}^{QR}(\tau) = \underset{b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \rho_\tau(Y_i - X_i^T b) \tag{609}$$

$$\hat{\beta}^{OLS}(\tau) = \underset{b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - X_i^T b)^2 \tag{610}$$

Standard Model :

$$Y_i = \beta_0 + X_i\beta_1 + \varepsilon_i \tag{611}$$
$$Q_{Y|X}(\tau|X) = \beta_0 + x_i\beta_1 + Q_\varepsilon(\tau) \tag{612}$$
$$\beta_0(\tau) = \beta_0 + Q_\varepsilon(\tau) \tag{613}$$
$$\beta_1(\tau) = \beta_1 \tag{614}$$

All the quantile information is captured by the constant term. This is called the location-shift model. The shape of the conditionnal distribution remains the same.

Suppose :

$$\varepsilon_i = (\delta_0 + \delta_1 x_i)u_i \tag{615}$$
$$y_i \text{ i.i.d. independent of } x_i \tag{616}$$
$$Q_{Y|X}(\tau|x) = \beta_0 + x_i\beta_1 + (\delta_0 + \delta_1 u_i)Q_u(\tau) \tag{617}$$
$$\beta_0(\tau) = \beta_0 + \delta_0 Q_u(\tau) \tag{618}$$
$$\beta_1(\tau) = \beta_1 + \delta_1 Q_u(\tau) \tag{619}$$

This is called location-scaled model. The covariates also affect the shape of the distribution. If $\delta_1 > 0$ then x_i increase $\beta_1(\tau)$ which means it increase heterogeneity.

9.13.5 Quantile Treatment

The meaning of "median treatment effect" is ambiguous.

The median of the treatment effect is $Q_{Y_1Y_0}(1/2)$. However, it has no reason to coincide with the treatment effect at the median; i.e. the treatment effect on the median individual, i.e. the treatment on the units such that $Y_i^0 = Q_Y^0(1/2)$. We may be interested in the latter: the distribution of the treatment effect conditional on being in rank in the non-treated population, i.e. in the distribution $Y_1|Y_0 = Q_{Y_0}(\tau)$, for instance, we may be interested in $E[Y_1|Y_0 = Q_{Y_0}(\tau)]$ (average treatment effect on the τ-quantile of the untreated).

However, even under the perfect randomization assumption, such quantities are impossible to identify. The reason is that they require the knowledge of the joint distribution of the potential outcomes – unlike the average treatment effect. For instance, education can reshuffle earnings ...

9.13.6 Rank Similarly Assumption

If Y^d continuously distributed, one can define the rank of observation Y_i^d as

$$U_i^d = F_1(Y_i^d) \tag{620}$$
$$Y_i^d = Q_1(U_i^d) \tag{621}$$

Under the rank similarly assumption, $U_i^1 = U_i^2$. Treatment do not have an effect on ranking.

9.13.7 Quantile Treatment Effect

$$Y_1|Y_0 = Q_1(\tau) \text{ coincides with : } Y_1|U = \tau \tag{622}$$
$$\delta_{QTE}(\tau) = Q_1(\tau) - Q_0(\tau) \tag{623}$$
$$\delta_{ATE} = \int_{\tau=0}^{1} \delta_{QTE}(\tau) d\tau \tag{624}$$

Indeed, $Q_d(u)$ as the same distribution as Y^d.

9.13.8 QTE under Perfect Randomization

Asumme perfect randomization and rank similarly. Under this assumption Q_0, Q_1 are computed given Y^* conditionned on treatment or not.

$$Q_0(\tau) = \underset{q}{\text{argmin}} \mathcal{E}[\rho_\tau(Y^* - q)|D = 0] \tag{625}$$
$$Q_1(\tau) = \underset{q}{\text{argmin}} \mathcal{E}[\rho_\tau(Y^* - q)|D = 1] \tag{626}$$

9.13.9 QTE under Unconfoundedness

Assume :

$$(Y^0, Y^1) \perp D|X \tag{627}$$
$$\text{Rank Similarly : } Y^d = Q_{Y^d|X}(U|X) \tag{628}$$
$$\text{Independence assumption : } X \perp U \tag{629}$$

$$\delta_{QTE}(\tau|x) = Q_{Y^1|X}(\tau) - Q_{Y^0|X}(\tau) \tag{630}$$
$$Q_0(\tau) = \underset{q}{\text{argmin}} \mathcal{E}[\rho_\tau(Y^* - q)|X = x, D = 0] \tag{631}$$
$$Q_1(\tau) = \underset{q}{\text{argmin}} \mathcal{E}[\rho_\tau(Y^* - q)|X = x, D = 1] \tag{632}$$

Under independence assumption :

$$\delta_{QTE}(\tau) = E[\delta_{QTE}(\tau|x)] \tag{633}$$

9.13.10 Instrumental variables quantile regression

Based on : Rank similarly assumption and Valid instrument ($Z \perp U$). If $U \perp D$ back on QTE framework (which could happen if $D = \phi(Z)$).

One can estimate the quantile $q = Q_0(\tau)$ of Y_0 , as well as the quantile treatment $\delta = \delta(\tau)$ by the set of conditions :

$$E[\tau - 1_{Y^* \leq \delta D + q}] = 0 \tag{634}$$
$$E[(\tau - 1_{Y^* \leq \delta D + q})Z] = 0 \tag{635}$$

Because :

$$1_{Y^* \leq \delta D + q} = 1_{U \leq \tau} \tag{636}$$

And $U \sim \mathcal{U}([0, 1])$ So First is right, and independence implies second.

9.14 Structural Models

Structural models. It consists in writing an equilibrium model which determines the treatment. Several prototypical example will be provided: demand estimation, dynamic choice, and equilibrium models.

9.14.1 Multinomial Choice

Binomial logistic regression :

$$\text{If} : D = 1_{u_0 \leq u_1 + \varepsilon} \tag{637}$$

$$Pr(D = 1) = \frac{\exp(u_1)}{\exp(u_1) + \exp(u_0)} \tag{638}$$

More generally, one may consider :

$$D = \text{argmax}_j(u_j + \varepsilon_j) \tag{639}$$

$$\text{Look for} : Pr(D = j) \tag{640}$$

$$j \in \mathcal{J} = \{0, 1, ..., m\} \tag{641}$$

Normalization is needed, usually $u_0 = 0$.

The agent's indirect utility is the utility associated to its most preferred choice, i.e.

$$\max_{j \in \mathcal{J}}(u_j + \varepsilon_j, \varepsilon_0) \tag{642}$$

$$\text{Expectation} : G(u) = E[\max_{j \in \mathcal{J}}(u_j + \varepsilon_j, \varepsilon_0)] \tag{643}$$

Let us compute p_{j^*} the probability that the agent choose j^*. It's equal to the probability that $u_{j^*} + \varepsilon_{j^*} \geq u_j + \varepsilon_j$. Which is the probability of $\varepsilon_j - \varepsilon_{j^*} \leq u_{j^*} - u_j$.

$$p_{j^*} = F_{\varepsilon_j - \varepsilon_{j^*}}(u_j^* - u_j) \tag{644}$$

We have :

$$\frac{\partial G(u)}{\partial u_{j^*}} = E[1_{u_{j^*} + \varepsilon_{j^*} \geq u_j + \varepsilon_j}] \tag{645}$$

$$= p_{j^*} \tag{646}$$

The Logit framework next introduce an instance where $G(u)$ can be obtained in closed form.

9.14.2 Logit Framework

We assume ε_i independent from one another and C.D.F. :

$$\exp(-\exp(-\varepsilon)) \tag{647}$$

It's the Gumbel distribution or extreme value I distribution. Mean $\gamma = .5772$. It is natural to use this distribution because $\varepsilon = \varepsilon_i - \varepsilon_j$ follows a logistic distribution !

Direct utility : Consider $Z = \max_j\{u_j + \varepsilon_j\}$.

$$F_Z(z) = Pr(\max_{j\in\{0,...,m\}}\{u_j + \varepsilon_j\} \leq z) \tag{648}$$
$$= Pr(\forall j, \varepsilon_j \leq z - u_j) \tag{649}$$
$$= F_\varepsilon(z - u_0, ..., z - u_m) \tag{650}$$
$$= \exp(-e^{-z}\exp(u_0 + ... + u_m)) \tag{651}$$
$$= \exp(-e^{-z-\log(e^{u_0}+...+e^{u_m})}) \tag{652}$$

Hence, $Z - \log(e^{u_0} + ... + e^{u_m})$ has a Gumbel Distribution.

Therefore :

$$G(u) = E[\max_{j\in\{0,...,m\}}\{u_j + \varepsilon_j\}] \tag{653}$$
$$= \log(e^{u_0} + ... + e^{u_m}) + \gamma \tag{654}$$

Hence :

$$p_j = \frac{e^{u_j}}{1 + e^{u_1} + ... + e^{u_m}} \tag{655}$$

We have also the 'log-odds-ratio' :

$$u_j = \log(\frac{p_j}{p_0}) \tag{656}$$
$$u_j - u_i = \log(\frac{p_j}{p_i}) \tag{657}$$

The Logit framework is restrictive : need independence of ε_j and is subject to a number of paradox. A more flexible model can be used : Generalized Extreme Value theory.

We would use $p_j(x)$ probability of an agent of type x to choose j.

9.15 Demand Estimation

9.15.1 Price Endogeneity

In demand estimation problems, the analyst has access to cross-sectional data. A number of markets are observed, with for each market, the price of a brand and the market share of that brand.

A possible explanation may be price endogeneity: an unobserved variable (quality) may have causal impact both on prices and demand.

9.15.2 Simple model

If i is a household, j is a brand, assume that the utility of household i for brand j in a given market is expressed by :

$$u_j + \varepsilon_{ij} \tag{658}$$
$$u_j = X_j\beta - \alpha\pi_j + \zeta_j \tag{659}$$
$$X_j \text{ Observed quality of good } j \tag{660}$$
$$\pi_j \text{ Price of good } j \tag{661}$$
$$\zeta_j \text{ Unobserved quality of good} \tag{662}$$

We can use log-odds ratio to recover u_j from market shares : $\hat{u}_j = \log\frac{p_j}{p_0}$. We could then regress u_j on (X_j, π_j) but it will fail if price of good is correlated with unobserved quality of good.

9.15.3 Berry's IV approach

If we have a variable z_J independent from ζ_j then we could estimate α, β using a IV regression :

$$\frac{1}{J}\sum_{j=1}^{J}(\hat{u}_j - \beta X_j + \alpha\pi_j)Z_j = 0 \tag{663}$$

In practice those Z_j are cost-shifters, independent from the brand quality but affect the price of various goods in the market. E.g. intensity of competition in a market (presence of close competitor).

9.16 Dynamic Choice

In a number of situations, the utilities associated with taking a certain action is endogenous, because it reflects the future benefits that this action leads to expect.

9.16.1 Rust's Model

Assume agents choose actions $j \in \mathcal{J}$ from a finite space $\mathcal{J} = \{0, 1, ..., m\}$. At each period, agents get utility flow from choosing j given by $\bar{u}_j + \varepsilon_j$. ε_j denotes the utility shock associated to action j, which differs accross agents and supposed i.i.d. Gumble distributed.

Assume the model stationnary, independent of time.

In Rust's setting, $j = 1$ if maintenance, 0 otherwise is whether or not to perform repair on a bus, X_j mileage since last maintenance. $\bar{u}_1(x)$ is the average profit if no maintenance and mileage x, $\bar{u}_0(x)$ if maintenance.

In a one-period context, agents would simply maximize $\bar{u}_j(x) + \varepsilon_j$ over j as in static discrete choice framework seen.

However, action chosen at time t, has an impact on state x_t at the next period. Agents take this into account and optimize instead :

$$\bar{u}_j(x) + \varepsilon_j + \beta E[\bar{V}(x', \varepsilon')|x, j] \tag{664}$$

over j with $\beta E[\bar{V}(x', \varepsilon')|x, j]$ is the expected discounted payoff from later periods conditional on being at state x and choosing action j at the current period. $\bar{V}$ is defined recursively. Prime denote future periodes.

9.16.2 Conditionnal Independence Assumption

We make the following assumptions :

$$Pr(x', \varepsilon'|j, x, \varepsilon) = Pr(\varepsilon'|x', j, x, \varepsilon)Pr(x'|j, x, \varepsilon) \tag{665}$$
$$= Pr(\varepsilon'|x')Pr(x'|j, x) \tag{666}$$

At each periods, agents do :

$$j = \underset{j\in\mathcal{J}}{\text{argmax}}\bar{u}_j(x) + \varepsilon_j + \beta E[\bar{V}(x', \varepsilon')|x, j] \tag{667}$$
$$\bar{V}(x, \varepsilon) = \max_j \bar{u}_j(x) + \varepsilon_j + \beta E[\bar{V}(x', \varepsilon')|x, j] \tag{668}$$

Introducing the ex-ante or integrated value function :

$$V(x) = E[\bar{V}(x, \varepsilon)|x] \tag{669}$$

The choice-specific value functions therefore consists of two terms: the per-period utility flow and the discounted continuation payoff :

$$w_j(x) \equiv \bar{u}_j(x) + \beta E[\bar{V}(x')|x, j] \tag{670}$$

$$V(x) = \log \sum_{j \in \mathcal{J}} e^{w_j(x)} \tag{671}$$

We cannot impose normalization as $w_0(x) \equiv \bar{u}_0(x) + \beta E[\bar{V}(x')|x, j = 0]$ second term is non-zero. We just set $\bar{u}_0(x) = 0$. The log-odds ratio becomes :

$$\log \frac{p_j(x)}{p_0(x)} = w_j(x) - w_0(x) \tag{672}$$

We need to identify the w_0 term.

$$V(x) = \log \sum_j \frac{p_j(x)}{p_0(x)} + w_0(x) \tag{673}$$

$$= w_0(x) - \log p_0(x) \tag{674}$$

$$w_0(x) = \beta E[\bar{V}(x')|x, j = 0] \tag{675}$$

$$E[\beta(x') - V(x)|x, 0] = \log p_0(x) \tag{676}$$

Let's define :

$$W = (\log p_0(x))_{x \in \mathcal{X}} \tag{677}$$

$$V = (V(x))_{x \in \mathcal{X}} \tag{678}$$

$$\Pi = (Pr(x_{t+1} = j|x_t = i, j = 0))_{ij} \tag{679}$$

$$W = (\beta\Pi - I)V \tag{680}$$

For $\beta < 1$ the matrix is invertible, thus :

$$V = (\beta\Pi - I)^{-1}W \tag{681}$$

$V(x)$ is identified by data. All other quantities $w_j, \bar{u}_j$ are all combination of previous known quantities.

9.17 Equilibrium Models

In equilibrium models, one needs to model supply and demand on the basis of model primitives: producers' technology and consumers' utility. These are exogenous objects.

The market is assumed to clear (supply and demand adjust) by the means of endogenous prices.

Usually prices are observed by the analyst. Sometimes more: producers and consumers characteristics, who buys and sells what, etc.

One of the simplest models is the hedonic model, in which consumer consume one unit of each good, and producer produce one unit of the good, but the good are differentiated in terms of quality. Examples include: real estate, personal computers, automobile, etc.

9.17.1 Hedonic Regression

In hedonic equilibrium models, the producers have a technology $x \in \mathbb{R}$, distributed as $\mathcal{N}(\mu_x.\sigma_x^2)$. Each producer produces one unit of the good, which comes in different quality z. If $w(z)$ is the unit price of quality z, then the producer surplus is $w(z)c(x, z)$ where :

$$c(x,z) = \frac{az^2}{2} - zx \tag{682}$$

A consumer of type $y \in \mathbb{R}$ distributed with gaussian ... has valuation :

$$U(y,z) = zy - \frac{gz^2}{2} \tag{683}$$

Assume $a + g > 0$. Only $w(z)$ is observed, not x or y.

Consider the consumer profit maximization :

$$\max_z w(z) + zx - \frac{az^2}{2} \tag{684}$$

$$\tag{685}$$

by first order condition, one has : $w'(z) = az - x$.

However, a cannot be obtained by the regression of $w'(z)$ on z as z is not independent of x; indeed, the producers with the higher x select into producing higher z.

The consumer utility maximization : $\max_z zy - \frac{gz^2}{2} - w(z)$ has first order $w'(z) = y - gz$. $-g$ cannot be obtained by regression of $w'(z)$ on z.

9.17.2 Equilibrium wages

Guess $w(z) = w_0 z + w_1 z^2/z$. By F.O.C :

$$z^s = \frac{x + w_0}{a - w_1} \tag{686}$$

$$z^d = \frac{y - w_0}{g + w_1} \tag{687}$$

Equality of means and variance :

$$\frac{\mu_x + w_0}{a - w_1} = \frac{\mu_y - w_0}{g + w_1} \tag{688}$$

$$\sigma_y(a - w_1) = \sigma_x(g + w_1) \tag{689}$$

$$\text{So : } w_1 = \frac{a\sigma_y - g\sigma_y}{\sigma_x + \sigma_y} \tag{690}$$

$$w_0 = \frac{\sigma_x\mu_y - \sigma_y\mu_x}{\sigma_x + \sigma_y} \tag{691}$$

$$w'(z) = w_1 z \tag{692}$$

$$\text{Regressing : } w'(z) \text{ on } z \text{ yields : } \frac{a\sigma_y - g\sigma_y}{\sigma_x + \sigma_y} \tag{693}$$

9.18 Causal calculus and probabilistic graphical models, An introduction to causal graphs

A large part of this course has been about describing dependence and independence relationships between random variables. Unconfoundedness, Heckman's selection model :

$$Y^0 = \alpha + \varepsilon \tag{694}$$
$$Y^1 = Y^0 + \delta D \tag{695}$$
$$D = 1_{\gamma Z + u \geq 0} \tag{696}$$

ε, u might be correlated, and Z independent from the two before.

One sees that if we'd like to express dependence between random variables, we either have to write a specific model, or write a cumbersome factorization of the distributions. Thanks to Pearl's causal graphs, we can do this in a much more intuitive way.

9.18.1 Causal Graph

One can always write :

$$f(x_1, ..., x_n) = \prod_j f(x_j | x_1, ..., x_{j-1}) \tag{697}$$

For any pair x_i, x_j draw an arc from x_i to x_j if $f(x_j|x_1, ..., x_{j-1})$ actually depends on x_i. Without any restriction we get a triangular graph where $x_i \to x_j$ if $i < j$. It depends on the ordering of indicies, clearly some are better than others.

Vocabulary :

- The causal relationships are represented by a directed graph.
- Nodes are random variables.
- Directed edges: single-headed arrows. $X \to Y$ means loosely "r.v. X has a causal effect on r.v. Y ".
- Cycles : a directed path from a node to itself.
- Directed acyclic graphs : directed graph without a cycle.

There is at least one node without parents.

Two types of nodes: solid circle • indicates observed random variable, while hollow circle indicates latent (unobserved) random variable.

Shorthand notation: a curved dashed bidirected edges between two variables indicates that those variables have a common latent variable among their ancestors.

With three variables :

- Mediation / Chain : $a \to b \to c$.
$$f(a, b, c) = f(a) f(b|a) f(c|b) \tag{698}$$
- Mutual Dependence / fork : $a \leftarrow b \to c$.
$$f(a, b, c) = f(b) f(a|b) f(c|b) \tag{699}$$
- Mutual Causation / collider : $a \to b \leftarrow c$
$$f(a, b, c) = f(a) f(c) f(b|a, c) \tag{700}$$

9.18.2 Blocking

Two sets X and Y are blocked (or d-separated) by Z if and only if Z blocks every path between X and Y . Denoted $X \perp Y | Z$.

Z blocks a path from a to b if :

- There is $c \in Z$ such as : chain $a \to c \to b$ or fork $a \leftarrow c \to b$.
- A collider : $a \to c \leftarrow b$.

Look at example for example between We observe the value of a variable, and we ensure the value of a variable. Ensure the value of varible = removing the edges that goes to that variable.

9.19 Pearl's causal graphs

To each variable X we associate an error term e_X, unobserved and independent.

In Pearl's formal definition :

- All variables (other than error terms) are observed.
- each variable X is associated with an unique error term e_X , and the e_X are independent.
- for each variable X , there exists a $do(X = x)$ operation, which replaces X by the constant x and removes all the arcs pointing at X, leaving everything else unchanged.

9.19.1 Backdoor paths

A back-door path is a path between any causally ordered sequences of variables that begins with a directed edge that points to the first variable.

9.19.2 Collider pitfall

As we saw before, in case of a fork, conditioning on the parent variable allows us to identify the causal effect. What about the case of a collider? Conditioning on a collider creates dependence.

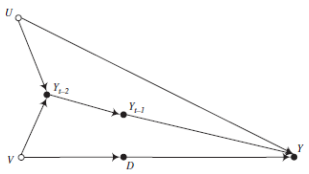
9.19.3 Backdoor criterion - Pearl's backdoor criterion

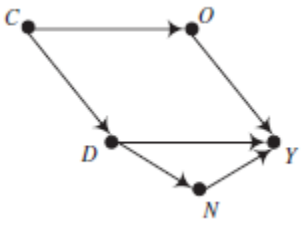
Goal=block paths that generate non-causal associations without blocking paths that generate causal effect.

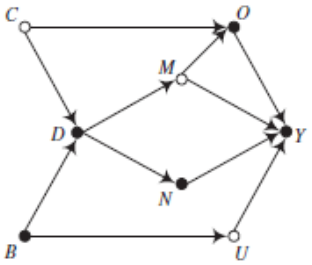
Back-door criterion: the causal effect is identified by conditioning on a set of observed variables Z if the following two conditions are met :

- All back-door paths between the causal variable and the outcome variable are blocked after conditioning on Z , which will will be the case if each back-door path:
  - Contains $A \rightarrow C \rightarrow B$, where C is in Z.
  - Contains a fork of mutual independence $A \leftarrow C \rightarrow B$, where C is in Z.
  - Contains an inverted fork of mutual causation $A \rightarrow C \leftarrow B$ where C and all its descendants are not in Z.
- No variables in Z are descendants of the causal variable that lie on (or descend from other variables that lie on) any of the directed path that begin at the causal variable and reach the outcome variable.

Conditioning on a set that satisfies the back-door criterion identifies the causal effect.

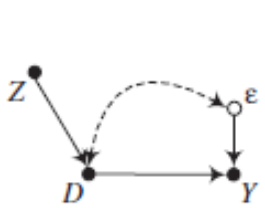




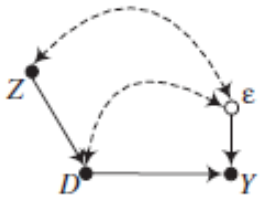


9.19.4 Self-selection bias

Consider the potential outcome model: $Y = DY_1 + (1D)Y_0$ which rewrites into $Y = Y_0 + \delta D$ with $\delta = Y^1 Y^0$. Heckman and Robb (1985, 1986,1989) assume $D = 1_{Z\phi+U\geq 0}$, where Z is a random vector of observable variables, ϕ is a vector of coefficient, and U is some unobserved factor. If $U \perp (Y, Z)$, then there is selection on the observables (=unconfoundedness); otherwise there is selection on the unobservables.



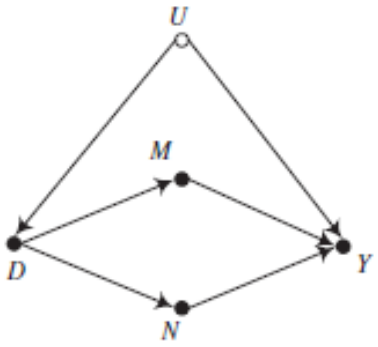
(a) Z is a valid instrumental variable for D



(b) Z is not a valid instrumental variable for D

9.19.5 Front-door criterion

blocked. Consider the directed graph below. The back-door criterion does not apply because U is unobservable, so the path $D \leftarrow U \rightarrow Y$cannot be both observed. Are the four causal effects $D \rightarrow M$, $D \rightarrow N$, $M \rightarrow Y$ However, the full effect of D on Y goes through M and N, which are and $N \rightarrow Y$ identified ?

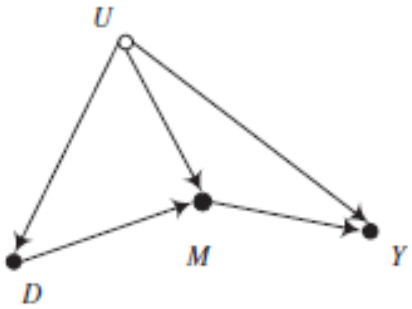


By the back-door criterion, the causal effects $D \rightarrow M$, $D \rightarrow N$ are identified (unconditionally). By the back-door criterion again, the causal effects $M \rightarrow Y$ and $N \rightarrow Y$ are identified (by conditioning on D).

If there is at least one unblocked back-door path connecting a causal variable D to an outcome variable Y, the causal effect is identified by a set X_i of observed variables if the following two conditions are met:

- Exhaustiveness: any directed path from D to Y contains at least one of the X_i 's.
- Isolation: there is no unblocked back connecting D to any of the X_i , and all back-door paths from any X_i to Y can be blocked by D.

For example, the following graph does not satisfy isolation:



Part IV
Third Semester

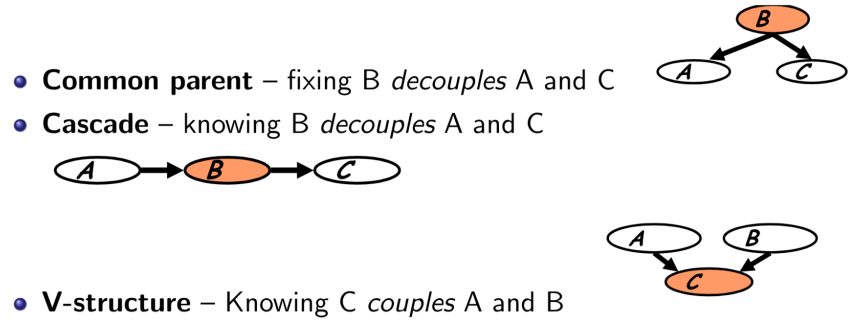
10 Statistical Natural Language Processing

11 Inference and Representation

11.1 Bayesian Networks

A Bayesian network is specified by a directed acyclic graph.

We can interpret Bayesian network as a generative process. For example, to generate an e-mail, we decide whether it is spam or not spam, by sampling $y \sim P(Y)$ and then sampling words with $P(x_i|Y)$.



- **Common parent** – fixing B *decouples* A and C
- **Cascade** – knowing B *decouples* A and C
- **V-structure** – Knowing C *couples* A and B
  - This important phenomona is called **explaining away** and is what makes Bayesian networks so powerful

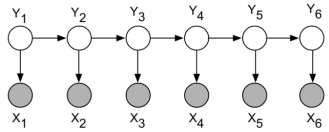
Figure 19: Bayesian Networks rules

- Let $I(G)$ be the set of all conditional independencies implied by the directed acyclic graph (DAG) G
- Let $I(p)$ denote the set of all conditional independencies that hold for the joint distribution p.
- A DAG G is an **I-map** (independence map) of a distribution p if $I(G) \subseteq I(p)$
  - A fully connected DAG G is an I-map for *any* distribution, since $I(G) = \emptyset \subseteq I(p)$ for all p
- G is a **minimal I-map** for p if the removal of even a single edge makes it not an I-map
  - A distribution may have several minimal I-maps
  - Each corresponds to a specific node-ordering
- G is a **perfect map** (P-map) for distribution p if $I(G) = I(p)$

Figure 20: Independence Map

Theorem : Not every distribution has a perfect map as a DAG. Example in class 2.

11.2 Hidden Markov Models



- Joint distribution factors as:

$$p(\mathbf{y}, \mathbf{x}) = p(y_1)p(x_1 \mid y_1) \prod_{t=2}^{T} p(y_t \mid y_{t-1})p(x_t \mid y_t)$$

- A **homogeneous** HMM uses the same parameters (β and α below) for each transition and emission distribution (**parameter sharing**):

$$p(\mathbf{y}, \mathbf{x}) = p(y_1)\alpha_{x_1,y_1} \prod_{t=2}^{T} \beta_{y_t,y_{t-1}}\alpha_{x_t,y_t}$$

Figure 21: Hidden Markov Models

11.3 Markov Random Fields

An alternative representation for joint distributions is as an undirected graphical model.

- Rather than CPDs, we specify (non-negative) **potential functions** over sets of variables associated with cliques C of the graph,

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(\mathbf{x}_c)$$

  Z is the **partition function** and normalizes the distribution:

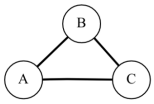
$$Z = \sum_{\hat{x}_1, \ldots, \hat{x}_n} \prod_{c \in C} \phi_c(\hat{\mathbf{x}}_c)$$

- Like CPD's, $\phi_c(\mathbf{x}_c)$ can be represented as a table, but it is *not normalized*

Figure 22: Undirected Graphs

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(\mathbf{x}_c), \qquad Z = \sum_{\hat{x}_1, \ldots, \hat{x}_n} \prod_{c \in C} \phi_c(\hat{\mathbf{x}}_c)$$

Simple example (potential function on each edge encourages the variables to take the same value):



$\phi_{A,B}(a,b) =$

A \ B	0	1
0	10	1
1	1	10

$\phi_{B,C}(b,c) =$

B \ C	0	1
0	10	1
1	1	10

$\phi_{A,C}(a,c) =$

A \ C	0	1
0	10	1
1	1	10

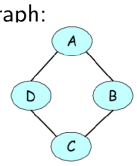
$$p(a,b,c) = \frac{1}{Z} \phi_{A,B}(a,b) \cdot \phi_{B,C}(b,c) \cdot \phi_{A,C}(a,c),$$

where

$$Z = \sum_{\hat{a},\hat{b},\hat{c} \in \{0,1\}^3} \phi_{A,B}(\hat{a},\hat{b}) \cdot \phi_{B,C}(\hat{b},\hat{c}) \cdot \phi_{A,C}(\hat{a},\hat{c}) = 2 \cdot 1000 + 6 \cdot 10 = 2060.$$

Figure 23: Undirected Graphs : Example 1

- We now have an **undirected** graph:



- The joint probability distribution is parameterized as

$$p(a,b,c,d) = \frac{1}{Z} \phi_{AB}(a,b) \phi_{BC}(b,c) \phi_{CD}(c,d) \phi_{AD}(a,d)\ \phi_A(a) \phi_B(b) \phi_C(c) \phi_D(d)$$

- **Pairwise potentials** enforce that no friend has the same hair color:

$$\phi_{AB}(a,b) = 0 \text{ if } a = b, \quad \text{and 1 otherwise}$$

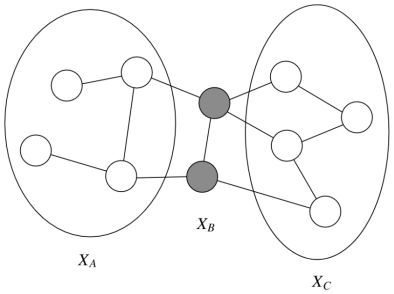
- **Single-node potentials** specify an affinity for a particular hair color, e.g.

$$\phi_D(\text{“red”}) = 0.6, \quad \phi_D(\text{“blue”}) = 0.3, \quad \phi_D(\text{“green”}) = 0.1$$

The normalization Z makes the potentials **scale invariant**! Equivalent to

$$\phi_D(\text{“red”}) = 6, \quad \phi_D(\text{“blue”}) = 3, \quad \phi_D(\text{“green”}) = 1$$

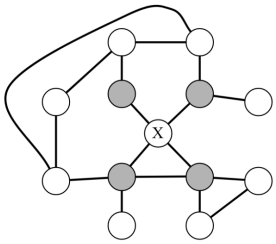
Figure 24: Undirected Graphs : Example 2



- $X_{\mathbf{A}} \perp X_{\mathbf{C}} \mid X_{\mathbf{B}}$ if there is no path from $a \in \mathbf{A}$ to $c \in \mathbf{C}$ after removing all variables in $\mathbf{B}$

Figure 25: Undirected Graphs : Conditional independence

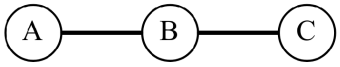
- A set $\mathbf{U}$ is a **Markov blanket** of X if $X \notin \mathbf{U}$ and if $\mathbf{U}$ is a minimal set of nodes such that $X \perp (\mathcal{X} - \{X\} - \mathbf{U}) \mid \mathbf{U}$
- In undirected graphical models, the Markov blanket of a variable is precisely its **neighbors** in the graph:



- In other words, X is independent of the rest of the nodes in the graph given its immediate neighbors

Figure 26: Markov Blanket

- We will show that $A \perp C \mid B$ for the following distribution:



$$p(a,b,c) = \frac{1}{Z}\phi_{AB}(a,b)\phi_{BC}(b,c)$$

- First, we show that $p(a \mid b)$ can be computed using only $\phi_{AB}(a,b)$:

$$\begin{aligned} p(a \mid b) &= \frac{p(a,b)}{p(b)} \\ &= \frac{\frac{1}{Z}\sum_{\hat{c}} \phi_{AB}(a,b)\phi_{BC}(b,\hat{c})}{\frac{1}{Z}\sum_{\hat{a},\hat{c}} \phi_{AB}(\hat{a},b)\phi_{BC}(b,\hat{c})} \\ &= \frac{\phi_{AB}(a,b)\sum_{\hat{c}} \phi_{BC}(b,\hat{c})}{\sum_{\hat{a}} \phi_{AB}(\hat{a},b)\sum_{\hat{c}} \phi_{BC}(b,\hat{c})} = \frac{\phi_{AB}(a,b)}{\sum_{\hat{a}} \phi_{AB}(\hat{a},b)}. \end{aligned}$$

- More generally, the probability of a variable conditioned on its Markov blanket depends *only* on potentials involving that node

Figure 27: Undirected Graphs : Distributions

11.3.1 Factor Graph

- G does not reveal the structure of the distribution: maximum cliques vs. subsets of them
- A **factor graph** is a bipartite undirected graph with variable nodes and factor nodes. Edges are only between the variable nodes and the factor nodes
- Each factor node is associated with a single potential, whose scope is the set of variables that are neighbors in the factor graph

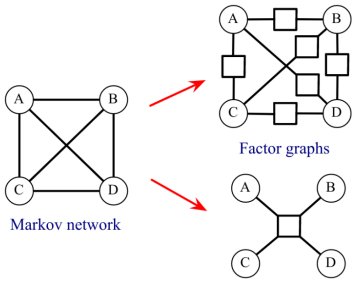
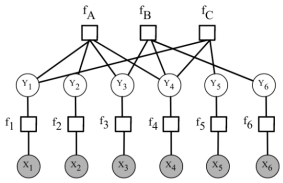


Figure 28: Factor graphs

We might need higher order potentials for representing distribution, i.e. : $\Phi(x, y, z)$. Hence Factor Graphs.

Introduce one potential function for each CPD: $\Phi_i(x_i, x_{pa(i)}) = P(x_i|x_{pa(i)})$.

11.3.2 Low-density parity-check LDPC



- The *decoding* problem for LDPCs is to find

$$\text{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$$

  This is called the **maximum a posteriori** (MAP) assignment
- Since Z and $p(\mathbf{x})$ are constants with respect to the choice of $\mathbf{y}$, can equivalently solve (taking the log of $p(\mathbf{y}, \mathbf{x})$):

$$\text{argmax}_{\mathbf{y}} \sum_{c \in C} \theta_c(\mathbf{y}_c, \mathbf{x}_c),$$

  where $\theta_c(\mathbf{x}_c) = \log \phi_c(\mathbf{y}_c, \mathbf{x}_c)$
- This is a discrete optimization problem!

Figure 29: Low-density parity-check

- Procedure for converting a Bayesian network into a Markov network
- The **moral graph** $\mathcal{M}[G]$ of a BN $G = (V, E)$ is an undirected graph over V that contains an undirected edge between X_i and X_j if
  1. there is a directed edge between them (in either direction)
  2. X_i and X_j are both parents of the same node

(term historically arose from the idea of "marrying the parents" of the node)

- The addition of the moralizing edges leads to the loss of some independence information, e.g., $A \rightarrow C \leftarrow B$, where $A \perp B$ is lost

Figure 30: Low-density parity-check

11.3.3 Gibbs Distribution

$p(x)$ is a Gibbs distribution over G if it can be written as :

$$p(x_1, \cdots, x_n) = \frac{1}{Z} \prod_{c \in C} \Phi_c(x_c) \tag{701}$$

(702)

- Theorem (**soundness of separation**): If $p(\mathbf{x})$ is a Gibbs distribution for G, then G is an I-map for $p(\mathbf{x})$, i.e. $I(G) \subseteq I(p)$
  Proof: Suppose **B** separates **A** from **C**. Then we can write

$$p(\mathbf{X_A}, \mathbf{X_B}, \mathbf{X_C}) = \frac{1}{Z} f(\mathbf{X_A}, \mathbf{X_B}) g(\mathbf{X_B}, \mathbf{X_C}).$$

Figure 31: Gibbs distribution

- What about the converse? We need one more assumption:
- A distribution is **positive** if $p(\mathbf{x}) > 0$ for all $\mathbf{x}$
- Theorem (**Hammersley-Clifford**, 1971): If $p(\mathbf{x})$ is a positive distribution and G is an I-map for $p(\mathbf{x})$, then $p(\mathbf{x})$ is a Gibbs distribution that factorizes over G
- Proof is in Koller & Friedman book (as is counter-example for when $p(\mathbf{x})$ is not positive)
- This is important for **learning**:
  - Prior knowledge is often in the form of conditional independencies (i.e., a graph structure G)
  - Hammersley-Clifford tells us that it suffices to search over Gibbs distributions for G – allows us to *parameterize* the distribution

Figure 32: Gibbs distribution

11.3.4 Learning graphical models from data

Three problems we can solve :

- Density estimation: we are interested in the full distribution (so later we can compute whatever conditional probabilities we want).
- Specific prediction tasks: we are using the distribution to make a prediction.
- Structure or knowledge discovery: we are interested in the model itself (often of interest in data science).

Closeness ?

$$D(p^*||p_\theta) = E_{x\sim p^*}[\log(\frac{p^*(x)}{p_\theta(x)})] \tag{703}$$
$$= -H(p^*) - E_{x\sim p^*}[\log(p_\theta(x))] \tag{704}$$
$$\tag{705}$$

Minimizing the KL-divergence is equivalent to maximizing the log likelihood. Problem : because of $\log$ if close to 0 inside will weight heavily. We also do no know p^*.

We can approximate log likelihood with empirical log likelihood :

$$\max_\theta \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \log p_\theta(x) \tag{706}$$

Remark : regularization is often equivalent to MAP where we put prior on θ.

$$objective(x, M) = loss(x, M) + R(M) \tag{707}$$
$$\log p(\theta|x) = \log p(x, \theta) + \log p(\theta) - constant \tag{708}$$

- Suppose that we know the Bayesian network structure G
- Let $\theta_{x_i|\mathbf{x}_{pa(i)}}$ be the parameter giving the value of the CPD $p(x_i \mid \mathbf{x}_{pa(i)}; \theta)$
- Maximum likelihood estimation corresponds to solving:

$$\max_\theta \sum_{n=1}^{N} \log p(\mathbf{x}^n; \theta) = \max_\theta \ell(\theta; \mathcal{D})$$

  subject to the non-negativity and normalization constraints
- This is equal to:

$$\begin{aligned}\max_\theta \sum_{n=1}^{N} \log p(\mathbf{x}^n; \theta) &= \max_\theta \sum_{n=1}^{N}\sum_{i=1}^{|V|} \log p(x_i^n \mid \mathbf{x}_{pa(i)}^n; \theta) \\ &= \max_\theta \sum_{i=1}^{|V|}\sum_{n=1}^{N} \log p(x_i^n \mid \mathbf{x}_{pa(i)}^n; \theta)\end{aligned}$$

- The optimization problem decomposes into an independent optimization problem for each CPD!

Figure 33: Learning bayesian networks

$$
\begin{aligned}
\ell(\theta; \mathcal{D}) = \log p(\mathcal{D}; \theta) &= \sum_{i=1}^{|V|} \sum_{n=1}^{N} \log p(x_i^n \mid \mathbf{x}_{pa(i)}^n; \theta) \\
&= \sum_{i=1}^{|V|} \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} \sum_{\substack{\hat{\mathbf{x}} \in \mathcal{D}: \\ \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{pa(i)} = \mathbf{x}_i, \mathbf{x}_{pa(i)}}} \log p(x_i \mid \mathbf{x}_{pa(i)}; \theta) \\
&= \sum_{i=1}^{|V|} \sum_{\mathbf{x}_{pa(i)}} \sum_{\mathbf{x}_i} N_{x_i, \mathbf{x}_{pa(i)}} \log \theta_{x_i | \mathbf{x}_{pa(i)}},
\end{aligned}
$$

where $N_{x_i, \mathbf{x}_{pa(i)}}$ is the number of times that the (partial) assignment $x_i, \mathbf{x}_{pa(i)}$ is observed in the training data

- We have the closed form ML solution:

$$
\theta^{ML}_{x_i | \mathbf{x}_{pa(i)}} = \frac{N_{x_i, \mathbf{x}_{pa(i)}}}{\sum_{\hat{x}_i} N_{\hat{x}_i, \mathbf{x}_{pa(i)}}}
$$

- We were able to estimate each CPD independently because the objective **decomposes** by variable and parent assignment

Figure 34: Learning bayesian networks

It won't work that easily for bayesian networks.

- The global normalization constant $Z(\theta)$ kills decomposability:

$$
\begin{aligned}
\theta^{ML} &= \arg\max_{\theta} \log \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x}; \theta) \\
&= \arg\max_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \left(\sum_{c} \log \phi_c(\mathbf{x}_c; \theta) - \log Z(\theta) \right) \\
&= \arg\max_{\theta} \left(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{c} \log \phi_c(\mathbf{x}_c; \theta) \right) - |\mathcal{D}| \log Z(\theta)
\end{aligned}
$$

Figure 35: Learning bayesian networks

11.4 Likelihood-Free Inference

Also Approximate Bayesian Computation, Simulation, Agent-based models.

Introduction You have to test an hypothesis, example : Does the Higgs boson exists. You have a simulator that can simulate examples with and without the Higgs boson theory. You can train a classifier to classify example x on one of the Hypothesis. Now you have a real experiment on a multi-billion dollar CERN particule accelerator and classify the results, which gives you which hypothesis is correct.

Likelihood Ratio This is the best classification region for hypothesis testing. The KL-divergence is the expect likelihood ratio.

$$L(s) = \int p(x|H_0)(0 - s(x))^2 + p(x|H_1)(1 - s(x))^2 dx \tag{709}$$

$$= \sum (y_i - s(x_i))^2 \tag{710}$$

$$s(x) = \frac{p(x|H_1)}{p(x|H_0) + p(x|H_1)} \tag{711}$$

$$\text{Likelihood Ratio} : \frac{p(x|H_1)}{p(x|H_0)} \tag{712}$$

$$\min - \int p(x) \log p_\theta(x) dx \tag{713}$$

$$\text{subject to} : \int p_\theta(x) dx = 1 \tag{714}$$

$$\approx \frac{1}{N} \sum \log r(x_i) (\text{MCMC}). \tag{715}$$

Known Likelihood : Draw $\theta_i \sim p(\theta)$ prior. Keep it with probability $p(x|\theta)$. Then $\theta \sim p(\theta|x)$.

Unknown Likelihood : Mechanical Rejection Algorithm. Draw $\theta \sim p(\theta)$, Simulate $X \sim f(\theta)$, accept θ if $D = X$, D data point. Extremly long to get samples from this method.

Approximate Bayesian Inference : Accept θ if $\rho(D, x) \leq \varepsilon$, ρ distance, ε slow vs bad sample parameter.

12 Natural Language Processing

12.1 Deep learning and Recurrent Neural Networks

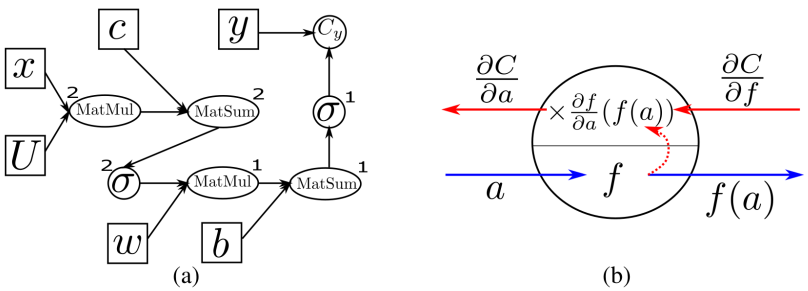


Figure 36: Back propagation on a Directed Acylcic Graph

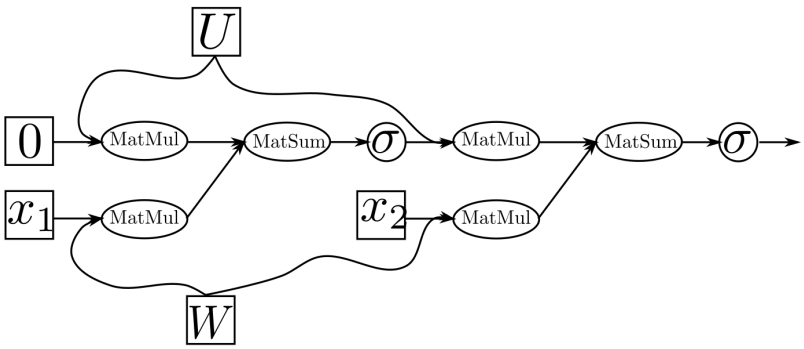


Figure 37: Recurrent Neural Networks Unfolded

Part-of-speech tagging You want to classify each word of a sentence into $NOUN, VERB, ADV, ADJ...$ If you run an RNN on the sentence only you have a prediction which use : $p(y_i|x_1,...,x_i)$. If you run the RNN from start to end and end to start you sum up the states or concatenate or whatever you get : $p(x_i|X)$. If now you run an RNN on the vectors outputted by the RNN you get $p(y_i|X,y_{i-1})$.

Part V

Papers

13 Papers summary

13.1 Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy ?

From [9].

Case where all weights are i.i.d. Gaussian - not trained then - no bias. We see that the angles of close points tends to be closer and the angles of far away points stay far away. The orders of angles stay the same. It performs a stable embedding of the data. The metric information is kept through the network. We can add metric preservation constraints - can help the network generalize better. It does explain a bit extreme learning machines.

We also want to distort the metrics of the data because we want to put elements of the same class closer together - inter class farther. During learning this is a behavior that we see.

Batch Normalization -> You change the referential which changes angles at the same time.

We can view deep neural networks as a stage-wise metric learning process.

Class boundary points are very interesting to study also, as the goal of learning can be seen as treating the class boundary points while keeping the other distances approximately the same.

Important results :

$$|d_{\mathbb{S}^{n-1}}(x,y) - d_{\mathbb{H}^m(g(x),g(y))}| \leq \delta \leq m^{-1/6}\omega(K)^{1/3} \tag{716}$$

$$\omega(K) = E_g[\sup_{x,y\in K} \langle g, x-y\rangle] \text{ width of set } K \text{ in direction } g. \tag{717}$$

$$g : (K \subset \mathbb{S}^{n-1}, d_{\mathbb{S}^{n-1}}) \to (g(K), d_{\mathbb{H}^m}) \tag{718}$$

$$g(x) = sign(f(Mx)) \tag{719}$$

It implies that the area in the image space whose activation pattern is the same as a width lower then δ.

Result on angles :

$$|\|\rho(Mx) - \rho(My)\|_2^2 - (\frac{1}{2}\|x-y\|_2^2 + \|x\|_2\|y\|_2\psi(x,y))| \leq \delta \tag{720}$$

$$0 \leq \angle(x,y) \triangleq \cos^{-1}(\frac{x^T y}{\|x\|_2\|y\|_2}) \leq \pi \tag{721}$$

$$\psi(x,y) = \frac{1}{\pi}(\sin\angle(x,y) - \angle(x,y)\cos\angle(x,y)) \tag{722}$$

This more or less dictate how an angle between two input points behave after a layer.

13.2 Deep Neural Networks are Easily Fooled : High Confidence Predictions for Unrecognizable Images.

From [1].

Generating images of a certain class. Two ways of doing so :

- Directly encoded evolutionary algorithm where you start with a pool of images, remove k worst ones, mutate p images remaining, loop.
- Indirectly encoded images : You have a network, given (x,y) pixel coordinate output p pixel value or r,g,b 3 pixels values. Evolutionnary algorithm to determine which network is the best.

Indirectly encoded generating generate images that looks like something, though not a real image. And Directly encoded EA generates white noise correctly classified. Harder for a directly encoded EA to fool the network on ImageNet. Mnist is easy to fool compared to ImageNet. Gradient Ascent is also working and produce white noise also.

We can train a network DNN_1, then generate fool images, add a class $n+1$ which is fool images, retrain, ... It becomes harder to find fooling images though it's still possible and doesn't reduce generalized error rate.

13.3 Self-Normalizing Neural Networks

From [5].

There exists multiple ways to normalize distributions within neural networks :

- Batch Normalization.
- Layer Normalization.
- Weight Normalization : Output distribution $(0,1)$ if input $(0,1)$

Fully connected neural networks suffer from high variance during training, and adding strong regularization to them often end up inreasing the variance which in return can make the learning process divergent.

$$selu(x) = \lambda \begin{cases} x & x > 0 \\ \alpha\exp(x) - \alpha & x \leq 0 \end{cases} \tag{723}$$

If we assume :

$$\begin{matrix}\mu \\ \nu\end{matrix} = g \begin{matrix}\mu \\ \nu\end{matrix} \tag{724}$$

$$\sum_i w_i = \omega, \sum_i w_i^2 = \tau x_i \text{ i.i.d. } \mu, \nu \tag{725}$$

$$x = w^T x \tag{726}$$

$$\mathbf{E}[z] = \sum_i w_i \mathbf{E}[x_i] = \mu\omega \tag{727}$$

$$\mathrm{Var}(z) = \nu\tau \tag{728}$$

$$\mu(\mu, \omega, \nu, \tau) = \int_{\mathbb{R}} selu(x) P_N(z; \mu\omega, \sqrt{\nu\tau}) dz \tag{729}$$

T.C.L. gives us (with independence used here) : $z \sim \mathcal{N}(\mu, \sqrt{\nu\tau})$. We solve the equations and assume $\mu = 0, \nu = 1$ and find α, λ so that $\mu = 0, \nu = 1$. It gives : $\alpha_{01} = 1.6733$, $\lambda_{01} = 1.0507$.

And $\mu, \nu = (0, 1)$ is stable and attracting because jacobian in the point as norm below than 1.

13.4 Exponential expressivity in deep neural networks through transient chaos

From [6].

Studying the case of random Gaussian weighted neural networks, looking at the evolution of distance (variance) of activations in a neural networks, correlations given the weight's standard deviation, Chaos behaviours out of it (depending on the scale of weights). Using different types of curvatures to shows that complexity is achieved in a neural network through deepness and note wideness.

Notations :

$$x^l = \phi(h^l) \tag{730}$$

$$h^l = W^l x^{l-1} + b^l \tag{731}$$

$$x^l \in \mathbb{R}^{N_l} \tag{732}$$

$$q^l = \frac{1}{N_l} \sum_i (h_i^l)^2 \text{ second moment of the empirical distribution of inputs.} \tag{733}$$

$$W_{i,j}^l \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_{l-1}}), b^l \sim \mathcal{N}(0, \sigma_b^2) \tag{734}$$

q^l converges to a zero-mean gaussian as sum of uncorrelated random variables. We have :

$$q^l = (q^{l-1} | \sigma_w, \sigma_b) = \sigma_w^2 \in \mathcal{D}z \phi(\sqrt{q^{l-1}} z)^2 + \sigma_b^2 \tag{735}$$

$$\mathcal{D}z = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz \tag{736}$$

$$q^1 = \sigma_w^2 q^0 + \sigma_b^2, q^0 = \frac{1}{N_0} x^0 \cdot x^0 \tag{737}$$

Bias free study of distance :

If $\sigma_w < 1$ only one fixed point $q^* = 0$, all inputs are shrinked to the origin. If $\sigma_w > 1$, $q^* = 0$ unstable fixed point and another fixed point appears (stable). There is rapid convergence to the fixed point in often 4 layers.

Now let's look at correlations, given two points $x^{0,a}, x^{0,b}$:

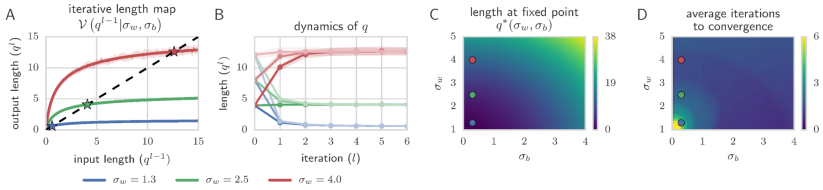


Figure 38: Graph of function

$$q^l_{a,b} = \frac{1}{N_l}\sum_i h^l_i(x^{0,a})h^l_i(x^{0,b}) \tag{738}$$

$$q^l_{1,2} = \mathcal{C}(c^{l-1}_{1,2}, q^{l-1}_{1,1}, q^{l-1}_{2,2}|\sigma_w,\sigma_b) \equiv \sigma_w^2 \int \mathcal{D}z_1\mathcal{D}z_2\phi(u_1)\phi(u_2) + \sigma_b^2 \tag{739}$$

$$u_1 = \sqrt{q^{l-1}_{11}}z_1, u_2 = \sqrt{q^{l-1}_{22}}[c^{l-1}_{12}z_1 + \sqrt{1-(c^{l-1}_{12})^2}z_2] \tag{740}$$

$$c^l_{12} = q^l_{12}(q^l_{11}q^l_{22})^{-1/2} \text{ correlation coefficient.} \tag{741}$$

$$\tag{742}$$

We compute c^* with :

$$c^l_{12} = \frac{1}{q^*}\mathcal{C}(c^{l-1}_{12}, q^*, q^*|\sigma_w,\sigma_b)\chi_1 \equiv \frac{\partial c^l_{12}}{\partial c^{l-1}_{12}}_{c=1} = \sigma_w^2 \int \mathcal{D}z[\phi'(\sqrt{q^*}z)]^2 \tag{743}$$

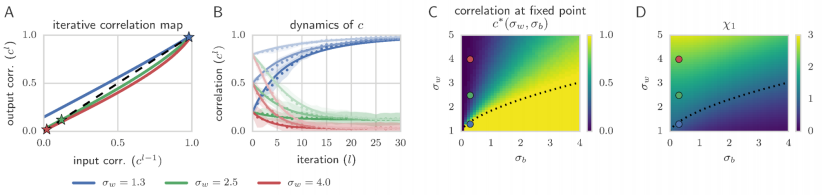


Figure 39: Graph of function

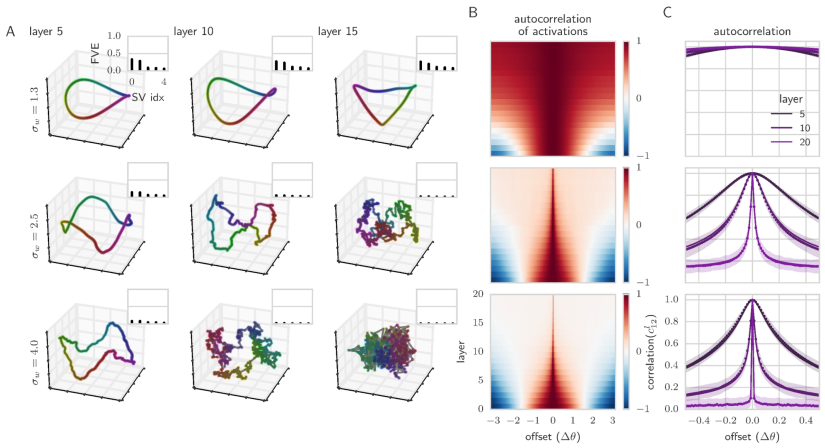
χ_1 multiplicative stretch factor. χ_1 partition the space into chaotic $\chi_1 > 1, c^* < 1$ and ordered $\chi_1 < 1, c^* = 1$ regions.

Now we look at the complexity of decision boundaries changing different hyper-parameters, first propagating a circle through random sigmoidal networks, varying σ_w fixed $\sigma_b = 0.3$:

Curvature : Given a simple input manifold such as $h^1(\theta) = \sqrt{N_1 q}[u^0\cos(\theta) + u^1\sin(\theta)]$ at each point of the manifold $h(\theta)$ you can define $v(\theta) = \partial_\theta h(\theta)$. Intuitively curvature is related to how quickly this tangent vector rotates in the ambiand space as one moves along the manifold or close to acceleration $a(\theta) = \partial_\theta v(\theta)$. At each θ, we have a vectorial space defined by $v(\theta), a(\theta)$ of dimension 2. Within this manifold there is a unique circle of radius $R(\theta)$ that has the same velocity and acceleration as the curve $h(\theta)$ in θ : the osculating circle which lead to extrinsic curvature $\kappa(\theta)$.

$$\kappa(\theta) = \frac{1}{(v\cdot v)^{3/2}}\sqrt{(v\cdot v)(a\cdot a) - (v\cdot a)^2} \tag{744}$$

If unit speed parametrization $v.v = 1, a.v = 0$ we find the acceleration.



Euclidean length $\mathcal{L}^E$: The euclidean metric induces a metric on the curve $g^E(\theta) = v(\theta) \cdot v(\theta)$. The distance $d\mathcal{L}^E$ when moving from θ to $\theta + d\theta$ on the curve is $d\mathcal{L}^E = \sqrt{g^E(\theta)}d\theta$. The total curve length is then :

$$\mathcal{L}^E = \int \sqrt{g^E(\theta)}d\theta \tag{745}$$

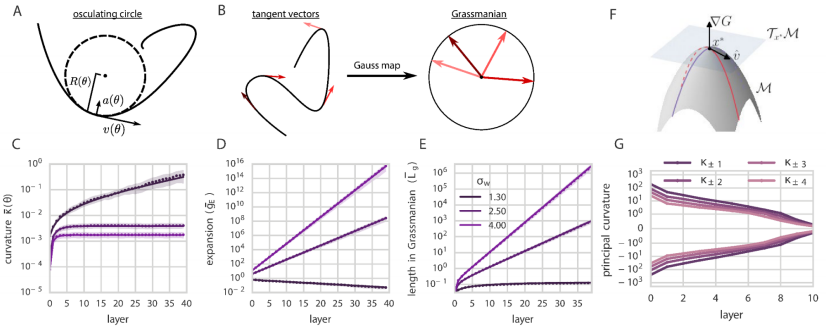
However even straight line segment can have large Euclidean length.

Gauss map. length metric : For a K dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^N$ the Gauss map maps a point $\theta \in \mathcal{M}$ to its K dimensional tengent plane $\mathcal{T}_\theta\mathcal{M} \in \varsigma_{K,N}$ grassmannian manifold of all K dimensional subspaces in $\mathbb{R}^n$. If $K = 1$ $\varsigma_{K,N} = \mathbb{S}^{n-1}$ is the sphere with antipodal point identified since 1-dimensional subspace can be identified with a unit vector modulo sign. The gauss map takes unit velocity vector : $\hat{v}(\theta) = \frac{v(\theta)}{\sqrt{v(\theta)\cdot v(\theta)}}$, the natural metric on the sphere induces a Gauss metric on the curve $g^G(\theta) = (\partial_\theta \hat{v}(\theta)) \cdot (\partial_\theta \hat{v}(\theta))$ measure how quickly the unit tangent vector changes with θ.

$$\mathcal{L}^G = \int \sqrt{g^G(\theta)}d\theta \tag{746}$$

$$g^G(\theta) = \kappa(\theta)^2 g^E(\theta) \tag{747}$$

Looking at those quantities renormalized with respect to the number of activation we have :



We can also show that with one hidden layer and growing the number of activation does not yield the same level of complexity as the one achieved with deepness.

13.5 On the Expressive Power of Deep Neural Networks

From [6].

Showing that :

- Complexity of a neural network grows exponentially with depth. Using a measure of expressivity based on the non-linearity component of neural networks.
- Initial layers matters more because they are more subject to noise.
- Trajectory regularization - a novel method for replacing batch normalization.

Defining :

- Activation patterns as $\{0,1\}$ for ReLU activations and $\{-1,0,1\}$ for Tanh activations.
- Transition : changing input from x to $x+\delta$ changes the activation pattern. Studying behaviour of transitions following a one dimension parametrized trajectory $x(t)$: Trajectory length grows exponentially in the depth of the network.
- $F_A(x;W)$ the neural networks activations. We can define :

$$\mathcal{T}(F_A(x(t);W)) \text{ number linear regions as we go along the path.} \tag{748}$$

$$\mathcal{AP}(F_A(x;W)) \text{ activation pattern - } \in \{0,1\}^{numneuron} \tag{749}$$

$$\mathcal{A}(F_A(x(t);W)) \text{ number of distinct activation patterns along the path.} \tag{750}$$

- Trajectory length :

$$l(x(t)) = \int_t \|\frac{dx(t)}{dt}\|dt \tag{751}$$

$$z^{(d)}(x(t)) = z^{(d)}(t) \text{ trajectory of } x(t) \text{ in layer } d. \tag{752}$$

Montufar et al. [4] theory of hyperplane arrangement : Choose a set of weights which result in an exponential increase of linear regions with the depth of the architectures. In reality we are from this construction at initialization but we learn to get close to it. Theorem generalizing that given $A_{(n,k)}$ neural network with n hidden layers of size k :

$$\forall W, \mathcal{AP}(F_A(\mathbb{R}^m;W)) \leq O(k^{mn}) \tag{753}$$

This result is strange and I don't fully understand it : the maximum number of hyperplanes possible is 2^{kn} no ?

The input space regions is partitioned by the neural network into convex polytopes corresponding to different linear function on each regions. Understanding the density of these regions during the training process would likely shed light on properties of the loss surface and improve optimization methods.

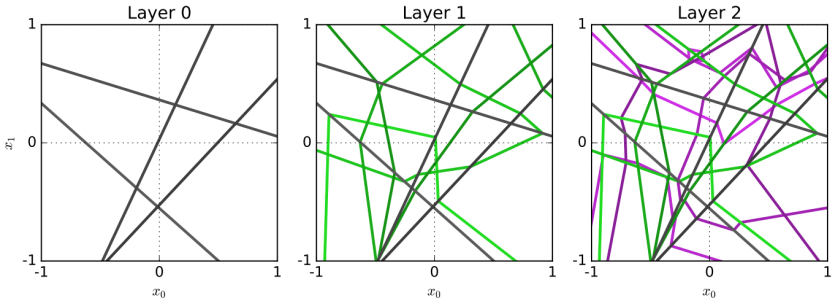


Figure 40: Polytopes of layer 1,2,3 of a neural network with input $\mathbb{R}^2$ and 4 units per hidden layer.

If $x(t)$ a trajectory with $x(t+\delta)$ having a non-trivial prependicular component to $x(t), \forall t, \delta$ then :

$$\mathbf{E}[l(z^{(d)}(t))] \geq \mathcal{O}\left(\frac{\sigma_w \sqrt{k}}{\sqrt{k+1}}\right)^d l(x(t)) \tag{754}$$

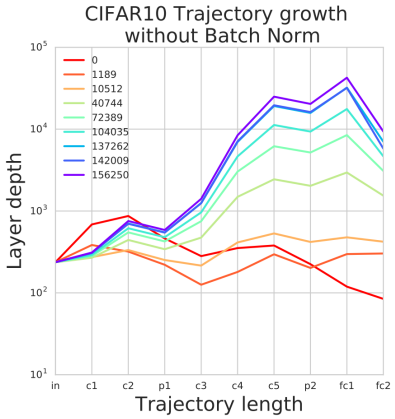
Figure 41: Forward of a circle into a neural network trained on MNIST, we see the length increase.

Another interesting result is that the number of transitions is linear in the trajectory length for a variety of neural networks and weight variance.

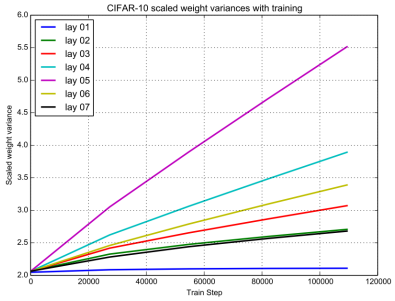
Also, if we add noise to one layer's weights, we see that the first layer is way more sensitive than other layers : A pertubation at a layer grows exponentially in the remaining depth after that layer. The converse does hold to some extent : If you take a network and train only one of its layer : The more depth you have the better accuracy you get.

Two interesting graphics :

- The length of path $l(l(z^{(d)}(t)))$ grows as you learn the network : At first you are far from having the exponential expressivity that you can reach but learn to get it.



- Learning increase the scale of weight by an enormous amount :



13.6 Geodesics of learned representations

From [8].

The goal is to find a way to asses the invariances of learned representations. In order to do so you can use Geodesics : Interpolating between two images under some conditions, so you can asses the invariances properties of a classification network. Invariances are fundamental requirement of pattern recognition as you can ignore irrelevant variations of the input, and image synthesis provides a powerful methodology to assess the invariances of arbitrary representations.

Example : Given an image, you can synthesize a new image with the same representation (under a transformation) and thus assess which invariances the transformation has (Synthesis test). For instance, with fourier transform you have shift invariance as well as various phases invariance which might not be perfect.

Issue with synthesis test : If you synthesis the same image, it's good but your representation might don't learn any invariance, like Identity function. If different image, you have invariances you don't want.

Conditional Geodesic : Given two images, initial and final $\{x_0, x_N\}$ we wish to synthesis a sequence of images $\gamma = \{x_n; n \in [0, ..., n]\}$ lying along a geodesic representation space :

$$L[f(\gamma)] = \sum_{n=1}^{N} \|f(x_n) - f(x_{n-1})\|_2 \tag{755}$$

$$E[f(\gamma)] = \sum_{n=1}^{N} \|f(x_n) - f(x_{n-1})\|_2^2 \tag{756}$$

$$\text{Cauchy-Schwartz} : L[f(\gamma)]^2 \leq NE[f(\gamma)] \tag{757}$$

$$E[\gamma] = \sum_{n=1}^{N} \|x_n - x_{n-1}\|_2^2 \tag{758}$$

If you find the path that minimize $E[f(\gamma)]$ there is an infinity, conditioned on minimizing $E[f(\gamma)]$ you minimize $E[\gamma]$ there is unicity.

Algorithm 11 Conditional geodesic computation

Require: f : continuous mapping
Require: x_0, x_N initial and final images
Require: N number of steps along geodesic path ($N = 10$ usually)
Require: λ gradient descent step size
Ensure: $\gamma = \{x_n; n = 0, ..., N\}$ minimizes $E[\gamma]$ conditionned on minimizing $E[f(\gamma)]$
  $x_n \leftarrow \frac{N-n}{N} x_0 + \frac{n}{N} x_N, n \in [0, ..., N]$
  minimize $E[f(\gamma)]$
  while γ has not converged **do**
    $d_r \leftarrow \nabla_\gamma E[f(\gamma)]$
    $d_p \leftarrow \nabla_\gamma E[\gamma]$
    $\hat{d_p} \leftarrow d_p - \frac{\langle d_r, d_p \rangle}{\|d_r\|_2^2} d_r$
    $\gamma \leftarrow \gamma - \lambda \hat{d_p}$
    minimize $E[f(\gamma)]$
  return γ

VGG is not very good to linearize basic geometric transformations, we can change the max pooling which creates aliasing artifacts. To avoid aliasing when subsampling by a factor 2, the Nyquist theorem requires blurring with filter whose cutoff frequency is below $\frac{\pi}{2}$: Pooling $L_2(x) = \sqrt{g \star x^2}$, with $g(.)$ 6×6 Hanning window which approx. Nyquist criterion.

When we progress in a neural network, the receptive field size increase with pooling/convolutions. In order to see if it's the receptive field size or the complexity (multiple convolutions) which encapsulate the linearized range of real-world transformation, experiment with few convnet and big pooling so receptive field size is the same.

The receptive field quantifies the strength of the connection between a location in the image and that neuron's activity, can be measured by computing the magnitude of the gradient of the neuron's activity with respect to the image. In order quantify a neuron's receptive field you can look at the gradient of plenty of white noise images.

Shallower networks even with same receptive field size are unable to linearize translations as well as deeper ones.

Coarse-to-fine approach : In a video, if there is a region in the image with periodic structure like woven cane texture of a chair, the motion between initial frame and last frame is ambiguous : Temporal aliasing. In motion estimation, this problem is usually solved with coarse-to-fine approach, in which the motion of the low frequencies is estimated first then used to condition motion estimates derived from higher frequencies. We can do this in our framework with nested conditionalization of geodesic. That is, each layer of the network can impose its own geodesic constraints, conditioned on those imposed by deeper layers. This hierarchical construction provides a means of solving the problem temporal aliasing.

Discussion : We could test the invariance of a system to a given transformation by examining the variability of a responses to objects deformed by the corresponding operation. But you need a meaning-full measure of variability which is hard to do. Geodesic sequences avoids this problem by expressing the invariance properties of the representation back in the input image domain.

13.7 Parseval Networks: Improving Robustness to Adversarial Examples

From [10].

When learning a classification neural network, there is the problem of adversarial examples. Several works has been done to regularize this problem. The most common way nowadays is data augmentation throughout learning by generating adversarial examples. Another proposed way : Make the network lipschitz with a small constant (1). If the network is lipschitz, then the network is more robust to adversarial examples.

In order to lipschitize a network, we can use Parseval tight frames :

$$R_\beta(W_k) = \frac{\beta}{2}\|W_k^T W_k - I\|_2^2 \tag{759}$$

$$\text{Easy optim} : W_k \leftarrow (1+\beta)W_k - \beta W_k W_k^T W_k \tag{760}$$

Also replace pooling operations by $\sum_{n'} \alpha^{(n,n')} n'(x)$ with $\alpha^{(n,n')} \geq 0$ and $\sum_{n'} \alpha^{(n,n')} = 1$.

It similar to better accuracy on test set, and is more robust on similar signal to noise ration (SNR).

Use of capacity : For each layer k of fully connected network, compute the activation's empirical co-variance matrix $\frac{1}{n}\sum_k \phi_k(x)\phi_k(x)^T$ and observe it's eigenvalues and look at the min number of eigenvalues necessary to encapsulate 99% of the variance.

13.8 Distilling the Knowledge in a Neural Network

From [12].

The idea :

Table 2. Number of dimensions (in % of the total dimension) necessary to capture 99% of the covariance of the activations.

	SGD-wd		SGD-wd-da		Parseval	
	all	class	all	class	all	class
Layer 1	72.6	34.7	73.6	34.7	89.0	38.4
Layer 2	1.5	1.3	1.5	1.3	82.6	38.2
Layer 3	0.5	0.5	0.4	0.4	81.9	30.6
Layer 4	0.5	0.4	0.4	0.4	56.0	19.3

Figure 42: Number of dimensions explaining the covariance.

- Learn multiple models on a task, or even a single neural network. You can use temperature to increase the entropy of the probability distribution over labels :

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{761}$$

- For each training example, you have now given your models a probability distribution over labels from multiple models, or even a single neural network.
- Given a smaller neural network - deployable in production - learn it to match the inputs to the soft-max with L_2 norm.

Benefits :

- From cumbersome models to deploy in production, to actually easy to deploy models.
- Actually help against overfitting.

13.9 Stochastic Optimization for Large-scale Optimal Transport - summary to write

From [3].

13.10 From optimal transport to generative modeling: the VEGAN cookbook - summary to write

From [7].

13.11 Sinkhorn-AutoDiff: Tractable Wasserstein Learning of Generative Models - summary to write

From [2].

13.12 Wasserstein Learning of Deep Generative Point Process Models

Examples of temporal point processes :

- Homogeneous and non-homogeneous Poisson processes.
- Self-exciting point processes.
- Self-correcting point process models.
- Survival processes.

Let S be a compact space equipped with a Borel σ-algebra $\mathcal{B}$. Ξ set of counting measures on S with $\mathcal{C}$ smallest σ-algebra on it. $(\Omega, \mathcal{F}, \mathbb{P})$ probability space. A point process S is a measurable map $\xi : \Omega \to \Xi$.

Every realization of a point process can be written as $\xi = \sum_{i=1}^{n} \delta_{X_i}$ with n integer-valued RV, X_i random elements of S - events. It can be represented by the counting process : $N(B) = \int_B \xi(x)dx$ number of events in each Borel subset B. The mean measure M of a point process $M(B) = \mathbb{E}[N(B)]$ expected number of events.

Inhomogeneous Poisson process $M(B) = \int_B \lambda(x)dx$ with $\lambda(x)$ intensity function, positive measurable function on S. Homoegenous poisson process : $\lambda(x) = \lambda, M(B) = |B|$.

Cox point processes $\lambda(x)$ can be a random density possibly depending on the history. Given $\lambda(.), N(B) \sim Poisson(\int_B \lambda(x)dx)$. If $B_1, ..., B_k$ disjoint, $N(B_1), ..., N(B_k)$ independent conditioning on λ.

Temporal point processes $S = [0, T[$. $\xi = \sum_{i=1}^{n} \delta_{t_i} = \{t_1, ..., t_n\}$. Using a conditional intensity rate function is used to characterized such processes.

In inhomogeneous poisson process, $\lambda(t) = \sum_{i=1}^{k} \alpha + i(2\pi\sigma_i^2)^{-1/2} \exp(-(t-c_i)^2/\sigma_i^2)$.

Self-exciting (Hawkes) process $\lambda(t) = \mu + \beta \sum_{t_i<t} g(t-t_i), g(t) = \exp(-\omega t)$ often.

Self-correcting processes $\lambda(t) = \exp(\eta t - \sum_{t_i<t})\gamma$.

Wasserstein distance for temporal point processes

$$\xi = \{x_1, \cdots, x_n\} \tag{762}$$

$$\rho = \{y_1, \cdots, y_m\} \tag{763}$$

$$\text{If } n = m : \|\xi - \rho\|_\star = \min \sum_{i=1}^{n} \|x_i - y_{\sigma(i)}\|_\circ \tag{764}$$

By Birkhoff's theorem. In general case assuming $n < m$:

$$\|\xi - \rho\|_\star = \min \sum_{i=1}^{n} \|x_i - y_{\sigma(i)}\|_\circ + \sum_{i=n+1}^{m} \|s - \tau_{\sigma(i)}\| \tag{765}$$

s fixed limiting point in border of the compact space S and the minimum is over all permutations.

Enforcing Lipschitz constant They regularize based on lipschtiz pairs of the training batch !

13.13 Reading List

- Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. `http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/`.
- Entropy Stochastic Gradient Descent !!! That's my idea wesh.

References

[1] Anh Nguyen et al. *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*. URL: `https://arxiv.org/pdf/1412.1897.pdf`.

[2] Aude Genevay et al. *Sinkhorn-AutoDiff: Tractable Wasserstein Learning of Generative Models*. URL: `https://arxiv.org/pdf/1706.00292.pdf`.

[3] Aude Genevay et al. *Stochastic Optimization for Large-scale Optimal Transport*. URL: `https://arxiv.org/pdf/1605.08527.pdf`.

[4] Guido Montufar et al. *On the Number of Linear Regions of Deep Neural Networks*. URL: `https://arxiv.org/pdf/1402.1869.pdf`.

[5] Günter Klambauer et al. *Self-Normalizing Neural Networks*. URL: https://arxiv.org/pdf/1706.02515.pdf.

[6] Maithra Raghu et al. *On the Expressive Power of Deep Neural Networks*. URL: https://arxiv.org/pdf/1606.05336.pdf.

[7] Olivier Bousquet et al. *From optimal transport to generative modeling: the VEGAN cookbook*. URL: https://arxiv.org/pdf/1705.07642.pdf.

[8] Olivier J. Henaff et al. *Geodesics of learned representations*. URL: https://arxiv.org/pdf/1511.06394.pdf.

[9] Raja Giryes et al. *Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy ?* URL: https://arxiv.org/pdf/1504.08291.pdf.

[10] Moustapha Cisse. *Parseval Networks: Improving Robustness to Adversarial Examples*. URL: https://arxiv.org/pdf/1704.08847.pdf.

[11] Carlos Fernandez-Granda. *Random projections and compressed sensing*. URL: http://www.cims.nyu.edu/~cfgranda/pages/OBDA_spring16/material/random_projections.pdf.

[12] Geoffrey Hinton. *Distilling the Knowledge in a Neural Network*. URL: https://www.cs.toronto.edu/~hinton/absps/distillation.pdf.

[13] Terence Tao. *Notes 1: Concentration of measure*. URL: https://terrytao.wordpress.com/2010/01/03/254a-notes-1-concentration-of-measure/.