

CIS 520 POD MATERIALS

Siming He

November 2, 2024, 02:15

Contents

1	POD 1	3
1.1	Introduction	3
1.2	How is the course going?	3
1.3	Entropy and Information Gain	3
1.4	Norms	4
2	POD 2	6
2.1	Logistics	6
2.2	Questions Collections	6
2.3	KNN and Decision Tree	6
2.4	MLE and MAP	11
3	POD 3	11
3.1	Logistic	11
3.2	Understand how different regularization penalizes weight	11
3.3	Bias and Variance	11
3.4	Streamwise/Stepwise/Stagewise Search	11
3.5	Different forms of the error	11
4	POD 4	14
4.1	Mutual Information	14
4.2	Generalized Linear Models	14
4.3	Gradient Descent Methods	14

5	POD 5	16
5.1	Convolutional Neural Network	16
6	POD 8	19
6.1	SVD, PCA, Autoencoder	19
7	POD 9	22
7.1	Recommend System, Evaluation Metrics	22
8	POD 10	26
8.1	K-Mean, GMM, EM	26
9	POD 11	28
9.1	Naive Bayes and LDA	28
10	POD 12	32
10.1	Reinforcement Learning	32

1 POD 1

1.1 Introduction

Name, Major, Why machine learning interests you?, What is your favorite hobby?

1.2 How is the course going?

1. Welcome email and question collection
2. Wiki Page, Ed discussion
3. Worksheets, Quiz, HWs, Due Dates?
4. Latex and Overleaf
5. Colab
6. pennkey and podname

1.3 Entropy and Information Gain

$$\begin{aligned} H(X) &= - \sum_i P(X = x_i) \log P(X = x_i) \\ &= \sum_i P(X = x_i) \log \frac{1}{P(X = x_i)} \end{aligned}$$

What is Entropy?

The smallest number of bits on average per symbol to transmit a stream of symbols from distribution of random variable X .

Expectation of information content (smallest number of bits on average) of a message of X (represented as bits $\log_2 p_j$).

Why use log probability to encode the length of information content?

The number of combinations of states is exponential. Hence, we need exponential number of "classes" to encode a message. Encode information content as bits.

Do you expect higher or lower entropy with more splits of a same distribution?

Splits over uniform distribution: $-\sum_{i=1}^2 \frac{1}{2} \log \frac{1}{2} = 1$, $-\sum_{i=1}^4 \frac{1}{4} \log \frac{1}{4} = 2$. More splits, more uncertainty, hence higher entropy.

What about a random distribution?

Which has higher entropy?

Show density function of Uniform(0,1), Uniform(-1,1), Normal(0,1), Normal(2,5)

Related back to the definition. More chaotic, need more bits to describe it, higher entropy.

$-\int P(X = x_i) \log P(X = x_i) dx$ needs this differential entropy where perfect certain means $-\infty$ entropy and total uncertain means ∞ entropy.

The entropy of Uniform(0,1) is 0 but it doesn't mean it's perfect certain. Such entropy is called differential entropy (not Shannon entropy) and the perfect certain means $-\infty$ entropy.

What is $-p_i \log p_i, p_i = 0$?

$= 0$

What is the relationship between information gain and entropy for a given dataset?

$$G(Y|X) = H(Y) - H(Y|X)$$

We rethink information gain under the framework of mutual information. During variable/data selection, we want to select the data X to maximize the information gain Y . In many cases, such information gain is also called mutual information of X and Y : $I(X, Y) = D_{KL}(P_{(X,Y)} || P_X \cdot P_Y)$ which is calculated differently but the result would be the same. On the Wikipedia page of mutual information, you can find why the two are fundamentally the same. Through this formula, we can also understand information gain as the additional bits needed to encode $P_{(X,Y)}$ if we have $P_X \cdot P_Y$. Intuitively, if X and Y are independent, we don't need additional bits to encode $P_{(X,Y)}$ and X provides no information gain of Y . And as X and Y are more dependent, $P_{(X,Y)}$ is more different from $P_X \cdot P_Y$ and we need more bits to encode $P_{(X,Y)}$ and the information gain is higher.

1.4 Norms

Is Euclidean distance a norm?

It's related: $d(x, y) = \|x - y\|_2$. L2 Norm could be considered as a way of calculating Euclidean distance.

Definition of Norm and Distance:

L_p norm is $\|x\|_p = (\sum_j |x_j|^p)^{\frac{1}{p}}$ such that

1. $L_p(av) = |a|L_p(v)$
2. $L_p(u + v) \leq L_p(u) + L_p(v)$
3. $L_p(v) = 0$ iff $v = 0$

Distance is $d_p(x, y) = \|x - y\|_p$ such that

1. $d_p(x, y) \geq 0$ and $= 0$ iff $x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$

Is the number of wrong answers a norm?

It's L1 Norm.

Is pseudo-norm L0 norm?

No, doesn't satisfy property 1 of norm.

Does k-nn require a norm?

No - any measure of similarity or dis-similarity would work.

Why decision tree if we already have knn?

Curse of dimension. Slow with a larger dataset (need to go through data to find neighbors for every new data point.)

What is the max dimensionality under which the knn algorithm is effective? I assume this dimensionality should somehow depend on k.

Theoretically KNN suffers from the curse of dimension, i.e. data become sparse when dimension is higher. Ideally, we need data to grow exponentially as dimension increases. However, it also

depends on the data. If the data are highly correlated (like voxels in a brain), then the data are effectively low dimensional (though in high dimension space), so it isn't a problem. We can do k-nn with hundreds of thousands of features of brain voxels and a hundred observations to predict disease.

2 POD 2

2.1 Logistics

Assignment of partners for Homeworks: follow the excel, **if anyone wants to work alone, please tell me.**

For final project, people pick teammates by themselves, **at least two people.**

Make-up pod session? Friday 4:00 is the designated make-up pod with Zoom link.

How much time are you spending on the course?

How was the worksheets/homeworks/quizzes?

2.2 Questions Collections

Q: What is the max dimensionality under which the knn algorithm is effective? I assume this dimensionality should somehow depend on k .

A: Theoretically KNN suffers from the curse of dimension, i.e. data become sparse when dimension is higher. Ideally, we need data to grow exponentially as dimension increases. However, it also depends on the data. If the data are highly correlated (like voxels in a brain), then the data are effectively low dimensional (though in high dimension space), so it isn't a problem. We can do k -nn with hundreds of thousands of features of brain voxels and a hundred observations to predict disease.

Q: How do you draw decision boundaries for KNN? (question 2 on HW1)

A: Check where the points have equal distance to data points.

2.3 KNN and Decision Tree

Q: Parametric (parameters in a fixed function form) vs. non-parametric

A: Non-parametric model is determined by training data and overfitting is the main concern. Parametric model form doesn't change, training parameters for prediction.

Q: How does k -NN model form change as n gets bigger

A: more points; more resolution

Q: How does a Decision Tree model form change as n gets bigger?

A: deeper tree since needs to recursively split more times

Q: How does a linear regression model form change as n gets bigger?

A: it doesn't

Q: Is the decision tree the optimal way to split data?

A: it is not since decision tree is greedy. It's possible to get a better tree by splitting some "worse" features first.

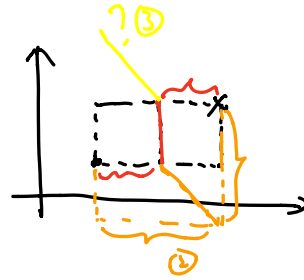
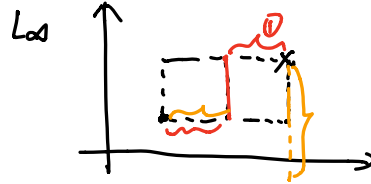
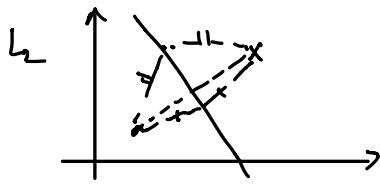
Q: Effect of noise if 90 percent of features are random on KNN and Decision Tree?

A: Decision tree is good at filtering out noise because it's doing feature selection through information gain (random features have low information gain). For KNN, if there is too much noise, random features will make the prediction not good.

Q: Effect of noise if 10 percent of label are random on KNN and Decision Tree?

A: For KNN, choose bigger k would improve performance. For decision tree, don't make the tree too deep (overfitting to the noise).

DB for KNN?



Parametric V.S.

Non-Parametric

Linear regression

Decision tree

k-NN

↳ is it parameter?

$|D_{train}| \uparrow$ form x change ; deeper ; more resolution

90% features are random good since feature selection not good. neighbor overwhelmed by noise.

10% label are random don't too deep $k \uparrow$

Intro to Bayesian Stat:

Classical Stat

θ is fixed

$$\text{MLE: } \hat{\theta}_{\text{MLE}} = \underset{\theta}{\text{argmax}} P(y|\theta)$$

e.g. $y_i \sim N(\mu, \sigma^2)$

$$\theta = (\mu, \sigma^2)$$

independence
→ assumption

$$\mu^*, \sigma^{*2} = \underset{\mu, \sigma^2}{\text{argmax}} \prod_{i=1}^n f_{N(\mu, \sigma^2)}(y_i)$$

Cons: 1. large sample size

CLT

2. we cannot make prob statement on θ

$$\Pr(\mu \in (-1, 1)) = 0 \text{ or } 1$$

Bayesian Stat

θ is r.v. with distribution!

$$\text{Bayes rule } P(\theta|y) = \frac{P(y|\theta) \cdot P(\theta)}{P(y)}$$

posterior $\propto P(y|\theta) \cdot P(\theta)$

"updating distribution of θ " \leftarrow prior

$$\text{MAP: } \hat{\theta}_{\text{MAP}} = \underset{\theta}{\text{argmax}} P(\theta|y)$$

e.g. $y_i \sim N(\mu, \sigma^2)$ fixed

$$\mu \sim N(\mu_0, \tau^2)$$

$$P(\mu|y) \propto P(y|\mu) \cdot P(\mu)$$

$$\propto \prod_{i=1}^n P(y_i|\mu) \cdot P(\mu)$$

$$\mu^* = \underset{\mu}{\text{argmax}} P(\mu|y)$$

$$= \underset{\mu}{\text{argmax}} \int N(\mu', \sigma'^2)(\mu)$$

$$\mu' = \frac{\frac{n}{\sigma^2} \bar{y} + \frac{1}{\tau^2} \mu_0}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}} \quad \sigma'^2 = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}}$$

cons: 1. more model assumption

2. more complex

Prior!

If data $P(y|\theta) \in \mathcal{F}$

$P(\theta) \in \pi$ is conjugate prior for \mathcal{F} if $P(\theta|y) \in \pi$

Normal - Normal

Beta - Binomial

Gamma - Poisson

$y \sim \text{Binomial}(n, \theta)$

$$P(y|\theta) = \binom{n}{y} \theta^y (1-\theta)^{n-y}$$

$$\theta \sim \text{Beta}(a, b)$$

$$P(\theta) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1}$$

$$P(\theta|y) \propto P(y|\theta) \cdot P(\theta)$$

$$\propto \theta^{y+a-1} (1-\theta)^{n-y+b-1}$$

$$\theta|y \sim \text{Beta}(y+a-1, n-y+b-1)$$

Informative prior

$$a=5 \quad b=1$$

vs

Non informative prior

$$a=b=1$$

$$a=b=0$$

$$a=b=\frac{1}{2}$$

(Jefferey's prior)

improper distribution

but posterior is good

ML View: minimize loss!

Linear Regression

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

$$\text{Find } \hat{y}(x) = f(x; \hat{w}, \hat{b})$$

Objective:

$$\hat{w}, \hat{b}$$

$$= \arg \min_{w, b} \sum_{i=1}^n (y^i - w^T x^i - b)^2$$

$$= \arg \min_{\tilde{w}} \|Y - \tilde{X} \tilde{w}\|_2^2$$

$$w^* = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T Y$$

Ridge regression

$$\arg \min_{\tilde{w}} \|Y - \tilde{X} \tilde{w}\|_2^2 + \lambda \|\tilde{w}\|_2^2$$

LASSO regression

$$\arg \min_{\tilde{w}} \|Y - \tilde{X} \tilde{w}\|_2^2 + \lambda \|\tilde{w}\|_1$$

Stat View: maximize prob! minimize $-\log \Phi$

MLE

(has to assume a probabilistic model!)

Model:

$$y^i | x^i; w, b = N(w^T x^i + b, \sigma_\epsilon^2)$$

$$P(D_{\text{train}}; w, b) = \prod_{i=1}^n P(y^i | x^i; w, b)$$

$$w^*, b^* = \arg \min_{w, b} -\log P(D_{\text{train}}; w, b)$$

$$= \arg \min_{w, b} \sum_{i=1}^n (y^i - w^T x^i - b)^2$$

MAP

$$\theta = (w, b)$$

$$P(\theta) \propto 1$$

$$w^*, b^* = \arg \max_{\theta}$$

$$P(\theta | D_{\text{train}})$$

$$= \arg \max_{\theta} \prod_{i=1}^n P(y^i | \theta) \cdot P(\theta)$$

how do we get this?



$$\arg \max_{\theta} \prod_{i=1}^n P(y^i | \theta) \cdot P(\theta) = \arg \min_{\theta} -\log \prod_{i=1}^n P(y^i | \theta) P(\theta)$$

$$= \arg \min_{\theta} \sum_{i=1}^n P(y^i | \theta) - \log P(\theta)$$

↓

$$\|Y - \tilde{X} \tilde{w}\|_2^2$$

$$-\log P(\theta); \theta \sim N(0, \tau^2) \rightarrow \frac{1}{\tau^2} \text{ as in normal distribution}$$

$$-\log P(\theta) \propto \lambda \|\theta - 0\|_2^2$$

$$\theta \sim \text{Laplace}(\lambda)$$

$$-\log P(\theta) \propto \lambda \|\theta\|_1$$

2.4 MLE and MAP

Q: What are MLE and MAP? How are they related?

A: MAP weights by the prior, if the prior is “flat” then it has no effect

Q: What effect does $p(x)$ have on linear regression ?

A: None: It is a conditional probability; we aren’t modeling $p(x,y)$, just $p(y|x)$ We don’t care what $p(x)$ is Why is this important? $p(x)$ can be really complex: e.g. x can be an image or a wiki page. It’s much easier to do supervised learning than do generative model.

Q: Will all the weights found by Ridge Regression always be smaller than the weights found by OLS?

A: No – only the L2 norm of the weights must be smaller

Q: What happens to an OLS model if you change one of the features from inches to centimeters.

A: $Y = c_0 + c_1 x(\text{inches}) + c_2 x(\text{dollars})$ $Y = c_0' + c_1' x(\text{cm}) + c_2' x(\text{dollars})$ ANS: x is 2.6 times as big, so c_1' is smaller than c_1

Q: What happens to a Ridge Regression model if you change one of the features from inches to centimeters?

A: $Y = c_0 + c_1 x(\text{inches}) + c_2 x(\text{dollars})$ $Y = c_0' + c_1' x(\text{cm}) + c_2' x(\text{dollars})$ ANS: since c_1' is smaller, it is shrunk less than c_1 We’re made size appear more important

Q: What is a “conjugate prior”?

A: One the you multiply and it gives a distribution of the same form Why are they important? Makes the math pretty? What two examples have we seen? bernoulli/beta, gaussian/gaussian

Q: We shrink weights towards zero in linear regression; are there cases where you might want to shrink towards something else?

A: Yes, shrink to what you think it would be. Shrink to 0.5.

3 POD 3

3.1 Logistic

1. Attendance
2. Questions on the course materials?

3.2 Understand how different regularization penalizes weight

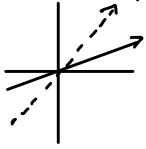
3.3 Bias and Variance

3.4 Streamwise/Stepwise/Stagewise Search

3.5 Different forms of the error

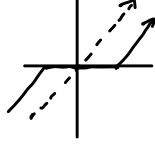
L0, L1, L2 Regularization

L2 Penalty (Ridge/Tikhonov)



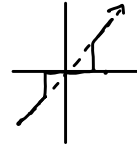
- X scale invariant
- Convex
- Gaussian prior
 $P(w) \sim \exp\left(-\frac{\|w\|_2^2}{\sigma^2}\right)$
- Argmin $\|y - Xw\|_2^2 + \lambda \|w\|_2^2$
- Shrink large weights most heavily
- $(X^T X + \lambda I)^{-1} X^T y$
 non singular

L1 Penalty (Lasso/Laese)



- X scale invariant
- convex
- Laplace Prior
 $P(w) \sim \exp\left(-\frac{\|w\|_1}{\sigma^2}\right)$
- Argmin $\|y - Xw\|_2^2 + \lambda \|w\|_1$

L0 Penalty (stepwise regression)



- scale invariant
- not convex
- spike and slab
- improper prior Dirac Delta
- Argmin $\|y - Xw\|_2^2 + \lambda \|w\|_0$
- Encourage weights to be zero most strongly
- Search.

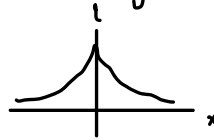
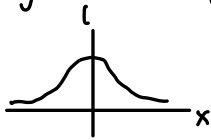
- Gradient descent

L0 & L1 can handle exponentially more features than observations.

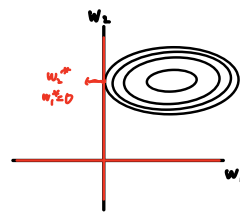
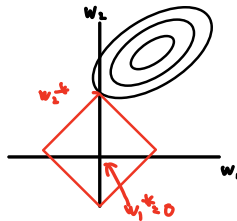
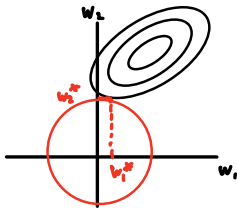
Elastic net combines L2, L1 Reg to both and it's convex!

Why the weight is changing in such way?

Bayesian:



Graph



Math: Assume $X_n = I_n$

$$\|y - Xw\|_2^2 + \lambda \|w\|_2^2 = \|y - w\|_2^2 + \lambda \|w\|_2^2$$

$$w_i^* = \underset{w_i}{\operatorname{argmin}} (y_i - w_i)^2 + \lambda w_i^2$$

$$-2y_i + 2w_i + 2\lambda w_i = 0$$

$$w_i^* = \frac{y_i}{1 + \lambda}$$

$$\|y - w\|_2^2 + \lambda \|w\|_1$$

$$w_i^* = \underset{w_i}{\operatorname{argmin}} (y_i - w_i)^2 + \lambda |w_i|$$

$$w_i^* = \begin{cases} y_i - \lambda & y_i > \lambda \\ 0 & |y_i| < \lambda \\ y_i + \lambda & y_i < -\lambda \end{cases}$$

$$w_i = \begin{cases} w_i & |y_i| > \lambda \\ 0 & |y_i| < \lambda \end{cases}$$

Streamwise/Stepwise/Stagewise Search

Search Strategy

- Streamwise Regression

Try adding features one by one, if adding a feature decreases error of model, keep it, o.w. discard it.

- Stepwise Regression $O(np)$

Repeat p times, each time find the feature that gives lowest error, add it.

- Stagewise Regression

Similar to stepwise regression, at each iteration, keep all w_j from the old model, just regress on residual $r_i = y_i - \sum_j w_j x_{ij}$ on new feature j .

∴ stuck in local optimal if you don't have weak learners, (cover et al.)

Bias and Variance

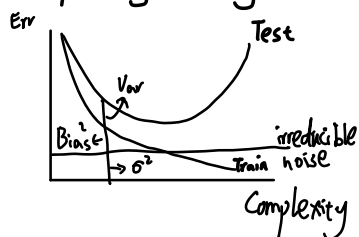
$$\text{Test Error} = \text{Training Error} + \text{Model Complexity} \\ = \text{Variance} + \text{Bias}^2 + \text{Noise}$$

$$E_{x,y,D} [(h(x;D) - y)^2] = E_{x,D} [(h(x;D) - \bar{h}(x))^2] \\ + E_x [(\bar{h}(x) - \bar{y}(x))^2] + E_{x,y} [(\bar{y}(x) - y)^2]$$

$h(x;D)$: model trained on D

$\bar{h}(x)$: expected model trained on n

$\bar{y}(x)$: expected y at every x



4 POD 4

4.1 Mutual Information

4.2 Generalized Linear Models

4.3 Gradient Descent Methods

Q: Can we define clearly the terms such as regularization, shrinkage, strong/weak prior? And where they come in linear regression?

A: Regularize the complexity of the model to deal with overfitting! Shrinkage of weights is the result of regularization at least in linear regression case. Strong/weak prior means more/less informative prior. $\lambda = \frac{\sigma^2}{\gamma^2}$. More informative prior means smaller γ which means larger λ , more regularization!

Read Bishop PRML Section 1.1 and Section 3.1.4. And Andrew Gelman Bayesian Data Analysis Section 2.4 for more information.

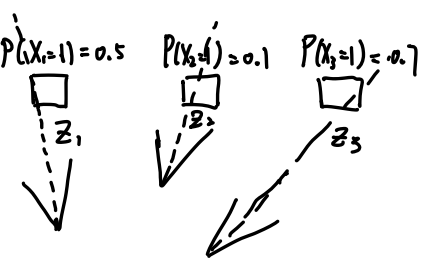
Q: Why is KL divergence not symmetric?

A: $D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$. Because it measures how a distribution differs from the true distribution. It's not the same as how a true distribution differs from a distribution. Which one is true distribution? P is the true distribution and KL divergence is the expectation of difference between the expectation of the logarithmic difference between the probabilities.

Q: Information gain and mutual information and KL divergence?

A: In many cases, such information gain is also called mutual information of X and Y : $I(X, Y) = D_{KL}(P_{(X,Y)}||P_X \cdot P_Y)$ which is calculated differently but the result would be the same. On the Wikipedia page of mutual information, you can find why the two are fundamentally the same.

$$\begin{aligned}
 H(X_i) - H(X_i|Z_i) &= I(X_i; Z_i) = D_{KL}(P(X_i, Z_i) \| P(X_i)P(Z_i)) \\
 &= \sum_{X_i \in \{0,1\}} P(Z_i, X_i) \cdot \log \frac{P(Z_i, X_i)}{P(Z_i)P(X_i)} \\
 &= \sum_{X_i \in \{0,1\}} \int_{\mathcal{Z}} P(X_i=x_i|z_i) \cdot P(z_i) \cdot \log \frac{P(X_i=x_i|z_i)}{P(X_i=x_i)} dz_i \\
 &= \int_{\mathcal{Z}} \sum_{\{0,1\}} \underbrace{\dots}_{\text{prior}} dz_i \cdot P(z_i) \cdot \left(P(X_i=0) \cdot \log \frac{P(X_i=0|z_i)}{P(X_i=0)} + P(X_i=1) \cdot \log \frac{P(X_i=1|z_i)}{P(X_i=1)} \right) \\
 &= \int_{\mathcal{Z}} P(z_i) \left(P(X_i=0) \left(\log \frac{P(X_i=0|z_i)P(X_i=1)}{P(X_i=0)P(X_i=1|z_i)} \right) + \log \frac{P(X_i=1|z_i)}{P(X_i=1)} \right) dz_i \quad \text{inverse sensor model}
 \end{aligned}$$


FSMI
 Zhang dong Zhang

Generalized linear model
 $y(x, w) = \underline{w}^T \cdot \underline{\phi}(x)$

Polynomial $\phi_j(x) = \beta_j x^j$ $\underline{w}^T = (w_0, \dots, w_p)$ $\underline{\phi}^T = (\phi_0, \dots, \phi_p)$

RBF: $\phi_j(x) = \exp\left\{-\frac{(x-\mu)^2}{2s^2}\right\}$

1. $|\underline{\phi}|$
2. size of s (larger, smooth, less complex)
3. means (concentrated / spread)

What controls complexity?

Logistic regression: $y(x) = \log \frac{P(1|x)}{P(0|x)}$

$$P(y=k|x; \theta_1, \dots, \theta_k) = \frac{\exp(\theta_k^T x)}{\sum_l \exp(\theta_l^T x)}$$

Sum of Squared Error (SSE)

$$SSE(\theta) = \frac{1}{n} \sum_{i=1}^n r_i(\theta)^2$$

$$\theta := \theta - \eta \nabla SSE(\theta)$$

$$\nabla SSE(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{d}{d\theta} r_i(\theta)^2$$

Coordinate descent: slow but steady

Repeat:
 For $j=1:P$
 $\theta_j := \theta_j - \eta \frac{dErr}{d\theta_j}$
 (update each feature one by one)

Momentum

$$\Delta w^t = \eta \frac{\delta Err}{\delta w} + m \Delta w^{t-1}$$
 same order of magnitude

Adagrad:

$$\Delta w_j^t = \frac{\eta}{\sqrt{\sum_{s=1}^t \|\delta w_j^s\|_2^2}} \frac{\delta Err}{\delta w_j^t}$$
 past square change

SGD: mini-batch/online

$$SSE(\theta) = \frac{1}{n} \sum_{i=1}^n r_i(\theta)^2 \quad \nabla SSE_k(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{d}{d\theta} r_i(\theta)^2$$

$$\theta := \theta - \eta \nabla SSE_k(\theta)$$

5 POD 5

5.1 Convolutional Neural Network

① Convolution Calculation $y(m,n) = x(m,n) * w(m,n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x(i,j) \cdot w(m-i, n-j)$
 cross-correlation: $y(m,n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x(i,j) \cdot w(m+i, n+j)$

$$\begin{bmatrix} 2 & 0 & -1 \\ 2 & 3 & 0 \\ -1 & 2 & -1 \end{bmatrix} * \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 0 & -1 \\ 2 & 3 & 0 \\ -1 & 2 & -1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 2 & -2 & -1 \\ 4 & 6 & 3 & 1 \\ -2 & 1 & -3 & -1 \\ 0 & 1 & -2 & 1 \end{bmatrix}$$

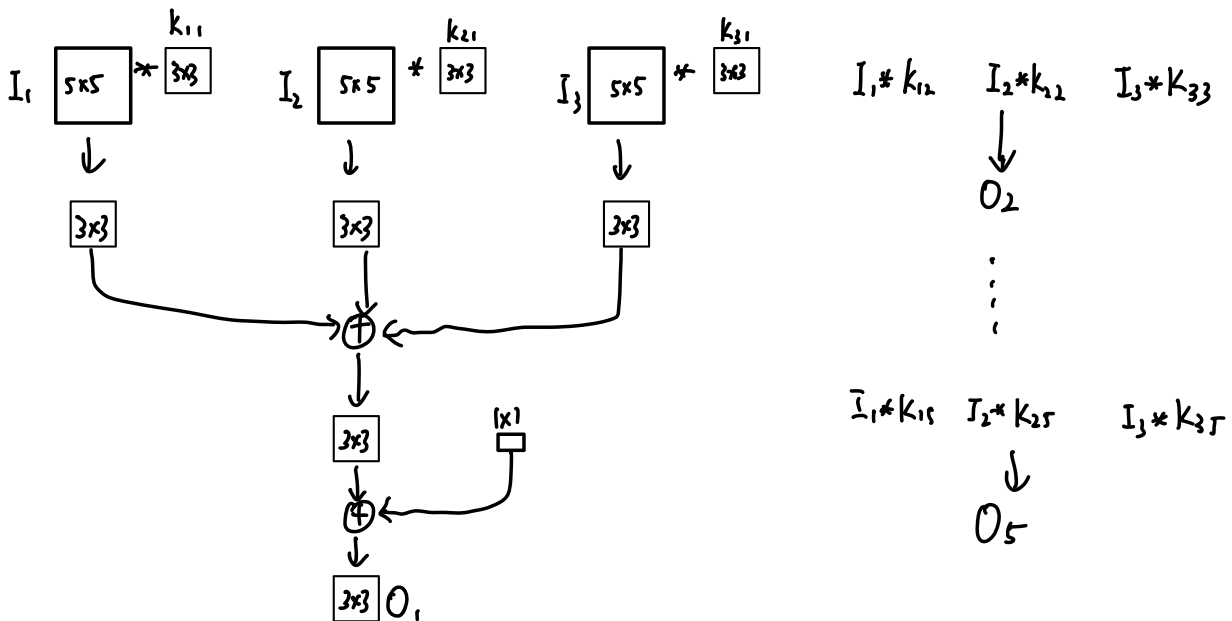
② Convolution Layer: $\mathbb{R}^{w_{in} \times h_{in} \times C_{in}} \rightarrow \mathbb{R}^{w_{out} \times h_{out} \times C_{out}}$ kernel: $w' \times h' \times C_{in} \times C_{out}$

e.g. $w_{in} = h_{in} = 5, C_{in} = 3, I_1, I_2, I_3$

kernel dim: $3 \times 3 \times 3 \times 5, k_{ij}, 1 \leq i \leq 3, 1 \leq j \leq 5, 15 \text{ kernels.}$

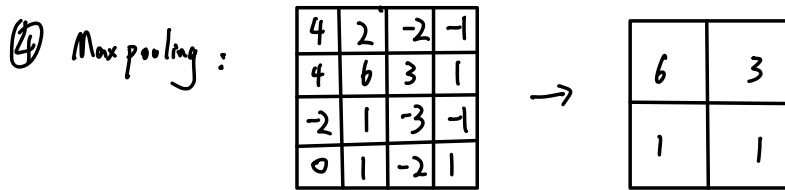
no padding

$w_{out} = h_{out} = 3, C_{out} = 5, O_1, \dots, O_5$



$$O_j = \sum_{i=1}^3 I_i * k_{ij} + b_j$$

③ 1 by 1 filters do dimension reduction



extract most important feature

⑤ dropout: Regularization (setting activation/function to zero)
Backprop not update it (penalize away from local minimum)

⑥ ReLU:
fast: gradient easy to calc

⑦ hyperparameter:

Architecture

LR,

Epoch (early stopping)

⑧ size of minibatch

⑨ weigh initialization

6 POD 8

6.1 SVD, PCA, Autoencoder

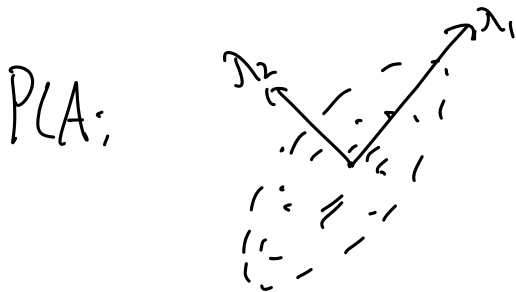
$$w = (X^T X)^{-1} X^T y$$

$$\underbrace{(X^T X)^{-1} X^T X}_I w = (X^T X)^{-1} X^T y$$

$X^T X$ not invertible?

pseudo-inverse

$$X^+ = (U_R \Lambda_R^{-1} V_R^T)^T = V_R \Lambda_R^{-1} U_R^T$$



Through SVD: $X = U S V^T$ I

$$V_R^T \cdot y$$

$\sigma \propto$ variance

maximize variance

$$o = V_R^T \cdot y$$

idea: compress features
into embedding

$$o = \sigma (W \cdot y + b)$$

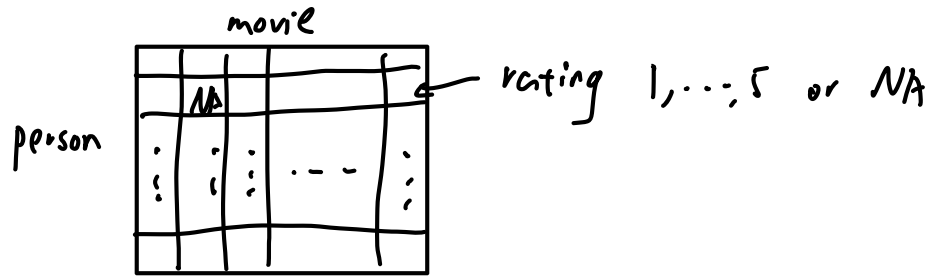
with minimal loss.

$$y = \sigma^{-1}(o)$$

7 POD 9

7.1 Recommend System, Evaluation Metrics

Recommender system



Goal: estimate NA

Matrix factorization

$$R = P \times Q^T \quad R^{m \times n} \quad \begin{matrix} P^{m \times f} \\ Q^{n \times f} \end{matrix}$$

\rightarrow # of latent factors
 \rightarrow movie rating wrt factors \rightarrow person interest in each factor

$$\min \sum_{(u,i) \in K} (r_{ui} - p_u q_i^T)^2$$

too much parameters in P and Q !

regularization

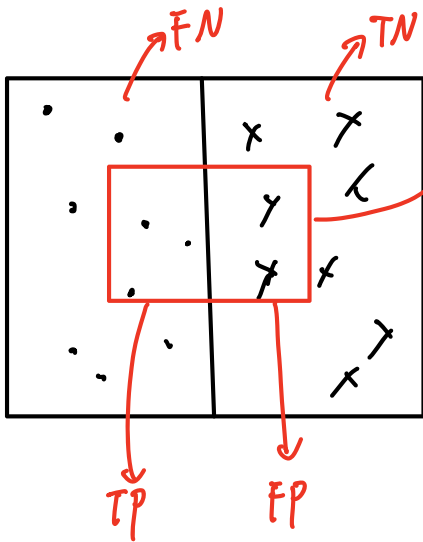
$$\min \sum_{(u,i) \in K} (r_{ui} - p_u q_i^T)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2)$$

How to solve this problem?

$P \times Q^T$ is bilinear, not convex!

- ① - fix P , solve Q
- fix Q , solve P
- repeat

② SGD!



model predict "." for data inside,
predict "x" for data outside

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{FN+TP}$$

$$\text{Specificity} = \frac{TN}{FP+TN}$$

$$\text{Error} = \frac{FP+FN}{FP+FN+TP+TN}$$

$$\text{Accuracy} = 1 - \text{Error}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

compare to Acc, don't care about TN!
solve balance

	True Yes	True No
Classify Yes	TP	FP
Classify No	FN	TN

Logistic regression

$$S = \text{sigmoid}(w^T x) \in (0, 1)$$

$$= "P(y=1 | x)"$$

↳ not actually true

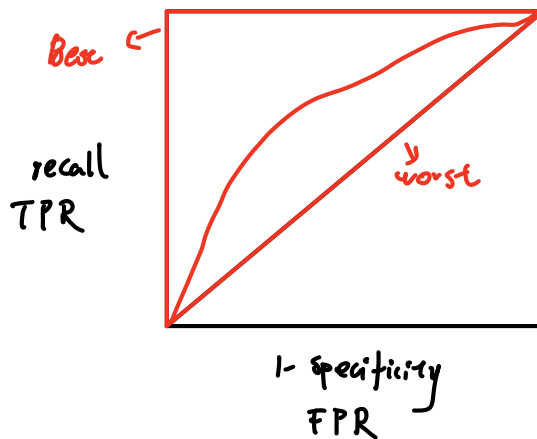
$$\hat{y} = 1 \text{ for } S > \theta \quad \theta \in (0, 1)$$

↳ TPR, FPR

ROC (Receiver Operating Characteristic) Curve

1. rank from high to low of being yes
2. sweep threshold (specificity) from 1 to 0
3. compute recall give threshold
4. plot curve (1-specificity) vs. recall

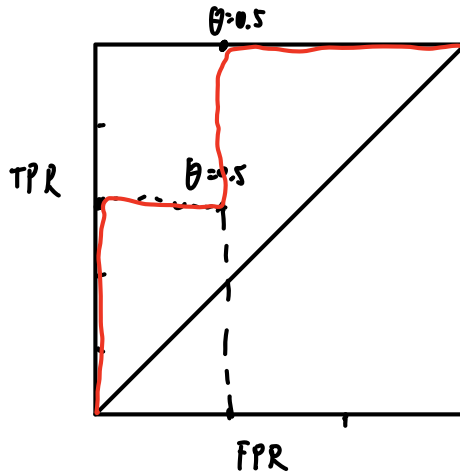
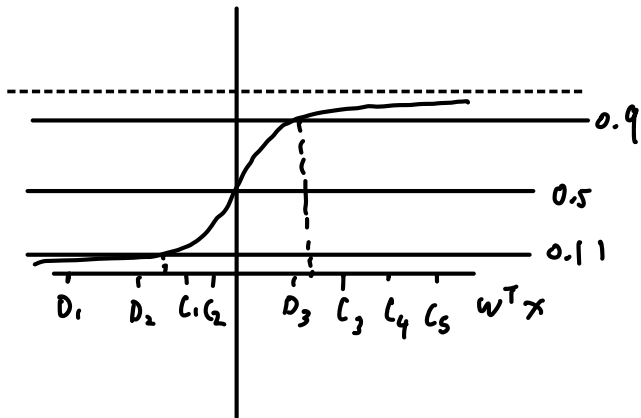
AUC = 0.5 is random guessing
= 1 is perfection



$D: \{D_i\}$

$C: \{C_i\}$

$$Y = C \text{ if } S > \theta$$



$$\theta = 0.5 \quad TPR = \frac{TP}{FN+TP} = \frac{3}{3+5} = \frac{3}{8}$$

$$FPR = 1 - \frac{TN}{FP+TN} = 1 - \frac{2}{1+2} = \frac{1}{3}$$

$$\theta = 0.25 \quad TPR = \frac{5}{0+5} = 1$$

$$FPR = 1 - \frac{2}{1+2} = \frac{1}{3}$$

$$\theta = 0.9 \quad TPR = \frac{3}{1+3} = \frac{3}{4}$$

$$FPR = 1 - \frac{3}{0+3} = 0$$

Larger AUC, better. i.e. higher TPR with lower FPR

8 POD 10

8.1 K-Mean, GMM, EM

Project: what's the data?

what interesting & original things you want to do?

k-means: $J(\mu, r) = \sum_{i=1}^n \sum_{k=1}^K r_{ik} \| \mu_k - X_i \|^2$

$\arg \min_r$ & $\arg \min_{\mu}$ alternatingly until converge.

Greedy Algorithm: rely on good initialization!



0. pick k cluster centroids

1. assign points to cluster to minimize $J(\mu, r)$ given μ EM

2. compute centroid of clusters to minimize $J(\mu, r)$ given r .

$$r_{ik} = \begin{cases} 1 & \text{if } d_i \text{ in cluster } k \\ 0 & \text{o.w.} \end{cases}$$

use alternating gradient descent.

Hierarchical Mode
 $\mu_j \sim N(\mu_j, \sigma^2)$
 $\mu_j \sim N(\mu_j, \tau^2)$
 to pass on global mean (non-terminating, μ_j, σ_j)

Gaussian Mixtures

$$P(x) = \sum_{k=1}^K \pi_k N(\mu_k, \Sigma_k) = \sum_{k=1}^K P(z=k) P(x|z=k)$$

Given $D = \{x_1, \dots, x_n\}$, estimate (π_k, μ_k, Σ_k)

for $k \in [1, K]$ using MLE/MAP

$\log P(D|\pi, \mu, \Sigma)$

-Variance Parameters

1. Full covariance: Σ_k are arbitrary for each class $O(km^2/2)$. clusters are ellipsoids

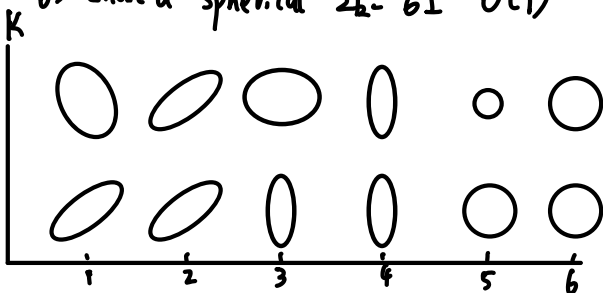
2. Shared full: Σ_k arbitrary but same for each class. $O(m^2/2)$. same shape ellipsoids

3. Diagonal (Naive Bayes) $O(Km)$

4. Shared diagonal: $O(m)$

5. spherical: $\Sigma_k = \sigma_k I$ $O(K)$

6. shared spherical $\Sigma_k = \sigma I$ $O(1)$



Expectation Maximization

EM for GM:

Random initialize μ_1, \dots, μ_k

Alternate till converge

E: estimate $P(z_i=k | x_i)$

M: estimate $\mu_k = \frac{\sum_i P(z_i=k | x_i) x_i}{\sum_i P(z_i=k | x_i)}$

9 POD 11

9.1 Naive Bayes and LDA

Naive Bayes Model

$$P(x, y) = \underset{\text{class prior}}{P(y)} \cdot \underset{\text{class model}}{P(x|y)}$$

$$P(\theta|D) \propto \underset{\text{Parameter prior}}{P(\theta)} \cdot P(D|\theta) = P(\theta, D) \text{ why this is not a generative model?}$$

How to use for prediction?

$$h(x) = \arg \max_y \hat{P}(y|x) = \arg \max_y \hat{P}(y) \hat{P}(x|y) \text{ (No MAP)}$$

estimate separately: $\hat{P}(y), \hat{P}(x|y)$

- Estimating $P(y)$ $\hat{\theta}_y = \frac{1}{n} \sum_i 1(y=y_i)$

- Estimating $P(x|y)$

Assumption: Conditional independence.

$$\hat{P}(x|y) = \prod_j \hat{P}(x_j|y)$$

Is it correct? No, e.g. $x_1 = \text{soccer}$, $x_2 = \text{Messi}$, not independent.

MLE on $P(x_j|y_i) = \frac{\sum_x 1(y=y_i, x=x_j)}{\sum_x 1(y=y_i)}$

MAP on $P(x_j|y_i) = \theta_{ji}$

$$P(\theta_{ji}|D) \propto P(D|\theta_{ji}) \cdot P(\theta_{ji})$$

↑ Binomial
↗ Beta(S, t)

$$\propto \theta_{ji}^{N_T} \cdot (1-\theta_{ji})^{N_F} \cdot \theta_{ji}^{S_T} \cdot (1-\theta_{ji})^{t-T}$$

$$\begin{aligned} \arg \max_{\theta_{ji}} P(\theta_{ji}|D) &= \arg \max_{\theta_{ji}} \theta_{ji}^{N_T+S_T-1} \cdot (1-\theta_{ji})^{N_F+t-T-1} \\ &= \arg \max_{\theta_{ji}} \log \theta_{ji}^{N_T+S_T-1} \cdot (1-\theta_{ji})^{N_F+t-T-1} \\ &= \arg \max_{\theta_{ji}} (N_T+S_T-1) \log \theta_{ji} + (N_F+t-T-1) \log(1-\theta_{ji}) \end{aligned}$$

$$(N_T+S_T-1) \frac{1}{\theta_{ji}} - (N_F+t-T-1) \frac{1}{1-\theta_{ji}} = 0$$

$$(N_T+S_T-1)(1-\theta_{ji}) = (N_F+t-T-1)\theta_{ji}$$

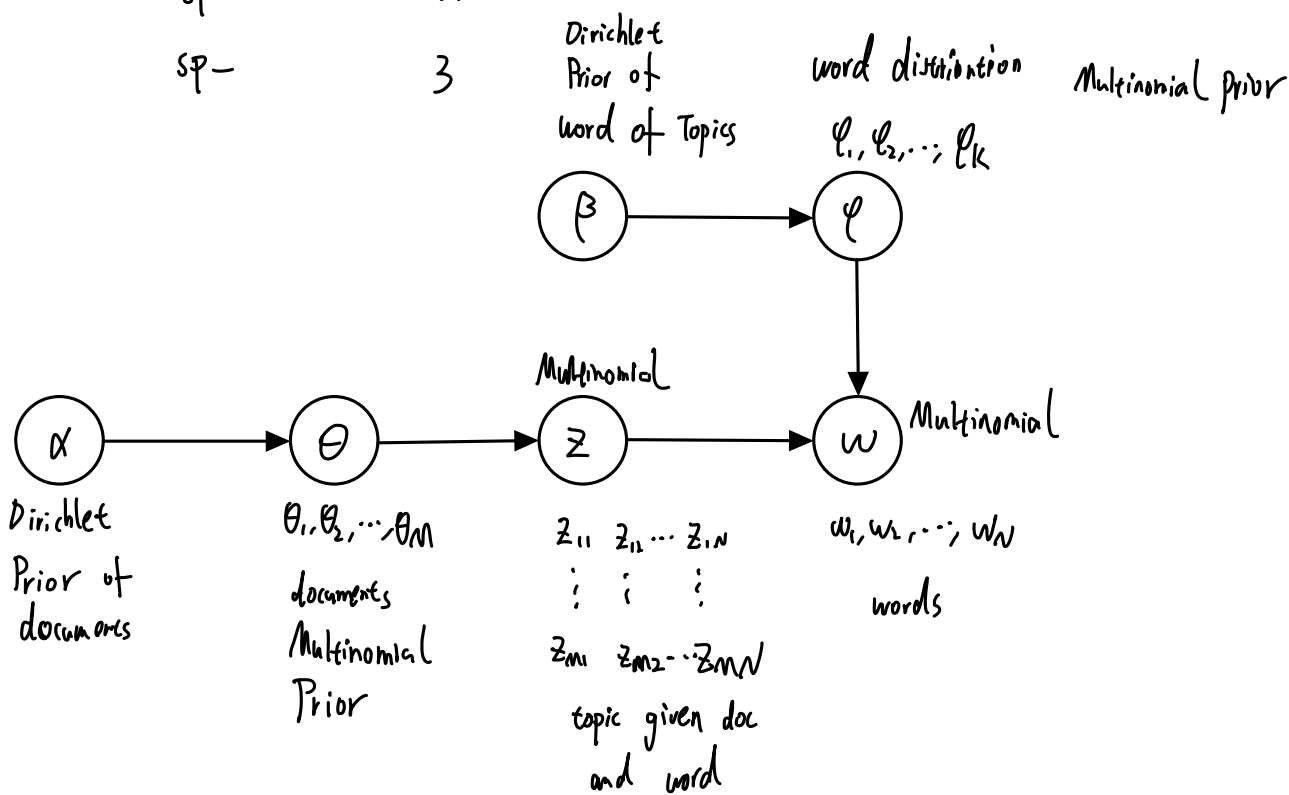
$$\hat{\theta}_{ji} = \frac{N_T+S_T-1}{N_T+N_F+S_T+t-2}$$

Prediction $H(x) = \arg \max_{y_i} \hat{P}(y=y_i|x) = \arg \max_{y_i} \hat{P}(y=y_i) \prod_j \hat{P}(x_j|y_i) = \arg \max_{y_i} \hat{P}(y=y_i) \prod_j \hat{\theta}_{ji}$

Example:

Word	Topic	Count	Class Model
a	sports	0	
ball	sp-	1	$P(a sports) = \frac{0+5-1}{0+10+5+1-2}$
carrot	sp-	0	$P(game sports) = \frac{2+5-1}{2+8+5+1-2}$
game	sp-	2	
I	sp-	2	...
saw	sp-	2	
the	sp-	3	

LDA:



$$P(w, z, \theta, \phi | \alpha, \beta) = \prod_{j=1}^M P(\theta_j | \alpha) \cdot \prod_{t=1}^N P(z_{jt} | \theta_j) \cdot \prod_{i=1}^k P(\phi_i | \beta) \cdot P(w_{jt} | \phi, z_{jt})$$

Dirichlet \leftarrow $\prod_{j=1}^M P(\theta_j | \alpha)$
 Multinomial \leftarrow $\prod_{t=1}^N P(z_{jt} | \theta_j)$
 Multinomial \leftarrow $\prod_{i=1}^k P(\phi_i | \beta)$
 Multinomial \leftarrow $P(w_{jt} | \phi, z_{jt})$

How do we generate a document of text?

- pick a document: $P(\theta = sports | \alpha) = 0.8$ $P(\theta = food | \alpha) = 0.2$
 we random sample and get sports document

2. pick topics of each word: $P(z = \text{Messi} | \theta = \text{sport}) = 0.6$

$$P(z = \text{World Cup} | \theta = \text{sport}) = 0.4$$

we random sample topic for each word:

1: Messi

2: Messi

3: World Cup

4: World Cup

5: Messi

3. gain the word distribution of each topic

$$P(\varphi_{\text{Messi}} = \text{PSG} | \beta) = 0.8$$

$$P(\varphi_{\text{Messi}} = \text{Barcelona} | \beta) = 0.2$$

$$P(\varphi_{\text{World Cup}} = \text{Qatar} | \beta) = 0.5$$

$$P(\varphi_{\text{World Cup}} = \text{Soccer} | \beta) = 0.5$$

4. Generate the word

1: PSG

2: PSG

3: Qatar

4: Soccer

5: Barcelona.

Parameter (W, Z) Estimation:

E: compute $P(W, Z | \theta, \varphi, \alpha, \beta)$ with variational inference

M: estimate α & β given (W, Z) distribution

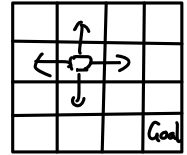
(Variational EM)

10 POD 12

10.1 Reinforcement Learning

RL is MDP: $\langle S, A, P, R, \gamma \rangle$

Car moves in 2D



- S : state
- A : action
- π : policy $S \rightarrow A$ or $S \rightarrow P_r(A)$
- R : reward $S \rightarrow R$ $S \times A \rightarrow R$
- γ : discount factor

(x, y)

move left, right, up, down

- $\pi_L(1,1) = \text{left}$ $\pi_L(1,1)(\text{left}) = 0.9$
- higher reward if $\pi_L(1,1)(\text{right}) = 0.05$
- move closer to goal. $\pi_L(1,1)(\text{up}) = 0.05$
- future value is discounted $\pi_L(1,1)(\text{down}) = 0$

Model: $(S_t, a) \rightarrow S_{t+1}$ (if model based)

Value function: $(s, a) \rightarrow V$

Model & reward: $P(S_{t+1}=s', R_{t+1}=r | S_t=s, A_t=a)$

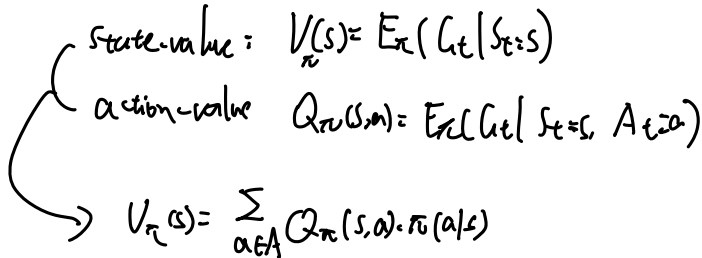
$$P(S_{t+1}=s' | S_t=s, A_t=a) = \sum_{r \in R} P(s', r | s, a)$$

$$R(s, a) = E(R_{t+1} | S_t=s, A_t=a) = \sum_{r \in R} r \sum_{s'} P(s', r | s, a)$$

Policy: deterministic $\pi(s) = a$

stochastic $\pi(a|s) = P_\pi(A_t=a | S_t=s)$

Value function: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$



action advantage function: $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$

Optimal value:

Generalized Policy Iteration (GPI)

$$V_*(s) = \max_{\pi} V_\pi(s)$$

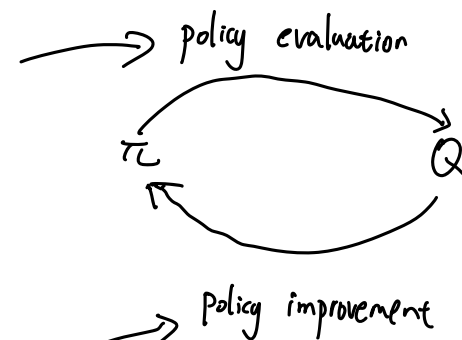
$$Q_*(s, a) = \max_{\pi} Q_\pi(s, a)$$

Optimal policy:

$$\pi_* = \operatorname{argmax}_{\pi} V_\pi(s)$$

$$\pi_* = \operatorname{argmax}_{\pi} Q_\pi(s, a)$$

$$= \text{greedy}(Q)$$



On-Policy vs. Off-Policy

↳ epsilon-greedy (SARSA)

same: on policy
 choose A' from S' using policy from Q (ϵ -greedy)
 $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

Problem: not learning the optimal policy
 $A' \neq \text{greedy}(Q)$

Q-learning

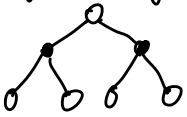
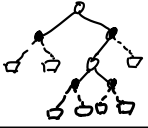

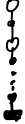
different: off policy
 Choose A from S using policy derived from Q (ϵ -greedy)
 Take action A , observe R, S'
 $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

Discount factor:

$\gamma = 1$ for game like chess

$\gamma < 1$ for survival.

RL Strategies

	One-step ahead	Search to end
Model-based Respond to all possible actions	Dynamic Programming 	Exhaustive Search 
Model-free Respond to one possible action	Temporal-difference 	Monte Carlo 

Monte Carlo: high variance
 low bias

Q-learning (TD): low variance
 high bias

Heine-Borel, Bolzano Weierstrass, and Examples of Convergence of Functions

Siming He, Bruce Lee

September 7, 2024

1 Review of Convergence of Functions

Definition 1 (Pointwise convergence) Let $(f_n)_{n=1}^{\infty}$ be a sequence of functions from one metric space (X, d_X) to another (Y, d_Y) , and let $f : X \rightarrow Y$ be another function. We say that $(f_n)_{n=1}^{\infty}$ converges pointwise to f on X if we have

$$\lim_{n \rightarrow \infty} d_Y(f_n(x), f(x)) = 0.$$

for all $x \in X$. Equivalently, for all $x \in X$ and all $\epsilon > 0$, there exists N such that $d_Y(f_n(x), f(x)) < \epsilon$ for all $n > N$.

Definition 2 (Uniform convergence) Let $(f_n)_{n=1}^{\infty}$ be a sequence of functions from one metric space (X, d_X) to another (Y, d_Y) , and let $f : X \rightarrow Y$ be another function. We say that $(f_n)_{n=1}^{\infty}$ converges uniformly to f on X if for every $\epsilon > 0$ there exists $N > 0$ such that

$$d_Y(f_n(x), f(x)) < \epsilon$$

for all $n > N$ and all $x \in X$. We call the function f the uniform limit of the functions f_n .

2 Examples of Convergence of Functions

Which of the following sequences of functions are pointwise convergent? Uniformly convergent? If they are convergent, then what are their limits?

- $\{f_n\}$, where $f_n : [0, 1] \rightarrow \mathbb{R}$ is defined by $f_n(x) = x^n$.

Answer: $f_n(x)$ converges pointwise to $f(x) = \begin{cases} 0 & \text{if } x \in [0, 1) \\ 1 & \text{if } x = 1 \end{cases}$. It doesn't converge

uniformly which can be proved by contradiction. Assume that $\{f_n\}$ converges uniformly. For any of $\epsilon \in (0, 1/2)$, there exists N such that $|f_n(x) - f(x)| < \epsilon$ for all $x \in [0, 1]$ and all $n > N$. For $x \in [0, 1)$, we have $f_n(x) < \epsilon < 1/2$ since $f(x) = 0$. For $x = 1$, we have $f_n(x) > 1 - \epsilon > 1/2$. It means that f_n is not continuous which contradicts the fact that x^n is continuous.

- $\{f_n\}$, where $f_n : [0, 0.99] \rightarrow \mathbb{R}$ is defined by $f_n(x) = x^n$.
Answer: $f_n(x)$ converges pointwise and uniformly to $f(x) = 0$. To prove that it converges uniformly, we show that for any $\epsilon > 0$, there exists an N such that $|f_n(x) - f(x)| = f_n(x) \leq 0.99^n < \epsilon$ for all $n > N$.
- $\{f_n\}$, where $f_n : \mathbb{R} \rightarrow \mathbb{R}$ is defined by $f_n(x) = x + n$.
Answer: Diverge.
- $\{f_n\}$, where $f_n : \mathbb{R} \rightarrow \mathbb{R}$ is defined by $f_n(x) = \frac{1}{n}$.
Answer: Converge both pointwise and uniformly.

- Let $\Delta(x) = \begin{cases} 0 & x \leq -1 \\ 1+x & -1 \leq x \leq 0 \\ 1-x & 0 \leq x \leq 1 \\ 0 & x \geq 1 \end{cases}$ Consider $\{f_n\}$, where $f_n : \mathbb{R} \rightarrow \mathbb{R}$ is defined by

$$f_n(x) = \Delta(x - n).$$

Answer: Converge pointwise to $f(x) = 0$. It doesn't converge uniformly since there will always be some x with $f_n(x) = 1 > \epsilon$ for all $\epsilon < 1$.

3 Uniform Convergence in Proof of the Independence of Binary Digits

This section is covered on page 147 of ToP.

On page 147, the proof of the independence of binary digits uses the fact that $\sum_{k=1}^n xr_k(t)2^{-k}$ converges uniformly to $x(1 - 2t)$. To show the uniform convergence, we begin by defining $f_n(t) = \sum_{k=1}^n xr_k(t)2^{-k}$ and $f(t) := \sum_{k=1}^{\infty} xr_k(t)2^{-k}$. $|f(t) - f_n(t)| = |\sum_{k=n+1}^{\infty} xr_k(t)2^{-k}| \leq |\sum_{k=n+1}^{\infty} x2^{-k}| \leq |x| \sum_{k=n+1}^{\infty} 2^{-k} = |x|2^{-n}$ which goes to zero as n goes to infinity.

The proof of the independence of binary digits relies on the uniform convergence to interchange the integral and limit, i.e.,

$$\lim_{n \rightarrow \infty} \int_0^1 \exp(if_n(t)) dt = \int_0^1 \lim_{n \rightarrow \infty} \exp(if_n(t)) dt = \int_0^1 \exp(if(t)) dt$$

To prove the interchange is permissible, we want to prove that $|\lim_{n \rightarrow \infty} \int_0^1 \exp(if_n(t)) dt - \int_0^1 \exp(if(t)) dt| = 0$. The proof eventually comes to

$$\left| \lim_{n \rightarrow \infty} \int_0^1 \exp(if_n(t)) dt - \int_0^1 \exp(if(t)) dt \right| < \lim_{n \rightarrow \infty} \int_0^1 |f_n(t) - f(t)| dt$$

By uniform convergence, we know that for any $\epsilon > 0$, there exists N such that $|f_n(t) - f(t)| < \epsilon$ for all $n > N$ and $t \in [0, 1]$. It implies that $\int_0^1 |f_n(t) - f(t)| dt < \epsilon$ for any ϵ .

4 Heine Borel Theorem

A **cover** of a subset \mathbb{I} on the real line is a collection $\{\mathbb{U}_n, n \geq 1\}$ of sets whose union contains \mathbb{I} . It is an **open cover** if each \mathbb{U}_n is an open interval (a_n, b_n) . A **subcover** is a subcollection whose union also contains \mathbb{I} . A **finite subcover** contains only a finite number of sets. If every open cover of a set \mathbb{I} has a finite subcover, we say that \mathbb{I} is **compact**.

Theorem 1 Heine-Borel Theorem: *Suppose \mathbb{I} is a closed and bounded interval. Then \mathbb{I} is compact, i.e. every open cover of \mathbb{I} has a finite subcover.*

Proof sketch: Let $\mathbb{I} = [a, b]$, and $\{\mathbb{U}_n\}$ be an open cover for \mathbb{I} . Let

$$\mathbb{A} = \{x \in \mathbb{I} \mid [a, x] \text{ can be covered by a finite subcollection of } \{\mathbb{U}_n \mid n \geq 1\}\}$$

Note that \mathbb{A} is nonempty, and bounded above and below by a and b respectively. Then the supremum $c = \sup \mathbb{A} \in [a, b]$ exists. To prove that \mathbb{I} has a finite subcover, we want to show 1) $c \in \mathbb{A}$ and 2) $c = b$.

Since c is in \mathbb{I} , it's in some open interval $\mathbb{U}_i = (c - \epsilon, c + \delta) \in \{\mathbb{U}_n\}$. Since c is supremum of \mathbb{A} , we know there exists $x \in (c - \epsilon, c)$ such that $x \in \mathbb{A}$. Let denote the finite cover of $[a, x]$ as $\mathbb{U}_1, \dots, \mathbb{U}_N$. Then, it's clear that $\mathbb{U}_1, \dots, \mathbb{U}_N, \mathbb{U}_i$ is a finite cover of $[a, c]$, i.e., $c \in \mathbb{A}$.

We assume for contradiction that $c \neq b$. Then, there exists an element $y \in (c, \min(c + \delta, b))$. Then, $\mathbb{U}_1, \dots, \mathbb{U}_N, \mathbb{U}_i$ is also a finite cover of $[a, y]$, i.e. $y \in \mathbb{A}$. Since $y > c$, it contradicts with the condition that c is the supremum of \mathbb{A} . ■

Example 1 *Let $I = [0, 1]$ be a closed and bounded interval in \mathbb{R} .*

1. *Consider the countable collection of open intervals $\{A_n\}_{n=1}^{\infty}$, where $A_n = (\frac{1}{n+1} - 0.1, \frac{1}{n} + 0.1)$ for each natural number $n \in \mathbb{N}$. Show that $\{A_n\}_{n=1}^{\infty}$ forms an open cover of the interval $[0, 1]$.*

2. *Find a finite subcover $\{A_{n_1}, A_{n_2}, \dots, A_{n_k}\}$ that still covers the interval $[0, 1]$.*

Example 2 *Let $I = (0, 1)$.*

1. *Is $\{A_n\}_{n=1}^{\infty}$, where $A_n = (\frac{1}{n}, 1)$ for each $n \in \mathbb{N}$ an open cover of $(0, 1)$? **Answer:** Yes.*

2. *Is there a finite subcover? **Answer:** No.*

Example 3 *Consider the set $[0, \infty)$. Is this set closed? Is this set bounded? Now consider the open cover $\{\mathbb{U}_n, n \geq 1\}$ where $\mathbb{U}_n = (-0.01, n)$. Justify that this is a cover. Does there exist a finite sub-cover?*

5 Bolzano-Weierstrass Theorem

Theorem 2 Bolzano-Weierstrass Theorem *Every bounded sequence has a convergent subsequence.*

Proof sketch: Suppose that $\{x_n\}$ is a bounded sequence. Then there exist some $M > 0$ such that $|x_n| \leq M/2$ for all n . Then at least one of the subintervals $[-M/2, 0]$ and $[0, M/2]$ contains an infinite number of members of the sequence. These members form a subsequence $\{x_{1n} | n \geq 1\}$. This is now a bounded sequence, so we may continue to bisect the interval and find a subsequence contained in one of the halves, labeling the j th such subsequence $\{x_{jn}\}$. The j th subsequence will be in an interval of length $\frac{M}{2^j}$. Then if we define the subsequence $\{y_n\}$ by $y_n = x_{nn}$, we see that $\{y_n\}$ is a subsequence of $\{x_n\}$ and that $\{y_n\}$ is a Cauchy sequence: given any $\epsilon > 0$, there exists N such that $|y_n - y_m| \leq \epsilon$ for $n, m \geq N$. In particular, if we set $N = \log_2(M/\epsilon)$, then for $n, m \geq N$, y_n and y_m are contained in an interval of length $M/2^{\log_2(M/\epsilon)} = \epsilon$. The sequence $\{y_n\}$ is thus convergent by the completeness of the real numbers.

Example 4 Let $\{x_n\}$ be the sequence defined by $x_n = (-1)^n + \frac{1}{n}$. Show that the sequence $\{x_n\}$ has a convergent subsequence and find the limit of that subsequence.

Answer: It has a convergence subsequence since it's bounded between -2 and $+2$. The subsequence $\{x_{2k}\}$ converges to 1 . The subsequence $\{x_{2k+1}\}$ converges to -1 .

More complete proofs of both Heine-Borel and Bolzano-Weierstrass may be found on page 776 of the text.

Notation: $\{I_i\}$ be half closed, disjoint intervals.

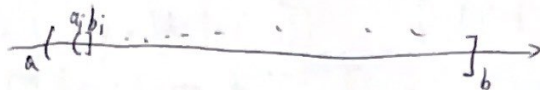
Simon

$$I = \bigcup_{i=1}^{\infty} I_i, \quad I_i = (a_i, b_i], \quad I = (a, b]$$

WTS $F(I) = \sum_i F(I_i)$ Countable Additivity (TOP 382 Lemma 5, part 1)

Typical proof procedure [?] ① showing $\sum_i F(I_i) \leq F(I)$. trivial as in TOP 382.

② showing $\sum_i F(I_i) \geq F(I)$.



we have countably infinite $(a_i, b_i]$, and we don't know how to work with it.

WE KNOW "Finite Additivity".

Can we somehow convert the problem so we can exploit finiteness [?]

Heine-Borel Theorem!

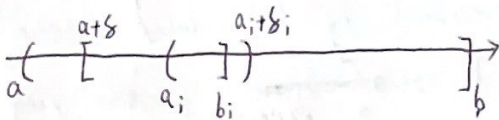
If I^* is compact, every open cover of I^* has finite subcover.

The big idea of the proof:

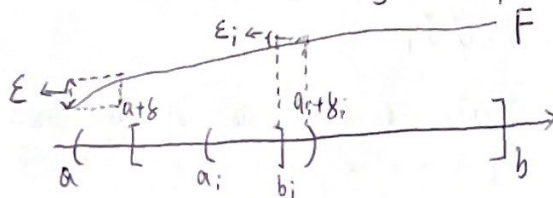
① making I closed and bounded



② making I_i open



③ show the impacts of making I compact and I_i open are arbitrarily small.



δ and δ_i can be arbitrarily small as δ, δ_i are arbitrarily small due to [?] right continuity

④ By Heine-Borel Theorem, there is a finite subcover. Then apply finite subadditivity

Formally: ① $(a+\delta, b] \subset [a+\delta, b] \subseteq \bigcup_{i=1}^n (a_i, b_i + \delta_i) \subset \bigcup_{i=1}^n [a_i, b_i + \delta_i]$.
□ Heine-Borel.

② $F(a+\delta, b] \leq F(\bigcup_{i=1}^n (a_i, b_i + \delta_i))$ □ monotonicity
 $\leq \sum_{i=1}^n F(a_i, b_i + \delta_i)$ □ subadditivity (finite)
 ~~$\leq \sum_{i=1}^{\infty} F(a_i, b_i + \delta_i)$ □ positivity~~ } why not additivity? □
we expand the set and may not be disjoint.
 $= \sum_{i=1}^n F(a_i, b_i] + F(b_i, b_i + \delta_i)$ □ additivity
 $\leq \sum_{i=1}^n F(a_i, b_i] + \sum_{i=1}^n \epsilon_i$ □ positivity

③ $F(a, b] = \underbrace{F(a, a+\delta]}_{\epsilon} + F(a+\delta, b]$ $\sum_{i=1}^n \epsilon_i$
 $\leq \sum_{i=1}^n F(a_i, b_i] + \epsilon + \sum_{i=1}^n \epsilon_i$

Since $\epsilon + \sum_{i=1}^n \epsilon_i$ can be arbitrarily small,

$$F(a, b] \leq \sum_{i=1}^{\infty} F(a_i, b_i]. \quad \square$$

Now we show countable additivity of disjoint half close intervals.

We want to extend it to the ring generated by half-closed intervals. (Part 2 Top 383)

Denote J halfclosed interval and I ~~see a element~~ a element of the ring.

$$\begin{aligned} F(I) &= \sum_k F(J_k) \quad \square \text{ finite additivity; } I = \bigcup_k J_k \text{ by def. of ring.} \\ &= \sum_k F\left[\bigcup_i (I \cap J_k)\right] \quad \square I = \bigcup_i I_i \\ &= \sum_k \sum_i F(I \cap J_k) \quad \square I \cap J_k \text{ is a finite union of half closed intervals.} \\ &\quad \text{By countable additivity of half close interval} \\ &= \sum_i \sum_k F(I \cap J_k) \\ &= \sum_i F(I_i) \quad \square \text{ by finite additivity.} \end{aligned}$$

Disjoint...

MCT Proof

$$0 \leq X_n \uparrow X \Rightarrow \lim_{n \rightarrow \infty} E(X_n) = E(X)$$

$$\textcircled{1} \lim_{n \rightarrow \infty} E(X_n) \leq E(X)$$

for any X_n dominated by X ,

we can find a simple function S dominated by X_n . S is also dominated by X .

$$E(X) = \sup_{X_n \leq S \leq X} (E(S))$$

$$= E(X)$$



$$\textcircled{2} \lim_{n \rightarrow \infty} E(X_n) \geq E(X)$$

Let's pick a simple function S dominated by X .

$$\text{i.e. } S(\omega) \leq X(\omega)$$

$$\text{For any } \varepsilon > 0, (1-\varepsilon)S(\omega) < X(\omega)$$

There always exists a X_n that dominates $(1-\varepsilon)S(\omega)$ and

$$\text{is dominated by } X: \text{not } \exists \delta \text{ s.t. } \inf \{ (1-\varepsilon)S(\omega) - X(\omega) \} \geq \delta$$

$$\text{Since } X_n \uparrow X, \exists N \text{ s.t. } \forall n > N \sup \{ X_n(\omega) - X(\omega) \} \leq \delta$$



$$\text{We now have } \lim_{n \rightarrow \infty} E(X_n) \geq E((1-\varepsilon)S) \geq (1-\varepsilon)E(S)$$

Since ε can be arbitrarily small, $E(X_n) \geq E(S)$ $\forall S$ simple and $S \leq X$.

$$\lim_{n \rightarrow \infty} E(X_n) \geq \sup \{ E(S) : S \text{ is simple, } S \leq X \} = E(X)$$

$$\lim_{n \rightarrow \infty} E(X_n) = E(X). \quad \square$$

Information Theory Recitation

Siming He, ESE 5460

November 2023

Contents

1	Elements of Information Theory	1
2	Information Bottleneck in Deep Learning	5
2.1	Optimization Phases of Neural Network	7
2.2	Input Compression Bound	7
2.3	$I(T, Y)$ and Performance	9
3	References	9

1 Elements of Information Theory

Information theory was developed as a mathematical theory of communication. A communication system can be divided into five parts:

source \rightarrow **transmitter** \rightarrow **channel** \rightarrow **receiver** \rightarrow **destination**

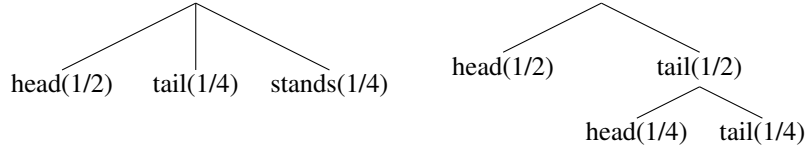
1. **Source** is the message that we want to send to the destination, e.g. text messages, images, videos
2. **Transmitter** converts the message into some signals that can be transmitted. For example, Morse code converts numbers and English alphabets into dots and dashes. Dots and dashes signals can be easily transmitted by short-duration light/sound and long-duration light/sound.
3. **Channel** is the medium used to transmit signal. For example, air transmits light/sound. And wires transmits pulses of voltages.
4. **Receiver** performs an inverse operation of transmitter to reconstruct messages from signals.
5. **Destination** would be the person who receives and reads your text.

Two fundamental questions that are essential to communication system and can be answered by information theory.

1. **What is the maximal compression of data?** We want the communication system to be energy-efficient, i.e., spend the minimum amount of energy to transmit messages. To achieve this, we want to (maximally and losslessly) **compress** our messages.
2. **What is the maximum rate that data can be transmitted?** The maximally compressed message would work well if the channel is noiseless. However, if the channel is noisy, compressed messages may be easily corrupted by the noise during transmission. To make the signal robust to noise, we also want to add **redundancy** into the signal. So the maximum rate of transmission depends on both compression and necessary redundancy.

To understand the design of transmitter or encoder, we have to understand the information in the source or input. However, the concept of information is vague and broad and is hard to be formulated mathematically. Instead, we can define a measure of the amount of information in a random variable. For a probability distribution $P^{(n)} = (p_1, \dots, p_n)$ over an alphabet of size n , we want to find an information measure $H_n(p_1, \dots, p_n)$ characterising the information content of chance experiment with $P^{(n)}$. Shannon argued that there are three required properties of desirable information measure:

1. $H_n(p_1, \dots, p_n)$ is a continuous function of p_1, \dots, p_n . This is intuitive since a small change in the probability distribution shouldn't lead to a sudden jump in information content.
2. For $H_n(p_1, \dots, p_n)$ where $p_1 = \dots = p_n$, H_n should be a monotonic increasing function of n . Intuitively, more events leads to more uncertainty and more information. Rolling a dice seems to be more complex than flipping a coin.
3. The measure should have $H_n(p_1, \dots, p_n) = H_{n-1}(p_1 + p_2, p_3, \dots, p_n) + (p_1 + p_2)H_2(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2})$. For example, $H_3(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ should be equal to $H_2(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H_2(\frac{1}{2}, \frac{1}{2})$. Intuitively, if the information measure measures the amount of information in a chance outcome, how the source produces such outcome shouldn't change the amount of information. For example, the source can flip a magical coin that is head with probability $\frac{1}{2}$, is tail with probability $\frac{1}{4}$, stands up with probability $\frac{1}{4}$. The source can also flip a fair coin and flip the coin again if the first flip results in tail. Those two process gives the same chance outcome so should contain the same amount of information. For the second process, we firstly gain the information of $H_2(\frac{1}{2}, \frac{1}{2})$. Then with probability $\frac{1}{2}$, we gain additional information $H_2(\frac{1}{2}, \frac{1}{2})$ from the second round.



Definition 1.1 (Entropy). The only H satisfying the three properties above is in the form of $H = -K \sum_{i=1}^n p_i \log p_i$. K is basically the unit of the measure and can be 1. Then, we have the definition of entropy:

$$H_n(p_1, \dots, p_n) = - \sum_{i=1}^n p_i \log p_i$$

For a random variable X , we can also write the entropy as

$$H(X) = - \sum_x p(x) \log p(x)$$

Remark. The quantity $-\log p(x)$ can be viewed as the information content of event x . The quantity aligns with our intuition that rare events should contain more information $-\log p(x) > -\log p(y)$ when $p(x) < p(y)$. Entropy can be viewed as the expected information content from a random variable.

Remark. Another view of entropy is that entropy of a random variable is a lower bound on the average length of the shortest description of the random variable. For each event x , we can encode it by l_x bits. Then, the expected length of the description of X is $\sum_x p(x) l_x$. Gibbs' inequality shows that when $l_x = -\log_2 p(x)$, we have the shortest expected length. Specifically, the inequality states $-\sum_x p(x) \log p(x) \leq -\sum_x p(x) \log q(x)$ for any description with length $l_x = -\log q(x)$. It's reasonable to use the shortest expected length of description as a measure of information, since a process with more information inevitably requires more bits to describe.

Definition 1.2 (Joint Entropy and Conditional Entropy). The joint entropy of a pair of discrete variables (X, Y) with joint distribution $p(x, y)$ is defined as

$$H(X, Y) = - \sum_x \sum_y p(x, y) \log p(x, y)$$

The conditional entropy $H(y|X)$ is then defined as

$$\begin{aligned} H(Y|X) &= - \sum_x p(x) \sum_y p(y|x) \log p(y|x) \\ &= \sum_x p(x) H(Y|X = x) \end{aligned}$$

Following the chain rule of probability, we also have the chain rule for entropy:

Theorem 1 (Chain Rule).

$$H(X, Y) = H(X) + H(Y|X)$$

Definition 1.3 (Relative Entropy / Kullback–Leibler(KL) Divergence). The relative entropy between two probability mass functions $p(x)$ and $q(x)$ is

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

KL Divergence measures the dissimilarity between two probability mass functions. However, KL Divergence is not a distance measure since it's not symmetric.

Theorem 2.

$$D(p||q) \geq 0$$

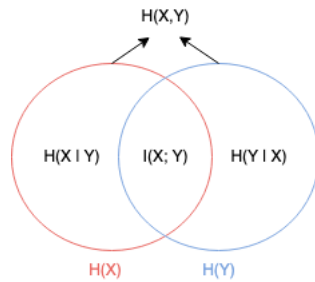
and

$$D(p||q) = 0 \text{ if and only if } p = q$$

Definition 1.4 (Mutual Information). The mutual information between two random variables X, Y is

$$\begin{aligned} I(X; Y) &= \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= D(p(x, y)||p(x)p(y)) \\ &= H(X) + H(Y) - H(X, Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

Mutual information measures the amount of information that one variable contains about another. This measure is symmetric.



This image demonstrates the relationship between mutual information and entropy. $H(X, Y)$ can be viewed as the information contained in joint (X, Y) . $H(X, Y) \leq H(X) + H(Y)$ since X and Y may have some shared information $I(X, Y)$. $H(X|Y)$ measures the amount of additional information we get from X if we already know Y , so we need to remove the amount of shared information: $H(X|Y) = H(X) - I(X; Y)$.

Theorem 3.

$$I(X;Y) \geq 0$$

and

$$I(X;Y) = 0 \text{ if and only if } X \text{ and } Y \text{ are independent}$$

Theorem 4 (Data-processing Inequality). *If we have of Markov chain of three random variables $X \rightarrow Y \rightarrow Z$, i.e., Z is conditionally independent from X given Y , then*

$$I(X;Y) \geq I(X;Z)$$

Remark. This is a really cool theorem! There is no data processing $Y \rightarrow Z$ such that we can get more information about X . Consider image classification as an example. Let X be the label of images. Let Y be images generated from the labels. And let f be a neural network converting Y into some embedding $Z = f(Y)$. We have a Markov chain $X \rightarrow Y \rightarrow f(Y)$. And based on this theorem, we know that $I(X;Y) \geq I(X;f(Y))$. It means that neural networks are not processing images to gain more information about Y ! So what are neural network actually doing from an information-theoretic view? We will soon talk about information bottleneck which answers the question to some extent.

Definition 1.5 (Sufficient Statistics). Given a family of probability mass functions $\{f_\theta(x)\}$. A statistic $T(X)$ is a function of the sample X from $\{f_\theta(x)\}$, e.g., sample mean or sample variance. We have a Markov chain $\theta \rightarrow X \rightarrow T(X)$. By Data-processing Inequality, we have

$$I(\theta, T(X)) \leq I(\theta, X)$$

$T(X)$ is called sufficient statistic if it contains all the information that X has about θ , i.e.,

$$I(\theta, T(X)) = I(\theta, X)$$

Remark. Taking the same example of classification. If the neural network f is a sufficient statistic, then we would have $I(X;Y) = I(X;f(Y))$. It means the embedding $f(Y)$ and the original image Y contain the same amount of information about label X .

2 Information Bottleneck in Deep Learning

In a typical supervised learning problem, we use input data from X to predict output label from Y . Ideally, we want the neural network to learn some embedding \hat{X} such that all the information relevant to Y is kept and other information is discarded. We consider the Markov chain $Y \rightarrow X \rightarrow \hat{X}$. To discard

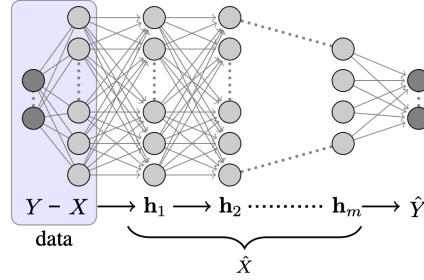


Figure 1: Source: Deep Learning and the Information Bottleneck Principle

irrelevant information, we want to maximally compress X so that only information about Y is kept. However, for a lossy compression, by Data-processing Inequality, $I(Y, \hat{X}) \leq I(Y, X)$. It means that \hat{X} will lose information about Y . Obviously, there is a trade-off between compressing the representation and preserving meaningful information about Y . Information bottleneck method is a principle designed to extract relevant information in X about Y . It's called "information bottleneck" because we are passing the information in X about Y through a "bottleneck" (e.g., a neural network) to form a more compact representation \hat{X} .

Definition 2.1 (Information Bottleneck Principle). To find an optimal representation \hat{X} , we want to minimize the functional

$$I(\hat{X}, X) - \beta I(\hat{X}, Y)$$

Remark. 1. Minimizing $I(\hat{X}, X)$ is maximizing compression. Without compression (\hat{X} has the same information as X), we have $I(\hat{X}, X) = I(X, X) = H(X)$. With maximal compression (\hat{X} only contains information of Y), we have $I(\hat{X}, X) = I(Y, X)$. And $I(Y, X) = H(X) - H(X|Y) \leq H(X)$.

2. Maximizing $I(\hat{X}, Y)$ is maximizing the relevant information in \hat{X} about Y . This is obvious from the definition of mutual information.
3. β is the Lagrange multiplier. If $\beta = 0$, X will be maximally compressed at the risk of losing a lot of relevant information about Y . If $\beta \rightarrow \infty$, \hat{X} would be a super detailed representation of X without any compression.

Now, if we consider an actual neural network as follow: This network forms the Markov chain $Y \rightarrow X \rightarrow h_1 \rightarrow \dots \rightarrow h_m \rightarrow \hat{Y}$. By Data-processing inequality, we have

$$H(X) = I(X, X) \geq I(X, h_1) \geq \dots \geq I(X, h_m) \geq I(X, \hat{Y})$$

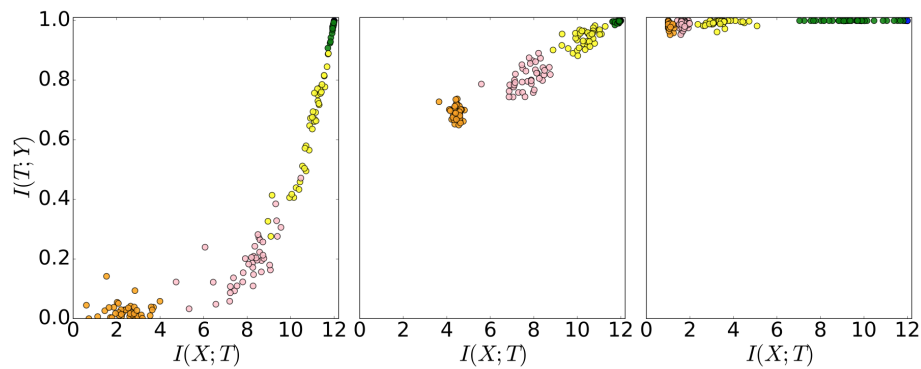
and

$$I(Y, X) \geq I(Y, h_1) \geq \dots \geq I(Y, h_m) \geq I(Y, \hat{Y})$$

By Information Bottleneck Principle, we want to maximize $I(Y, \hat{Y})$ while minimizing $I(X, \hat{Y})$. For each layer, we want to maximize $I(Y, h_i)$ while minimizing $I(h_{i-1}, h_i)$.

Theorem 5 (Information Plane). *Most supervise learning algorithms try to balance the trade-off between the performance on sampled data and the generalization to unseen data. Naftali Tishby argues that it's reasonable to consider $I(Y, T)$ as a measure of performance and consider $I(X, T)$ as a control of generalization.*

2.1 Optimization Phases of Neural Network



Source: OPENING THE BLACK BOX OF DEEP NEURAL NETWORKS VIA INFORMATION.

2.2 Input Compression Bound

Definition 2.2 (Vapnik-Chernovenkis (VC) bound). As learned in lecture, the bound is

$$R(f_{ERM}) \leq R(f^*) + 2\sqrt{\frac{1}{2n} \log \frac{4|\mathcal{F}|}{\delta}}$$

where n is the number of samples, δ is the confidence, and $|\mathcal{F}|$ is the size of the hypothesis class. This bound becomes vacuous for neural networks which have super large $|\mathcal{F}|$.

Definition 2.3 (Input Compression Bound). We want to get a bound that depends on the information of input instead of the large hypothesis class. The bound we will get is

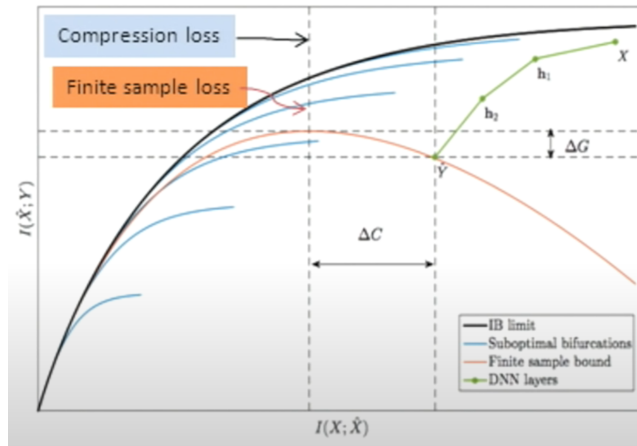
$$R(f_{ERM}) \leq R(f^*) + 2\sqrt{\frac{1}{2n} \left(2^{I(T_\epsilon, X)} + \log \frac{1}{\delta} \right)}$$

where T_ϵ is a ϵ partition of X . T_ϵ can be considered as embedding space where each X is compressed into some code. Note that if we can reduce $I(T_\epsilon, X)$ by k (k more bits of compression), we can reduce the training samples by a factor of 2^k !



This figure above shows the partition of the input space. We partition the input space X into small red balls b_1, b_2, \dots . Each red ball can either be 0 or 1. Under this partition, each $x \in X$ has a more compact binary representation, e.g., $b_1 = 0, b_2 = 1, b_3 = 1, \dots$. We can think of supervise learning as creating a good partition of X that aligns with the labels Y . The typical set of X has size $2^{H(X)}$ (you can read more in Elements of Information Theory 3.29). Similarly, each red ball has size $2^{H(X|T_\epsilon)}$. Then the size of T_ϵ (the number of red balls) is $|T_\epsilon| = \frac{2^{H(X)}}{2^{H(X|T_\epsilon)}} = 2^{I(X, T_\epsilon)}$. If we consider the hypothesis class of Boolean function that maps x to the binary representation $b_1 = 0, b_2 = 1, \dots, b_{|T_\epsilon|}$, this class has size $2^{2^{I(X, T_\epsilon)}}$. Then, we can get the input compression bound from $\log 2^{2^{I(X, T_\epsilon)}}$.

2.3 $I(T, Y)$ and Performance



Source

If we know X, Y , we will have optimal lossy compression which is the black line. Then, the model's performance is subject to the compression loss. Since we only have finite data, the compression is worse and has the red line as the lower bound. The model's performance is subject to an additional finite sample loss.

3 References

1. A Mathematical Theory of Communication
2. Elements of Information Theory
3. Deep Learning and the Information Bottleneck Principle
4. Opening the black box of Deep Neural Networks via Information
5. Learning and generalization with the information bottleneck
6. A great unpublished note from Santosh Venkatesh on An Axiomatic Approach to Information

AWS Tutorial

ESE 5460

October 2023

Contents

1 Overview	1
2 Why use the Cloud?	2
3 Creating an Account	3
4 Launching a GPU instance	3
4.1 Use Penn Campus Network or University VPN	3
4.2 Switch your zone to Virginia (U.S East) on the top right of the screen.	3
4.3 Name and Tags	3
4.4 Select Amazon Machine Image (AMI)	4
4.5 Choose Instance Type	5
4.6 Create a keypair from the EC2 console	5
4.7 Security Groups	6
4.8 Launch the Instance	6
4.9 Stop and Start Instance	6
5 Using Your Instance	7
5.1 Use Your Instance in VS Code	7
5.2 Check the deep learning environment	7
5.3 SCP: Secure Copy for File Transferring	7
5.4 tmux	8
5.5 Use Your Instance in Jupyter Notebook	8

1 Overview

The EC2 (Elastic Cloud Compute) service on AWS (<https://aws.amazon.com>) offers cloud computing infrastructure and provides access to GPU resources. Each student will be given compute credits to use on AWS. AWS could be used for the programming assignments and is a useful tool for the projects. EC2 gives you access to GPU instances which are charged by the hour. In this short tutorial, we will go over the basics, best practices and some useful tools

2 Why use the Cloud?

- Use latest GPU architectures
- Easier to scale
- No maintenance costs

3 Creating an Account

You will get an email titled “AWS @ Penn access” from `isc-cloud-solutions@isc.upenn.edu` with instructions on how to create your account and use the AWS credits allocated to ESE 546.

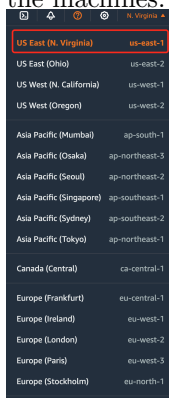
4 Launching a GPU instance

4.1 Use Penn Campus Network or University VPN

[VPN Link](#)

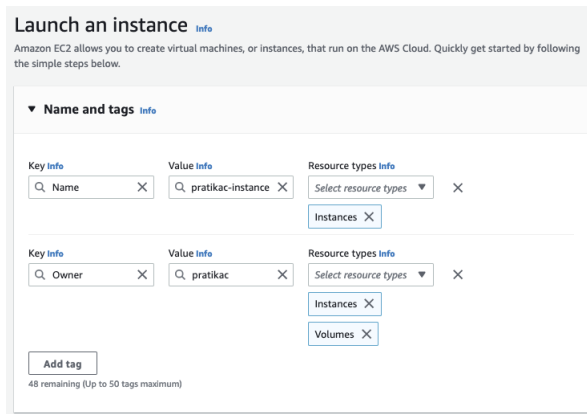
4.2 Switch your zone to Virginia (U.S East) on the top right of the screen.

We are going to use this zone and it generally gives you better availability for the machines.



4.3 Name and Tags

Search and go to the EC2 service in AWS and click on **Launch instance** to begin.

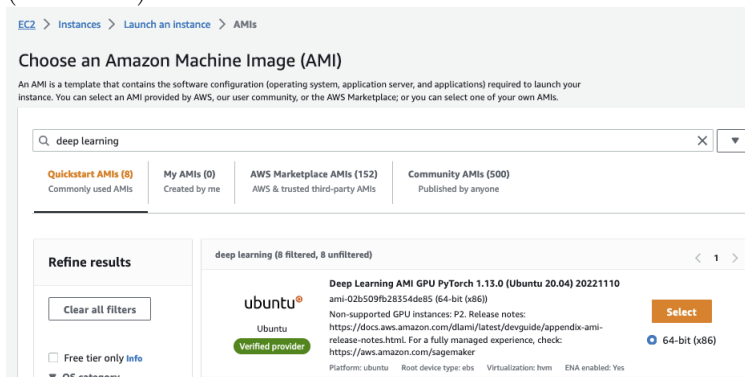


In Name and tags,

1. Give a name to your instance (it can be anything but it would be good to include your name e.g. pratikac-instance)
2. Click on **Additional Tags**, create a key called "Owner" and value as "your Penn email including everything after @" (i.e., pratikac@seas.upenn.edu for Pratik). Then add Instances and Volumes to the Resource types. *We have scripts to prevent people from deleting each other's instances accidentally. So you will not be able to interact with any instance or volume that was not launched with "Owner:your-key".*

4.4 Select Amazon Machine Image (AMI)

Under **Application and OS Images**, click **Browse more AMIs**, search **Deep Learning AMI**, and select "Deep Learning AMI GPU PyTorch ... (Ubuntu ...)".



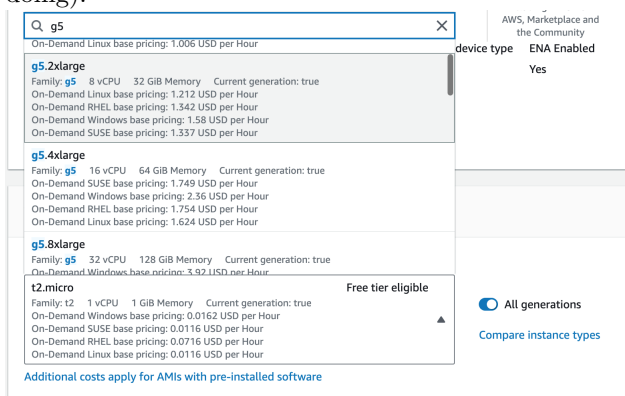
You can also use a standard Ubuntu but you will have to install everything (deep learning environment and libraries) yourself. so unless you know what you are doing, I would advise using the AMI.

4.5 Choose Instance Type

Under **Instance type**, change the instance type to one of the GPU instance. **You should use the following instances:**

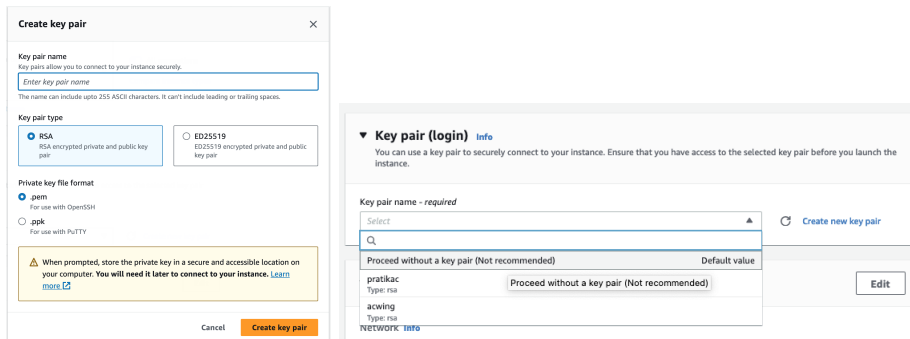
1. g5.xlarge: 1 Nvidia A30 GPU (4 CPU cores, 16 GB memory). This should be sufficient for almost everyone and it is also the cheapest.
2. g5.2xlarge: 1 Nvidia A30 GPU (8 CPU cores, 32 GB memory). If you want run a dataloader with many threads, e.g., for lots of augmentation.
3. g5.12xlarge: 4 Nvidia A30 GPUs (48 CPU cores, 192 GB memory). Do not use this in general. Use it only if you need all 4 GPUs. This instance costs 12 dollars/hour so you will use your 100-150 dollars or so credits very quickly.
4. You can also use g4 instances and they are only a tiny bit slower (these have Nvidia T4 GPU which is the same one that Google Colab, so they are equally fast).

You can run multiple training runs simultaneously on the same GPU without changing any of your code. This is not different from running multiple processes on the same CPU. You do not need multiple GPUs just because you are training multiple neural networks. **Do not use the other g5 instances** unless you know for sure that you will benefit from them (and you know what you are doing).



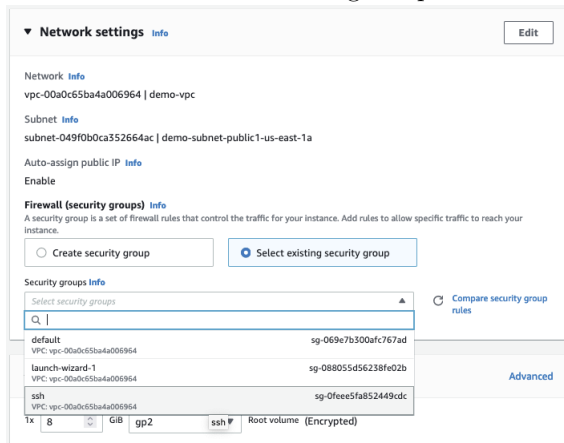
4.6 Create a keypair from the EC2 console

Under the Key pair (login) section, **Create new key pair**. Without this keypair you will not be able to SSH into the instance and will not be able to use it. It is a good idea to call the keypair by your Penn email, e.g., pratikac for Pratik.



4.7 Security Groups

Under **Network Settings**, choose the option to **Select existing security group** and select the security group titled "ssh". Do not try to create a new one (you will not be able to). We have forwarded a few ports (they are: 60000-60100, 1024-1048, and 22) that you can use to connect to Jupyter, VScode, Tensorboard, etc. Make sure that the option "**Auto-assign public IP**" is set to enable as in some cases it might be set to disable by default (you might need to click the **edit** button on right top to see the option).



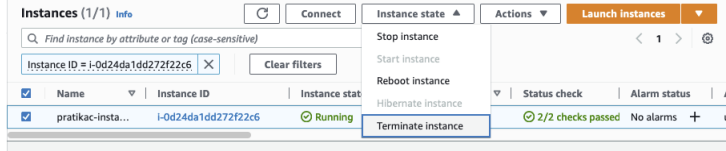
4.8 Launch the Instance

Click the **Launch Instance** button and you have a launched GPU instance to play with!

4.9 Stop and Start Instance

We have created a script that shuts down idle instances after 1 hour. If you see that your instance has "stopped" you can just start it using the button at the top. **Note: Always stop your instance after use.** Although the provided

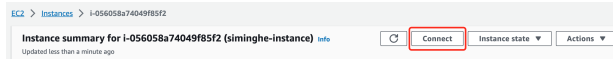
credits should suffice for this course, it is important to use resources carefully. An unstopped instance can quickly deplete the credits. **Stopping** the instance will not delete your data. **Terminating** the instance will delete all your data.



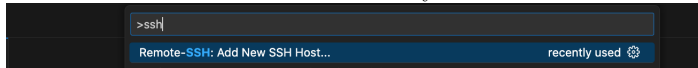
5 Using Your Instance

5.1 Use Your Instance in VS Code

Click **Connect** and you will be directed to a page called **Connect to instance**. Choose the **SSH Client** tab. You will see an example: “ssh -i ‘your-pennkey.pem’ address.com”. You can use this to access your instance from your laptop. **Important:** you need to change ‘your-pennkey.pem’ to ‘/absolute-path/your-pennkey.pem’ so that your computer can find the file.



Go to vs code, press **command + shift + P**, type in ssh, and select **add new SSH Host**. Then type in “ssh -i ‘/absolute-path/your-pennkey.pem’ address.com”. You will be connect to your instance.



5.2 Check the deep learning environment

You can check your environment by opening an terminal in the vs code window and type

```
conda info --env
```

You can activate the environment you want, e.g., pytorch, by

```
conda activate pytorch
```

and run your code by

```
python file-name.py
```

5.3 SCP: Secure Copy for File Transferring

You can use the following command to copy file from your computer to your AWS instance and vice versa. You can search for more details online.

```
scp -i '/absolute-path/penn-key.pem' source_path_of_file destination_path_of_file
```

5.4 tmux

Training may take a long time. If you accidentally closed the terminal or vs code, the terminal process will be closed which means your training will be killed and lost. You don't want this to happen! If you use tools like tmux, even if you close your terminal, the process will still be ran in the background and you can access it when you come back. Basic commands:

```
# session management
tmux ls (or tmux list-sessions)
tmux new -s session-name
Ctrl-b d Detach from session
tmux attach -t [session name]
tmux kill-session -t session-name

Ctrl-b c Create new window
Ctrl-b d Detach current client
Ctrl-b l Move to previously selected window
Ctrl-b n Move to the next window
Ctrl-b p Move to the previous window
Ctrl-b & Kill the current window
Ctrl-b , Rename the current window
Ctrl-b q Show pane numbers (used to switch between panes)
Ctrl-b o Switch to the next pane
Ctrl-b ? List all keybindings

# moving between windows
Ctrl-b n (Move to the next window)
Ctrl-b p (Move to the previous window)
Ctrl-b l (Move to the previously selected window)
Ctrl-b w (List all windows / window numbers)
Ctrl-b window number (Move to the specified window number, the
default bindings are from 0 - 9)

# Tiling commands
Ctrl-b % (Split the window vertically)
CTRL-b * (Split window horizontally)
Ctrl-b o (Goto next pane)
Ctrl-b q (Show pane numbers, when the numbers show up type the key to go to that pane)
Ctrl-b ( (Move the current pane left)
Ctrl-b ) (Move the current pane right)

# Make a pane its own window
Ctrl-b : "break-pane"

# add to ~/.tmux.conf
bind | split-window -h
bind - split-window -v
```

5.5 Use Your Instance in Jupyter Notebook

Follow this tutorial! [Link](#)