Probability

Arthur J. Redfern
arthur.redfern@utdallas.edu

Outline

- Motivation
- Probability spaces
- Random variables
- Random processes
- Information theory
- References

Disclaimer

- This set of slides is more accurately titled “A brief refresher of a subset of probability for people already somewhat familiar with the topic followed by it’s specific application to xNN related items needed by the rest of the course”

- However, that’s not very catchy so we’ll just stick with “Probability”

- In all seriousness, recognize that probability is a very broad and deep topic that has and will continue to occupy many lifetimes of work; if interested in learning more, please consult the references to open a window into a much larger world

Motivation

Probability

- xNNs can frequently be thought of as a function approximating P(output | input)
  - Input: images, words in language 1, sounds, ...
  - Output: localized and labeled objects, words in language 2, text, ...

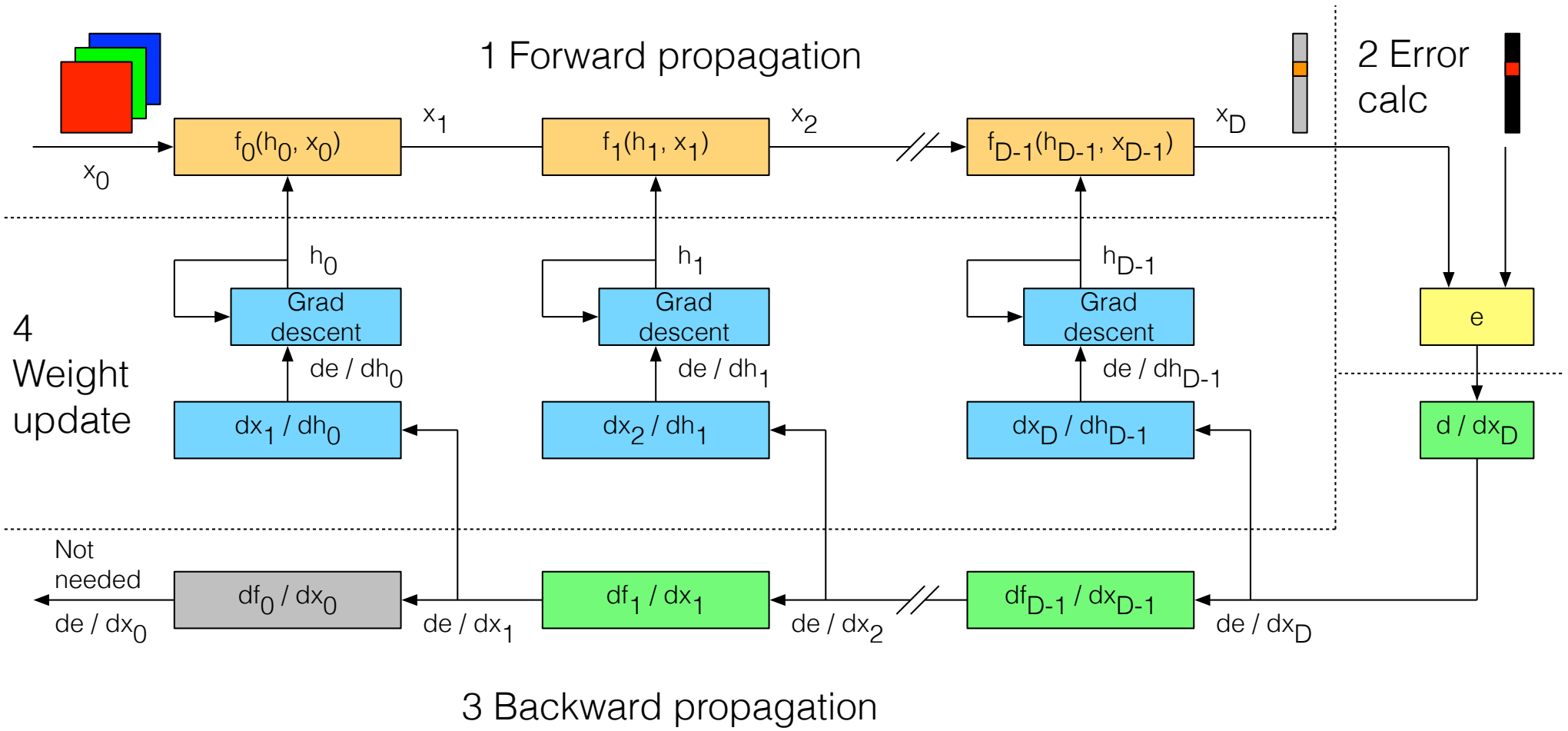
- Probability is used throughout xNN design, training and implementation
  - Batch normalization used to improve convergence
  - Soft max used to generate probability mass functions
  - KL divergence, cross entropy and MSE to form errors

Information

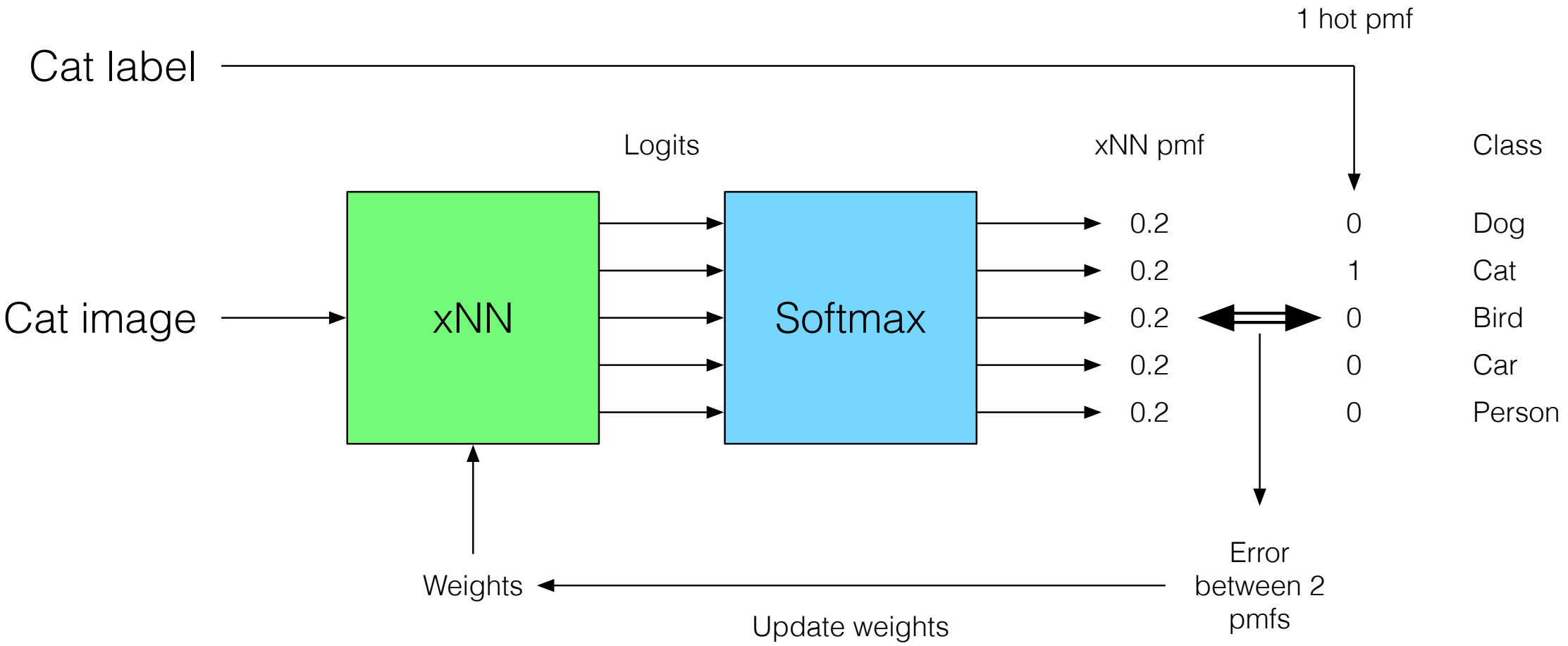
- Probability is the math that describes information
  - This course uses xNNs to extract information from data
  - This course uses xNNs to generate data from information

- Information theory is used everywhere implicitly or explicitly
  - Understanding machine learning as information extraction from training data (knowledge) to apply to the problem of information extraction from testing data
  - Understanding the flow of information through the network and implications of network design
  - Compressing filter coefficients and feature maps towards an information bound

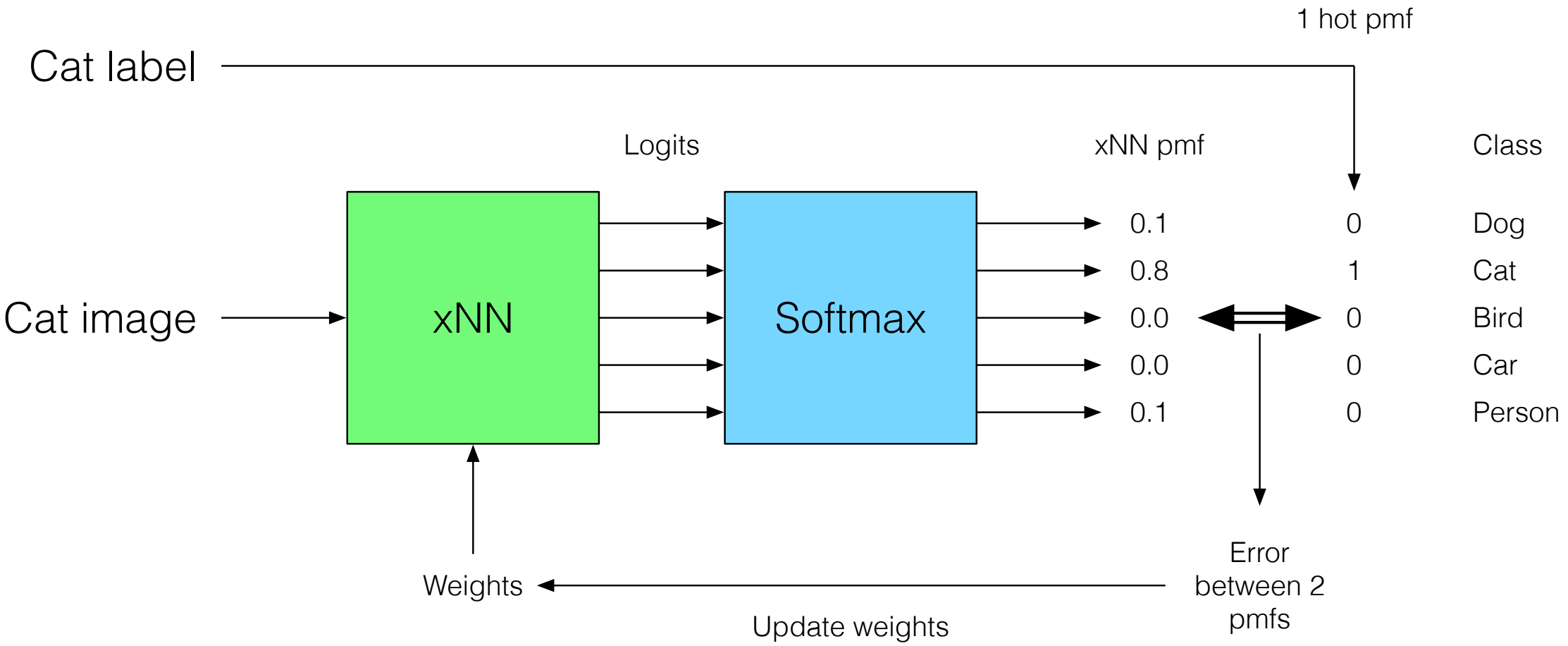
Big Picture Reminder



Focusing On The Mapping And Error



Focusing On The Mapping And Error



Probability Spaces

Probability Space Definition (S, E, P)

- A sample space S of all possible outcomes
  - Think: S is all possible outcomes of an experiment
  - Ex: flipping a coin 2x and recording heads (H) or tails (T) for each flip
  - S = {HH, HT, TH, TT}

- An event space E where each event is a set of 0 or more outcomes from the sample space
  - Think: E is all possible subsets of the sample space S (including nothing and everything)
  - E = {

∅,	// null subset
{HH}, {HT}, {TH}, {TT},	// all subsets of 1 outcome
{HH, HT}, {HH, TH}, {HH, TT}, {HT, TH}, {HT, TT}, {TH, TT},	// all subsets of 2 outcomes
{HH, HT, TH}, {HH, HT, TT}, {HH, TH, TT}, {HT, TH, TT}	// all subsets of 3 outcomes
{HH, HT, TH, TT} }	// sample space subset

- A probability measure function P: E → [0, 1] that satisfies
  - $P(A) \in R$ and $P(A) \geq 0$ for all events $A \in E$
  - $P(S) = 1$
  - $P(\cup_i A_i) = \Sigma_i P(A_i)$ for mutually exclusive events A_i
  - Think: P is a function that assigns probabilities to subsets of the sample space

Individual And Joint Probability

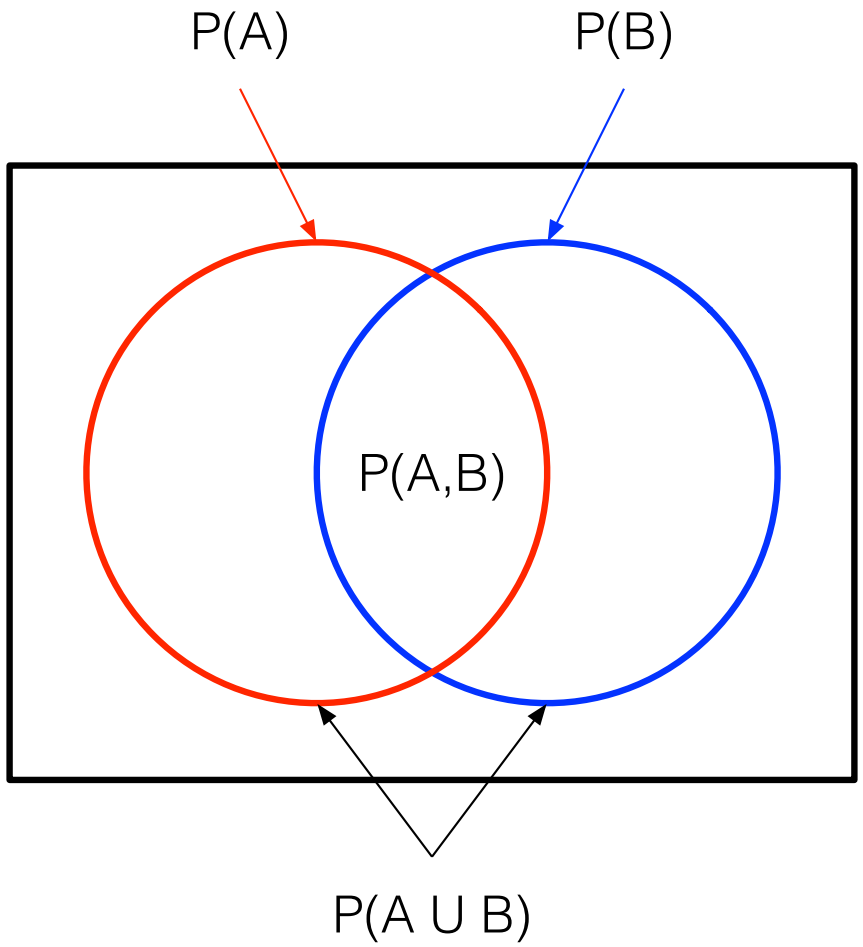
- Notation
  - 1 event A
  - 2 events {A, B}
  - K events $\{B_0, ..., B_{K-1}\}$

- Single
  - The probability of an event occurring — $P(A) \in [0, 1]$
  - The probability of an event not occurring — $P(A^c) = 1 - P(A)$ — A^c denoting not A

- Joint
  - The probability of events A and B occurring — $P(A, B)$ — also written as $P(A \cap B)$
  - If $A \subseteq B$ — $P(A, B) = P(A)$
  - If A and B are independent — $P(A, B) = P(A)\,P(B)$

- Union
  - The probability of event A or B occurring — $P(A \cup B) = P(A) + P(B) - P(A, B)$
  - If A and B are mutually exclusive — $P(A \cup B) = P(A) + P(B)$

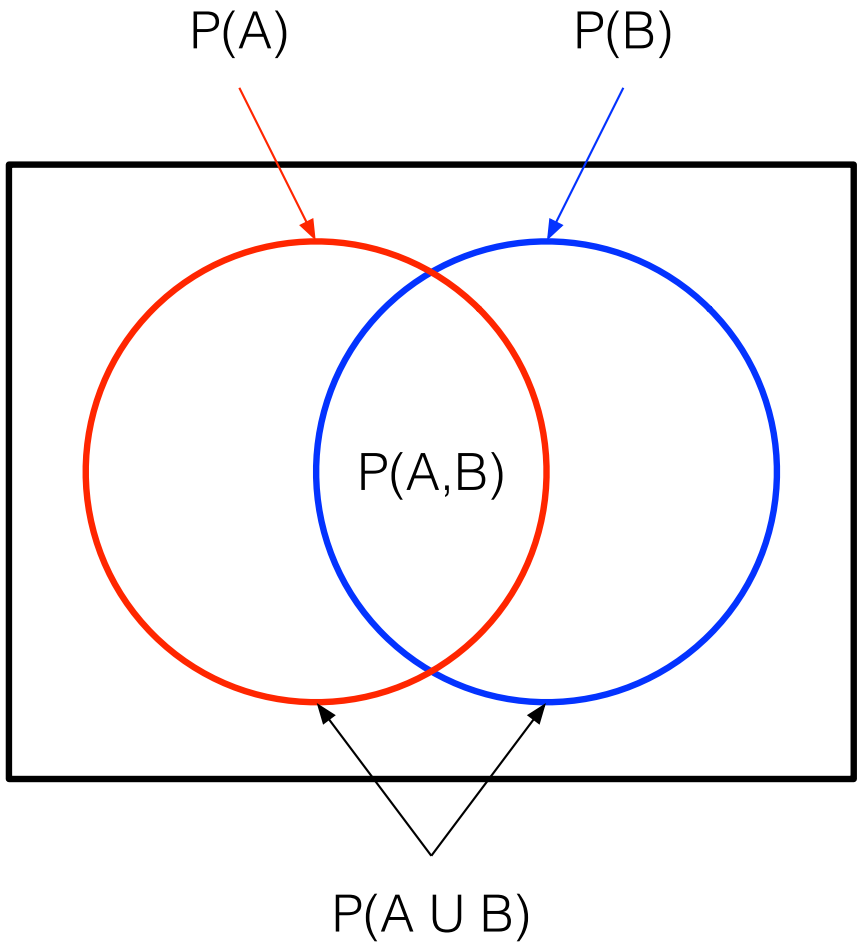
Individual And Joint Probability



Conditional Probability

- Conditional
  - The probability of event A given event B — $P(A|B) = P(A, B) / P(B)$
  - If A and B are independent — $P(A|B) = P(A)$
  - Bayes' theorem — $P(A|B) = P(B|A)\, P(A) / P(B)$
  - Chain rule of probability — $P(A_{K-1}, ..., A_0) = P(A_{K-1}|A_{K-2}, ..., A_0)P(A_{K-2}, ..., A_0)$
  - Can recursively apply to 2nd term on RHS

Conditional Probability



$$P(A \mid B) = P(A, B) / P(B)$$
$$P(B \mid A) = P(A, B) / P(A)$$

Random Variables

Discrete

- A discrete random variable is a function X with a finite or countably infinite range that maps outcomes s from the sample space S to numbers $x \in R$
  - $X(s) = x_k$
  - x_k is a realization of X

- Note that a random variable is not random and it's not a variable
  - The outcome of the experiment s is random
  - The mapping $X(s) = x_k$ by the random variable (function) to a real number is deterministic

Discrete

- A discrete random variable is described by it's probability mass function that specifies the probability that it takes on a specific value or it's cumulative distribution function that specifies the probability that it's value falls within an interval

- Probability mass function
  - Single
    $p_X(x_k) = P(X(s) = x_k)$
    where $\Sigma_k\ p_X(x_k) = 1$
  - Joint and conditional
    $p_{X,Y}(x_j, y_k) = p_{X|Y}(x_j|y_k)\ p_Y(y_k) = p_{Y|X}(y_k|x_j)\ p_X(x_j)$
  - Marginal
    $p_X(x_j) = \Sigma_k\ p_{X,Y}(x_j, y_k) = \Sigma_k\ p_{X|Y}(x_j|y_k)\ p_Y(y_k)$
  - Independent X and Y
    $p_{X,Y}(x_j, y_k) = p_X(x_j)\ p_Y(y_k)$
    $p_{X|Y}(x_j|y_k) = p_X(x_j)$

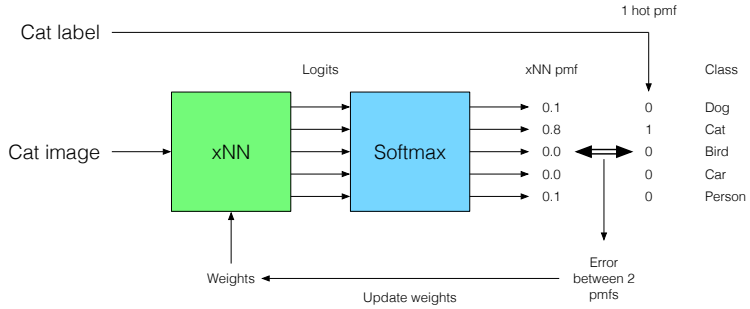
- Cumulative distribution function
  - Single
    $F_X(x_k) = P(X(s) \leq x_k) = \Sigma_{xj \leq xk}\ p_X(x_j)$

Softmax

- The softmax function maps a ~ arbitrary real vector input to a vector output with the following properties
  - All elements of the output are in (0, 1)
  - All elements of the output sum to 1

- Definition
  - $f(\mathbf{x}) = (1/(\Sigma_k e^{x(k)}))\ [e^{x(0)}, e^{x(1)}, ..., e^{x(K-1)}]^T$
  - Sometimes **x** is scaled by a non negative constant α
    - Values of α > 1 result in peakier pmfs
    - Values of α < 1 result in flatter pmfs

- In practice, we'll use this anytime we want to convert an arbitrary vector to a pmf
  - Ex: converting the vector output of a classification network to a probability mass function representing class probabilities



- Softmax is used all over the place in xNNs for this purpose so be comfortable with this
- ReLU, tanh and sigmoid are pointwise nonlinearities (each element can be computed without looking at the others)
- Softmax is different in that it's dependent on the full input vector for each output it creates

Expected Value

- Expected value is a linear operator that maps functions of random variables to a probability weighted average of all events (shown here for a discrete random variable)
  - $E[f(X(s))] = \Sigma_k \, p_X(x_k) \, f(x_k)$

- Scalar examples
  - Mean — $\mu_X = E[X(s)]$
  - Variance — $\sigma_X^2 = E[(X(s) - \mu_X)^2]$
  - Standard deviation — σ_X
  - nth order moment about the mean — $E[(X(s) - \mu_X)^n]$
  - Covariance (units of X(s)*Y(s)) — $cov(X(s), Y(s)) = E[(X(s) - \mu_X)\,(Y(s) - \mu_Y)] = E[X(s)\,Y(s)] - \mu_X\,\mu_Y$
  - Correlation ([-1, 1]) — $corr(X(s), Y(s)) = cov(X(s), Y(s)) \,/\, (\mu_X\,\mu_Y)$
  - Independent X(s) and Y(s) — $cov(X(s), Y(s)) = corr(X(s), Y(s)) = 0$

    $cov(X(s), Y(s)) = corr(X(s), Y(s)) = 0$ famously does not imply independent X(s) and Y(s)

Expected Value

- Vector examples
  - Notation $\mathbf{x} = [X_0(s), ..., X_{K-1}(s)]^T$
  - Mean vector $\boldsymbol{\mu}_\mathbf{x} = E[\mathbf{x}] = [E[X_0(s)], ..., E[X_{K-1}(s)]]^T$

- Matrix examples
  - Covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x},\mathbf{x}} = E[(\mathbf{x} - \boldsymbol{\mu}_\mathbf{X})\,(\mathbf{x} - \boldsymbol{\mu}_\mathbf{X})^T]$

    $\Sigma_{\mathbf{x},\mathbf{x}}(m, k) = E[(X_m(s) - \mu_{Xm})\,(X_k(s) - \mu_{Xk})]$

Experiment

- A class generated discrete probability mass function

- Experiment
  - Think of a 2 digit number between 10 and 50
  - Both digits are odd
  - Both digits are different from each other

Experiment

- A class generated discrete probability mass function
- Experiment
  - Think of a 2 digit number between 10 and 50
  - Both digits are odd
  - Both digits are different from each other
- How many people thought of the number 37?

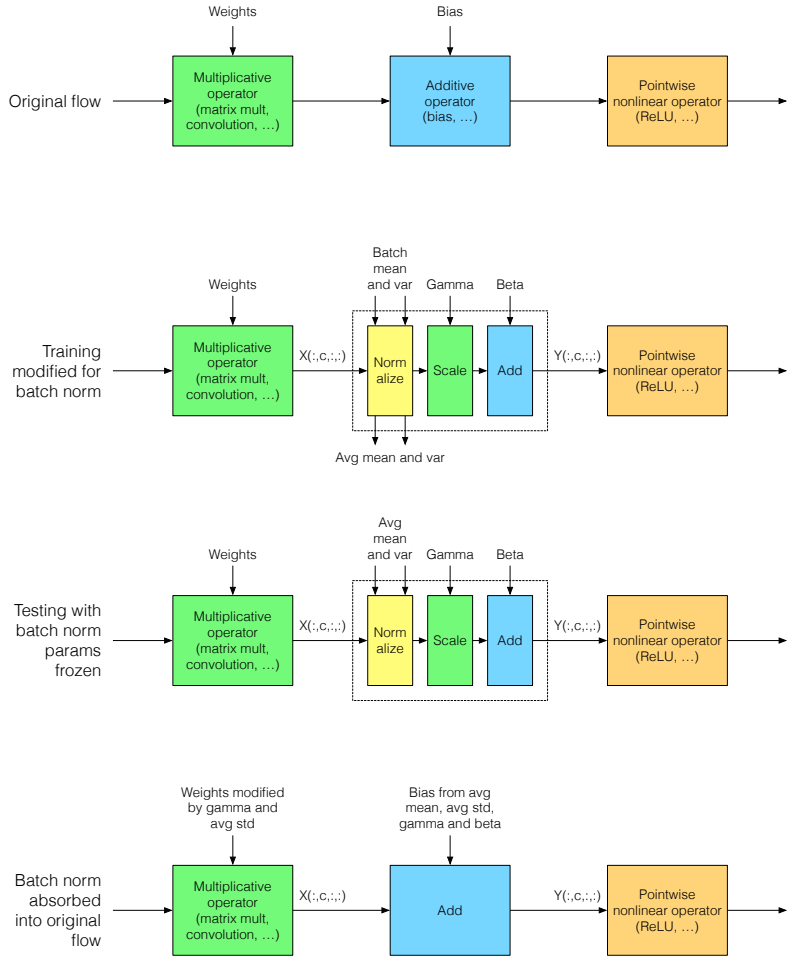
Random Variable Normalization

- Purpose
  - Take a random variable with an arbitrary distribution and normalize it to 0 mean and unit variance
    - Note that other variations of normalization exist
  - This is used by batch norm layers in CNNs to improve convergence during training
  - Note: CNNs use the word norm and normalization a lot for different operations
    - Input data normalization (a variant of what is described here)
    - Normalization layer (operates across feature maps, famous in AlexNet, rarely used now)
    - Batch normalization layer (a variant of what is described here, used in many places to improve training, can frequently be absorbed into convolution for deployment)
    - Group normalization layer (similar purpose to batch normalization, different operation)

- Normalization
  - $Y(s) = (X(s) - \mu_x) / \sigma_x$
  - $E[Y(s)] = E[(X(s) - \mu_x) / \sigma_x] = (1 / \sigma_x)(E[X(s)] - \mu_x) = (1 / \sigma_x)(\mu_x - \mu_x) = 0$
  - $E[(Y(s))^2] = E[((X(s) - \mu_x) / \sigma_x)^2] = (1 / \sigma_x^2)\, E[(X(s) - \mu_x)^2] = (1 / \sigma_x^2)\, \sigma_x^2 = 1$

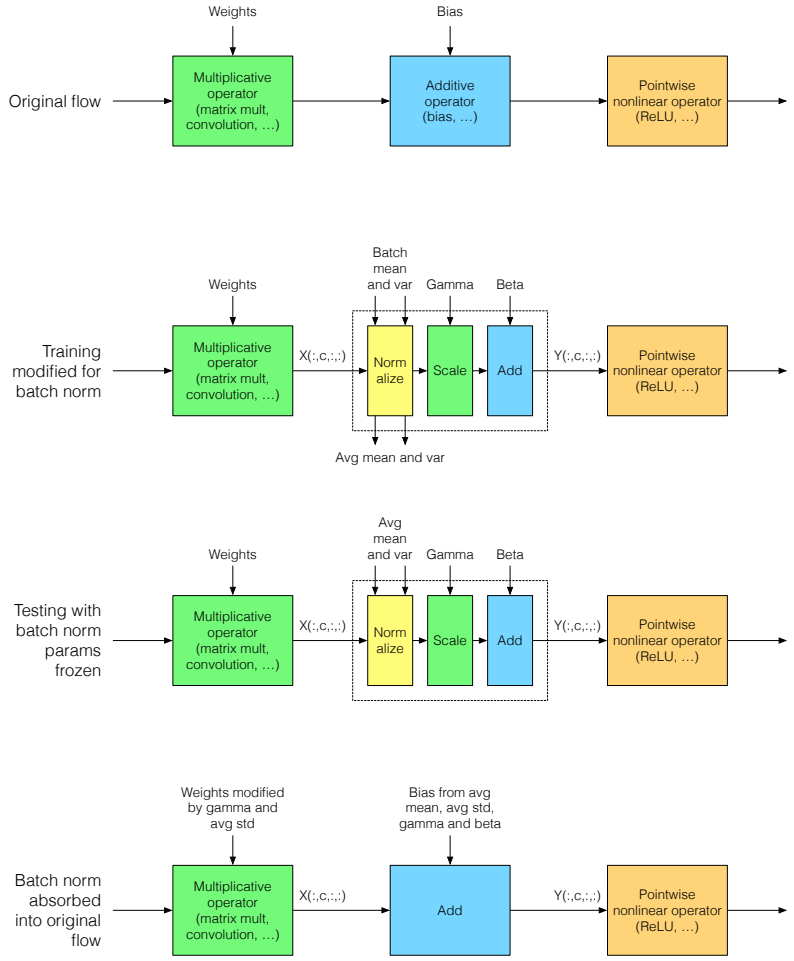
Batch Normalization

- Arguably 1 of the 2 most important techniques used to improve convergence during the training of deep xNNs (residual connections are the other)
- Notation: number N x channel C x height H x width W
  - Input X(n, c, h, w)
  - Output Y(n, c, h, w)
- Normalization is done on a per channel basis
  - Starting point of convolution, bias and ReLU
  - Modified during training to replace bias with batch norm
  - Batch norm params frozen to avg values during testing
  - Or subsequently absorbed back into convolution and bias



Batch Normalization

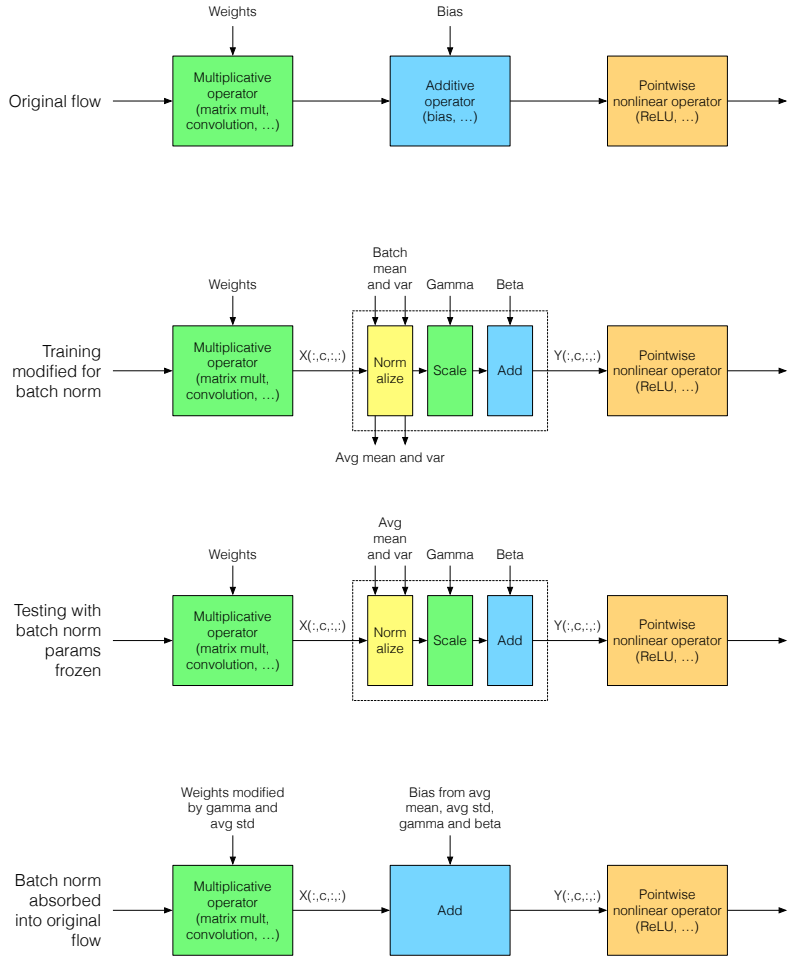
- During training for each channel c
  - Compute mean and variance for batch b
    - $\mu_{c,b} = 1/(NHW) \Sigma_{n,h,w} X(n, c, h, w)$
    - $\sigma_{c,b}^2 = 1/(NHW) \Sigma_{n,h,w} (X(n, c, h, w) - \mu_{c,b})^2$
  - Transform data per channel using per batch per channel mean and variance and per channel trainable scale γ_c and trainable bias β_c
    - $Y(:,c,:,:) = \gamma_c (X(:,c,:,:) - \mu_{c,b}) / \sigma_{c,b} + \beta_c$
  - Track running average of mean and variance across batches for subsequent use during testing ($\alpha \sim 0.99$)
    - $\mu_c = \alpha \mu_c + (1 - \alpha) \mu_{c,b}$
    - $\sigma_c^2 = \alpha \sigma_c^2 + (1 - \alpha) \sigma_{c,b}^2$
  - Update per channel trainable scale γ_c and trainable bias β_c along with other weights during weight update



Batch Normalization

- During testing for each channel c
  - Option 1: leave in batch norm with fixed params and using the running averages for the mean and variance
    - $Y(:,c,:,:) = \gamma_c (X(:,c,:,:) - \mu_c) / \sigma_c + \beta_c$
  - Option 2: absorb the batch norm into a convolution and bias and get rid of the batch norm
    - $Y(:,c,:,:) = \gamma_c (X(:,c,:,:) - \mu_c) / \sigma_c + \beta_c$
      $= (\gamma_c / \sigma_c) X(:,c,:,:) + (\beta_c - \gamma_c \mu_c / \sigma_c)$
    - Scale weights for convolution channel c by (γ_c / σ_c)
    - Add bias of $(\beta_c - \gamma_c \mu_c / \sigma_c)$

- Note: many other techniques for normalization exist for xNNs



Law Of Large Numbers

- Let $X_0(s)$, $X_1(s)$, ... be a sequence of independent identically distributed random variables with $E[X_i(s)] = \mu_X$ and let the sample average be
  - $X_{0:K-1}^{bar}(s) = (X_0(s) + ... + X_{K-1}(s))/K$

- $X_{0:K-1}^{bar}(s)$ converges to μ_X as $K \rightarrow \infty$
  - In probability for the weak law (unlikely outcome probability reduces as $K \rightarrow \infty$)
  - Almost surely for the strong law (pointwise)

- Variants exist that replace the independence constraint with a variance constraint

- The law of large numbers allows the expected value of a random variable with a finite mean to be estimated from it's sample average
  - Note that the sample average is a random variable

Central Limit Theorem

- The central limit theorem describes the distribution of the sample average on the previous slide (which is a random variable) about μ_X as $K \rightarrow \infty$
- Let $\{X_0(s), ..., X_{K-1}(s)\}$ be a set of independent identically distributed random variables each with mean μ_X and finite variance σ_X^2, then
  - $K^{1/2} (X_{0:K-1}^{bar}(s) - \mu_X) \rightarrow N(0, \sigma_X^2)$
- $N(0, \sigma^2)$ is 0 mean σ^2 variance Gaussian distribution
  - So $X_{0:K-1}^{bar}(s)$ is "close" to $N(\mu_X, \sigma_X^2 / K)$
- Convergence is in distribution (the cdf converges as $K \rightarrow \infty$)
- Variants exist that replace the independent identically distributed condition

Central Limit Theorem

- A few places where the central limit theorem sort of sometimes comes up
  - Viewing the inner product in matrix vector or matrix matrix multiplication as a weighted sum of random variables
  - Viewing the DFT operation as a (rotated) sum of random variables

- Why this matters (more specifically, 1 reason why this matters)
  - Inputs can have ~ arbitrary distribution, maybe nicely bounded
  - But the output of the operation starts to look Gaussian
  - Gaussian random variables have long tails
  - With finite precision arithmetic this affects accuracy

- More on precision when CNN performance and implementation is discussed

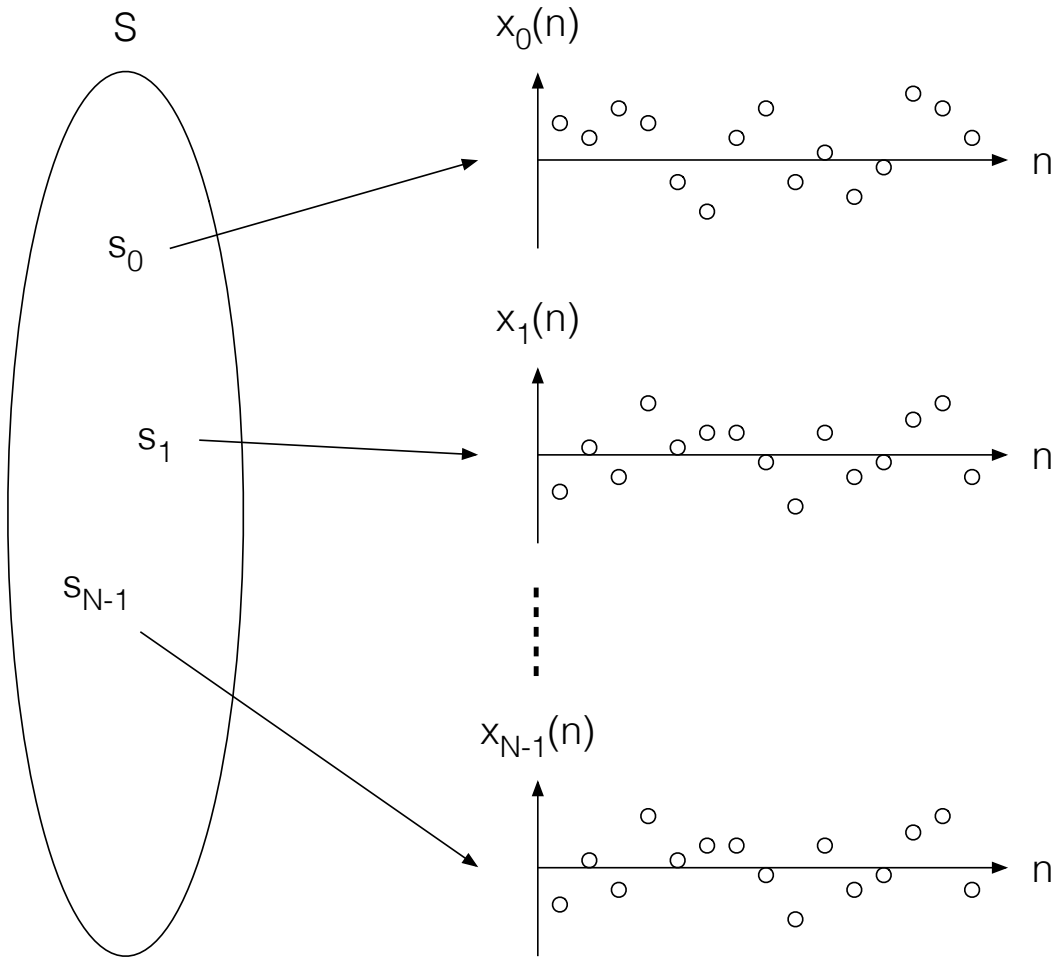
Random Processes

Definition

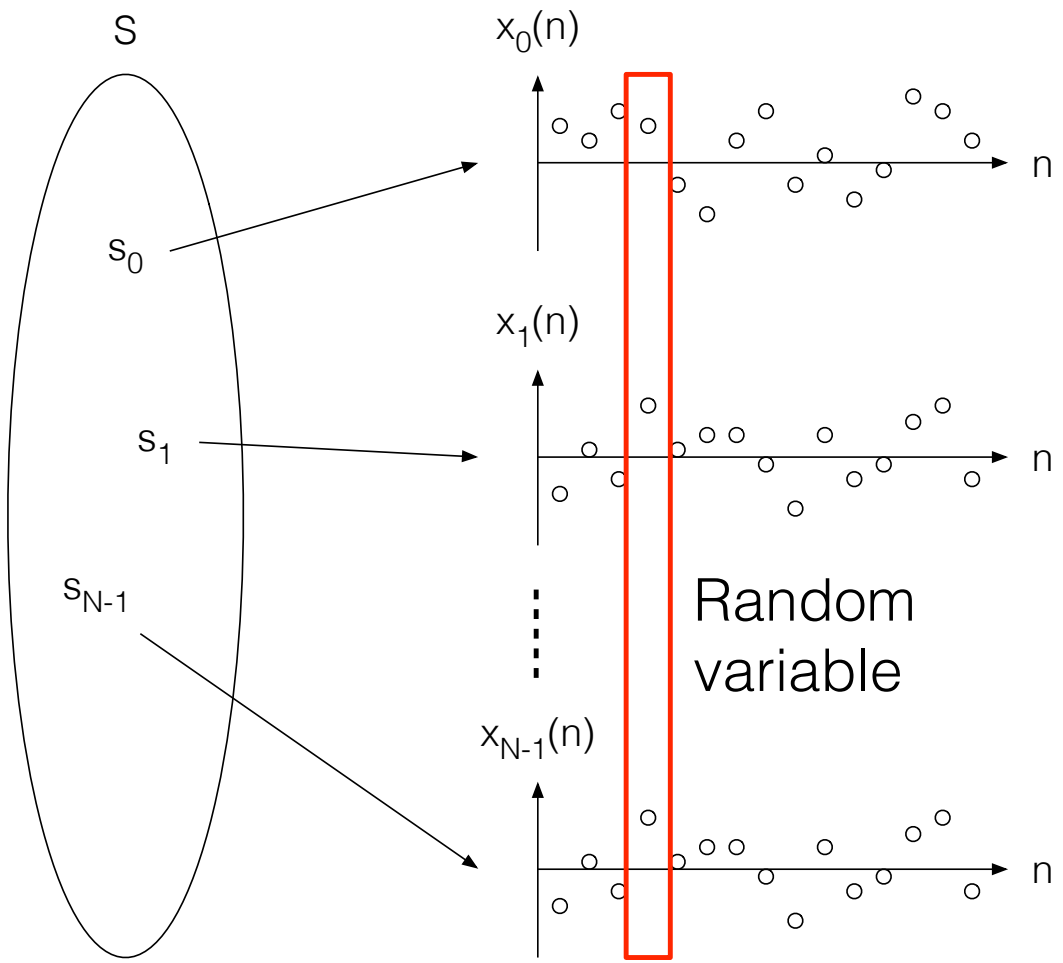
- A random process X(s, n) maps events s from the sample space S to functions x(n) where the domain of the function is the index set and the range of the function is the state space
  - X(s, n) is a random variable at a fixed n
    - By considering all times n this leads to the observation that a random process can be considered a collection of random variables $\{X(s, n_0), ..., X(s, n_{N-1})\}$
  - X(s, n) is a deterministic function of n for a fixed s
    - This is referred to as a realization of the random process
    - The set of all possible functions is referred to as the ensemble
  - X(s, n) is a number for a fixed s at a fixed n

- Names
  - If n refers to time then X(s, n) is called a random process
  - If n has multiple dimensions like width and height of an image then X(s, n) is called a random field

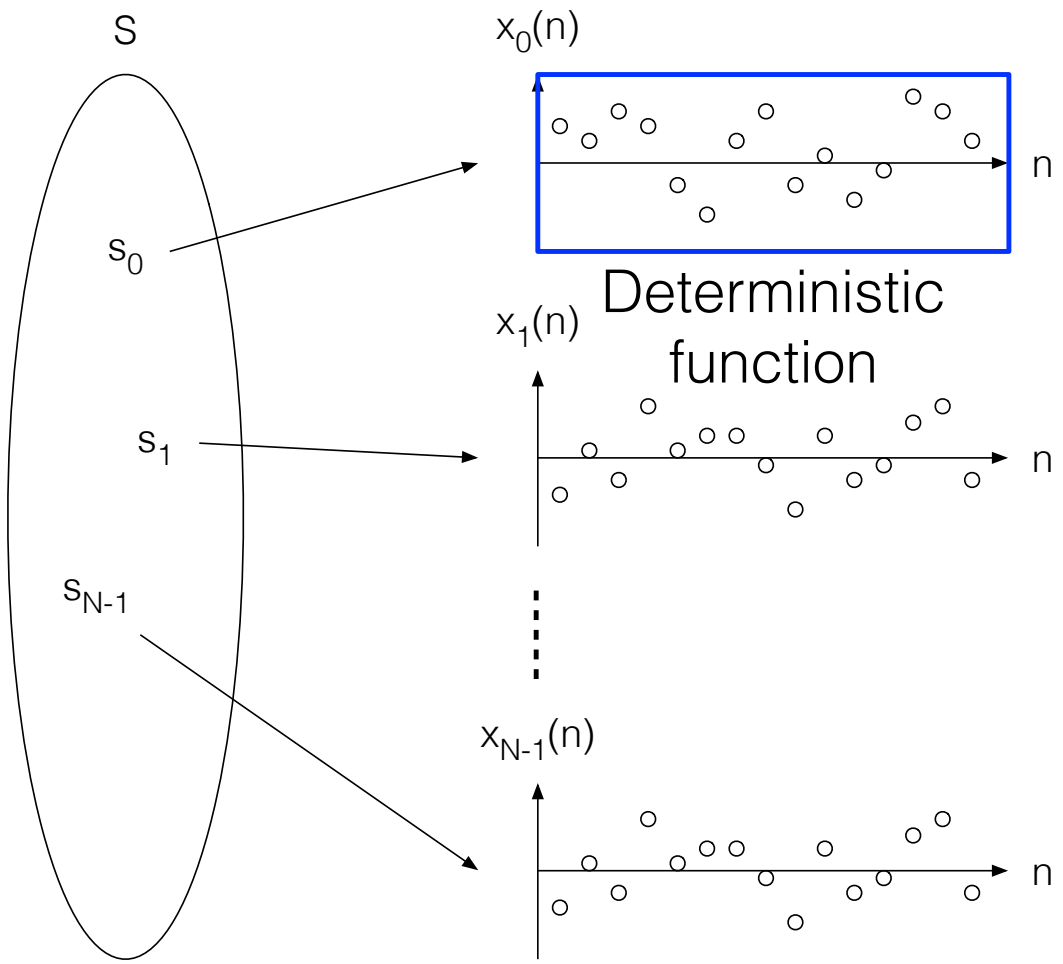
Definition



Definition



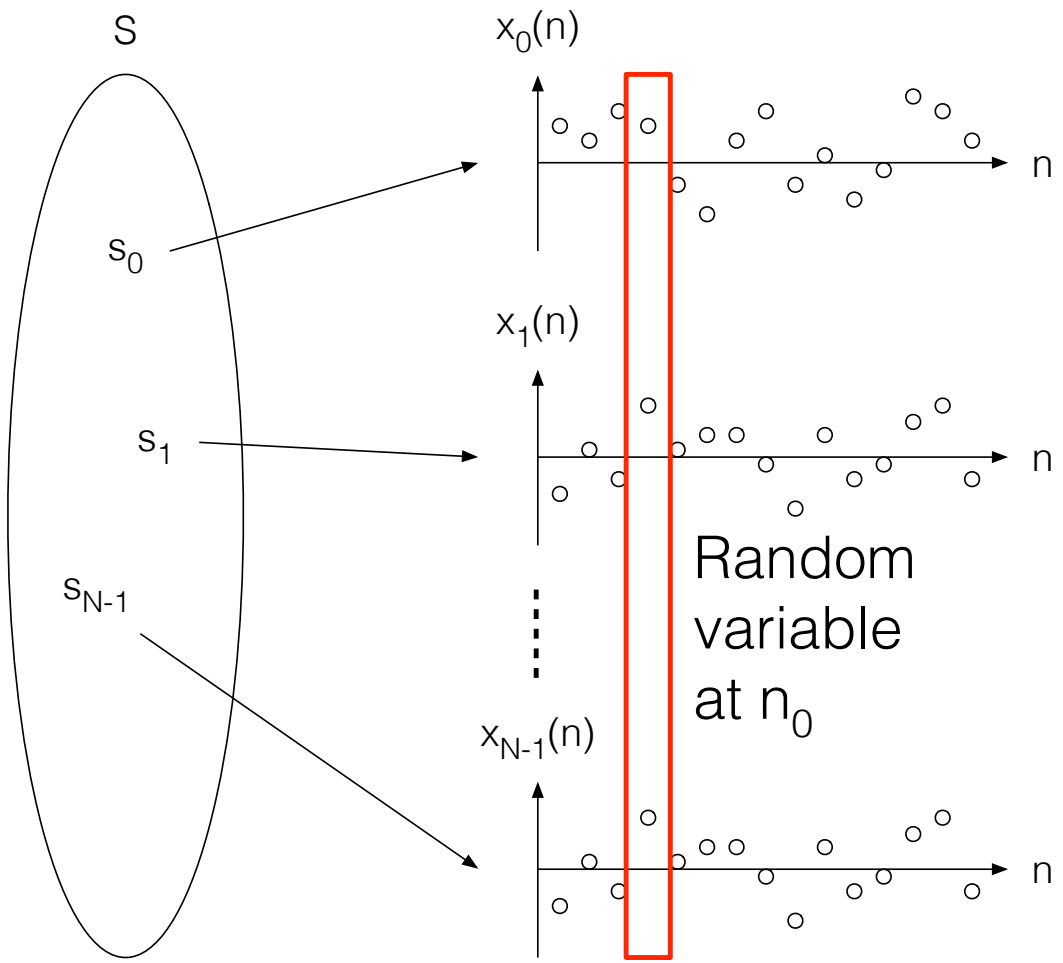
Definition



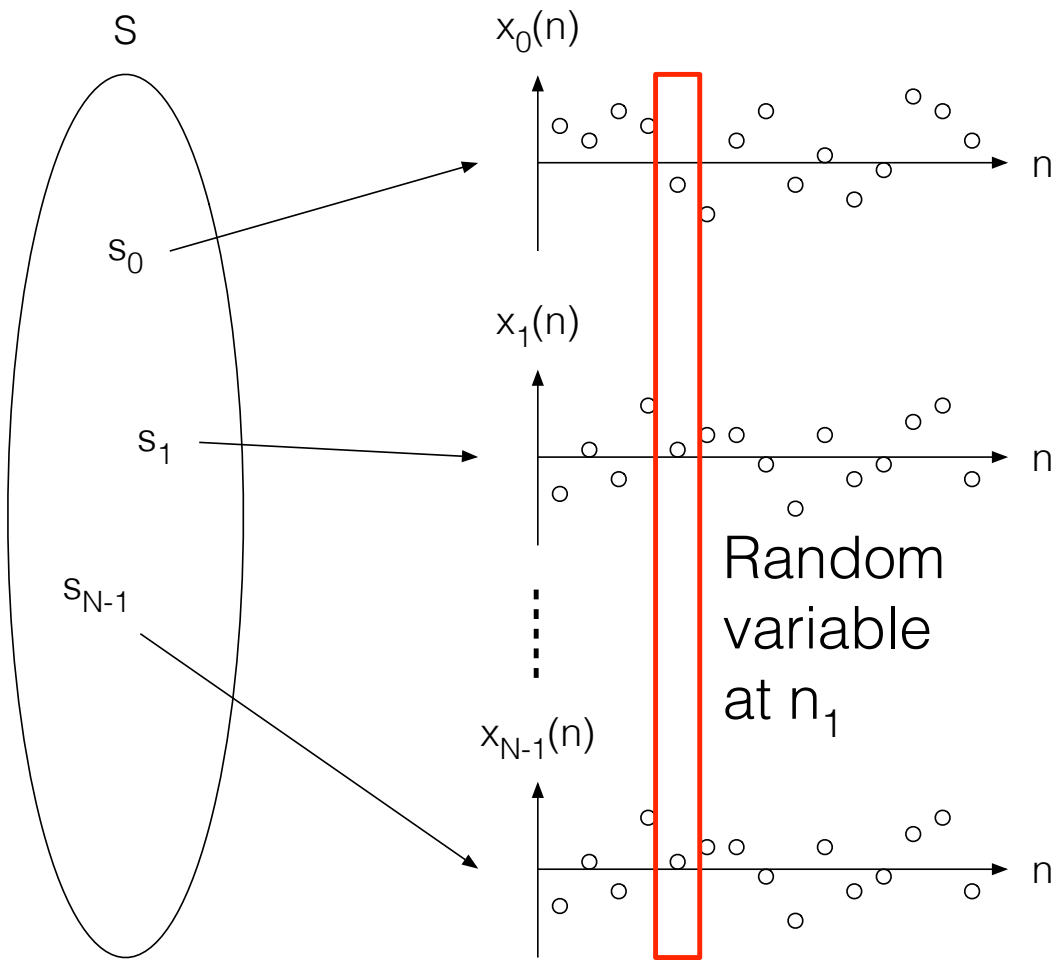
Stationarity

- Non stationary
  - Using the view of a random process as a collection of random variables, a random process is defined by is joint CDF $F_{X\_0,\ldots,X\_\{N-1\}}(x_{n\_0}, \ldots, x_{n\_(N-1)})$ which in general is a function of n_k
  - Informally, a non stationary random process has a CDF that changes with n (and doesn't fit neatly into 1 of the less restrictive stationary categorizations)
- (Strictly) stationary
  - Random processes X(s, n) for which the joint CDF does not change with time
  - $F_{X\_0,\ldots,X\_\{N-1\}}(x_{n\_0+\tau}, \ldots, x_{n\_(K-1)+\tau}) = F_{X\_0,\ldots,X\_\{N-1\}}(x_{n\_0}, \ldots, x_{n\_(K-1)})$ for all K, n and τ
- Weakly (wide sense or second order) stationary
  - Random processes X(s, n) for which the mean and auto covariance do not change with time
  - Autocorrelation only depends on time difference $\tau = n_1 - n_2$
- Other types of stationarity exist (e.g., cyclostationary)

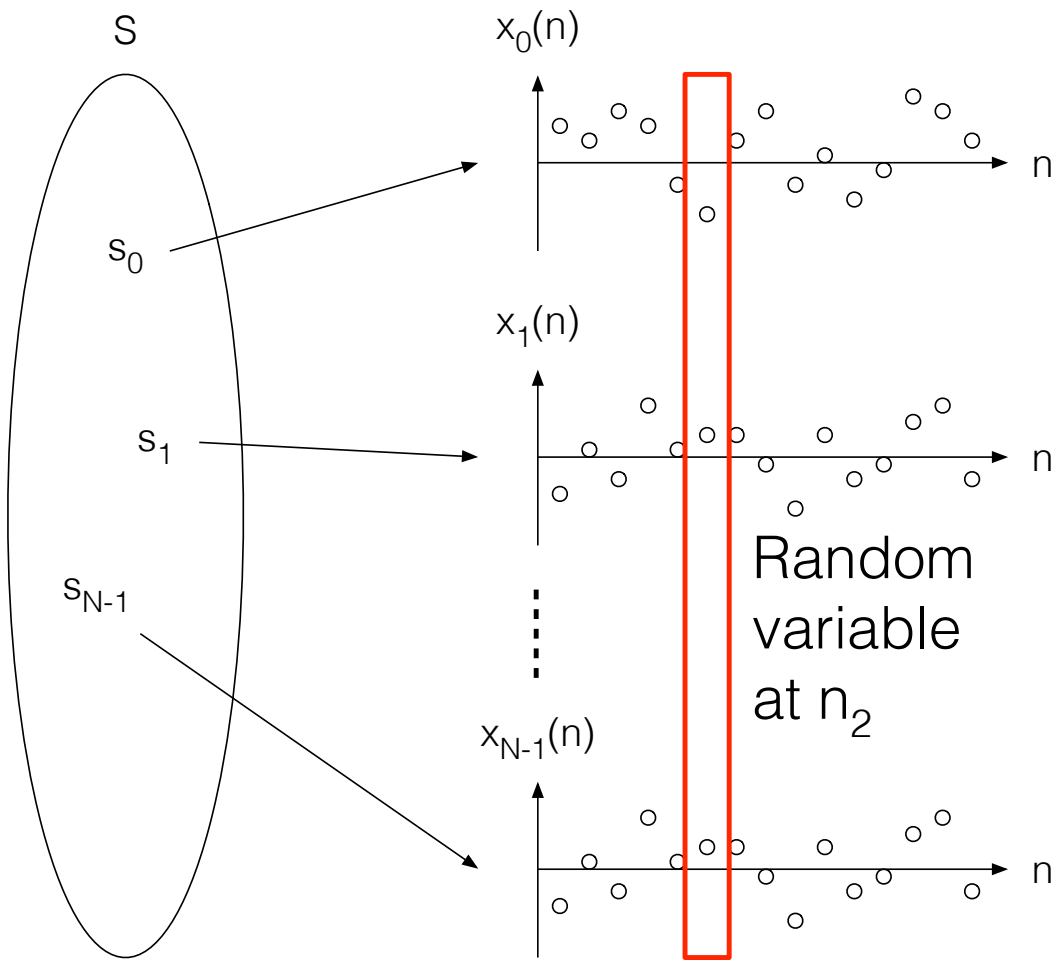
Stationarity



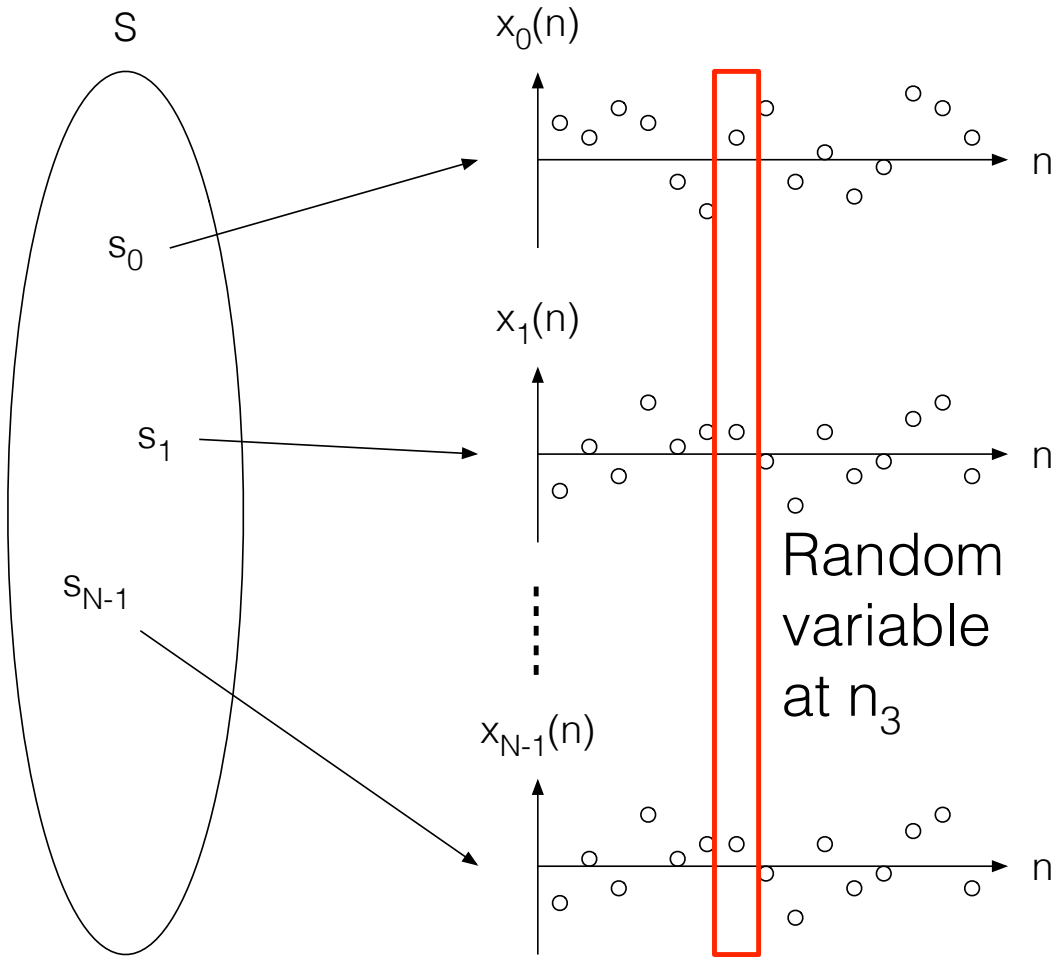
Stationarity



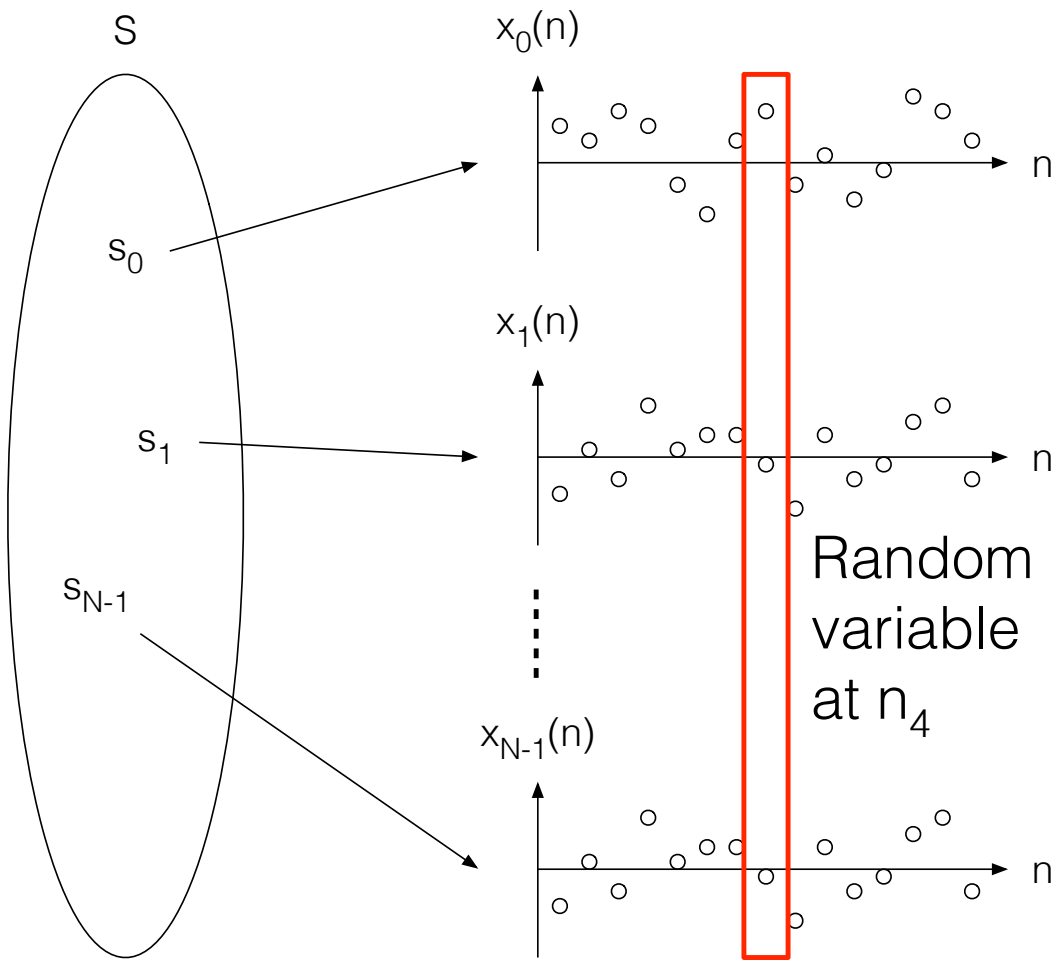
Stationarity



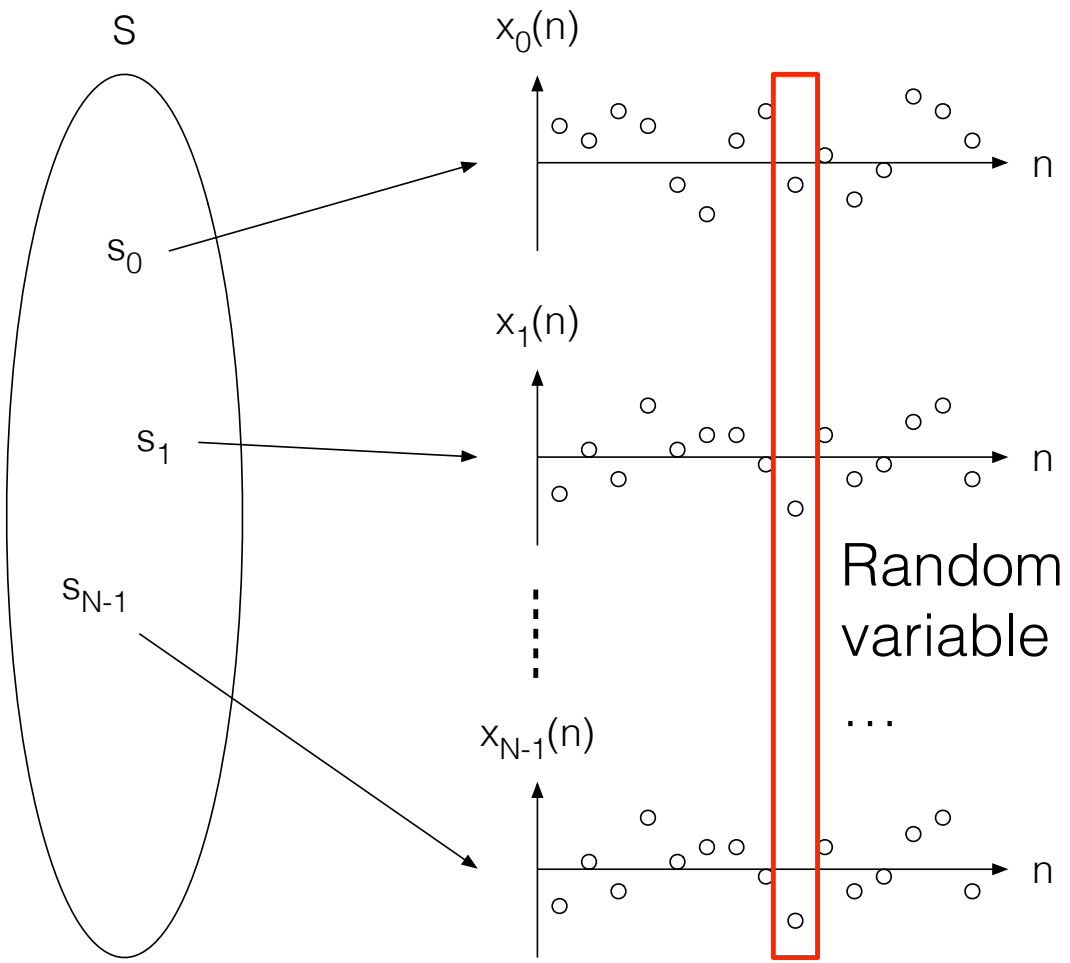
Stationarity



Stationarity



Stationarity

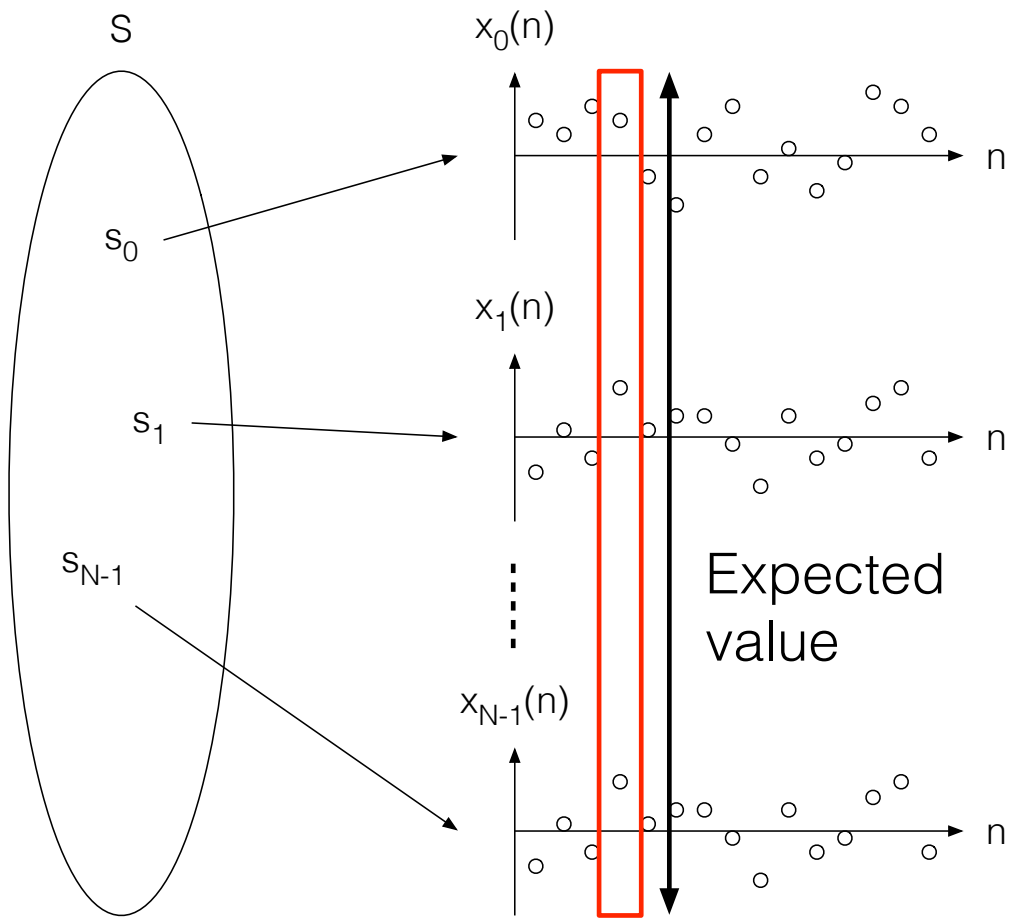


Expected Value

- The expected value of a random process is found by viewing the random process as a random variable at a fixed n and applying the expected value operator as before
  - Conceptually, it operates across many realizations s of a random process at a single n
  - Mean, variance and higher order moments are defined as in the case of a random variable

- Let $p_X(x_i, n) = P(X(s, n) = x_i)$ at a fixed n, then
  - $E[f(X(s, n))] = \Sigma_i \, p_X(x_i, n) \, f(x_i)$
  - Which in general is a function of n

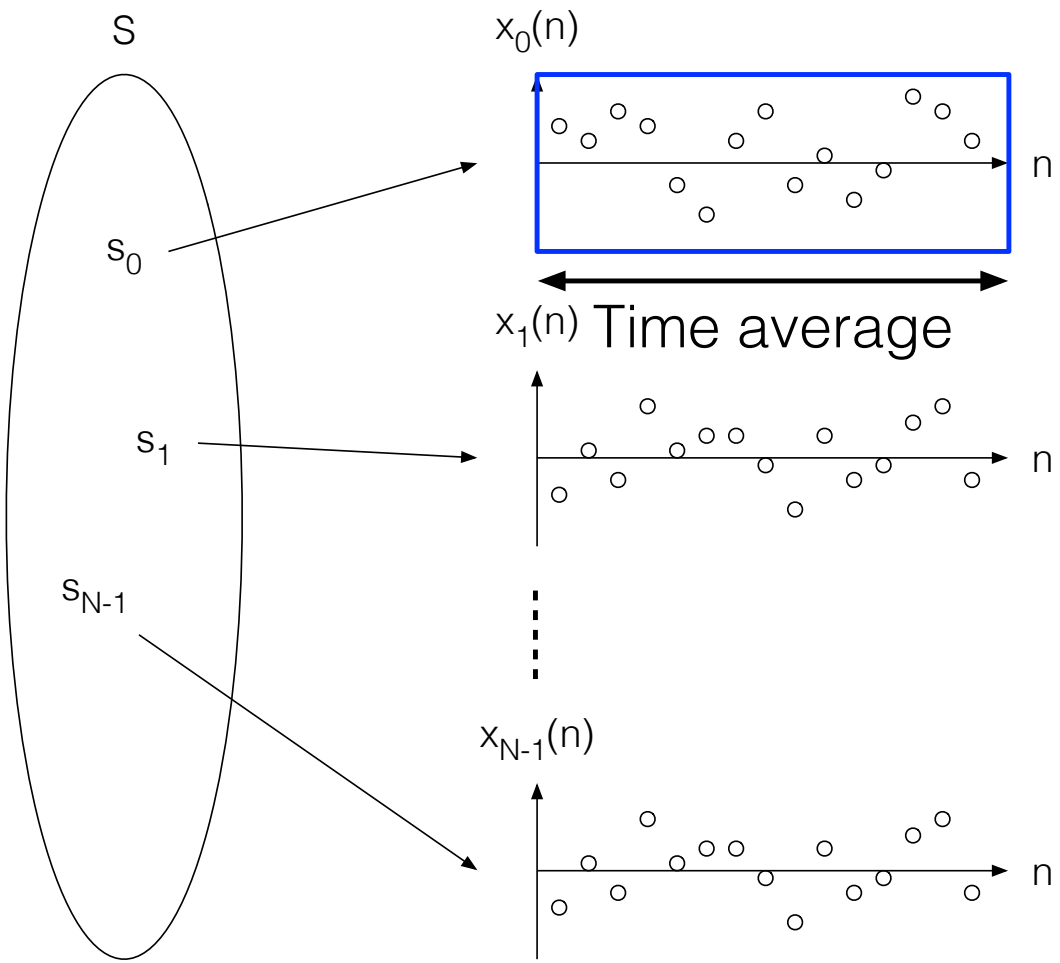
Expected Value



Time Average

- The time average of a random process is found by viewing the random process as a deterministic function for a fixed s and applying the time average operator
  - Conceptually, it operates across 1 realization s of a random process at many points n
  - Different time averages are defined similar to the expected value of a random variable
  - The time average itself is a random variable as it depends on the chosen s
  - $\langle f(X(s, n)) \rangle = 1/N \, \Sigma_n \, f(X(s, n))$

Time Average



Ergodicity

- Ergodicity: when time averages converge to expectations
  - In some sense (e.g., mean square)
  - For some orders of moments for which the process is stationary

- Example: mean ergodic
  - $\langle X(s, n) \rangle$ converges in the mean and in the mean square sense to $E[X(s, n)]$
    - $\lim_{N \to \infty} E[((1/N \sum_n f(X(s, n))) - \mu_X)] = 0$
    - $\lim_{N \to \infty} E[((1/N \sum_n f(X(s, n))) - \mu_X)^2] = 0$

Exponential Moving Average

- This is a common method for estimating a local average for slowly time varying data
  - Some xNN weight update methods use it for this purpose
  - We've already seen it used in batch norm

- Choose $\alpha \in (0, 1)$ to control the relative weighting between the current sample and the previous estimate
  - α close to 1 weights the current sample more and adapts faster but averages less
  - α close to 0 weights the previous estimate more and adapts slower but averages more

- Computation
  - $\mu_X(n) = x(n), \quad n = 0$ (other initializations possible)
    $= \alpha\, x(n) + (1 - \alpha)\, \mu_X(n - 1), \quad n > 0$

Information Theory

1 Word Definition

- Information is surprise

Before Formalities

- How many fingers do you think an alien has on their hand?
  - My favorite question in Cover and Thomas' book Elements of Information Theory
- Why do slides with lots of equations on them have 0 information during their presentation?
- Claude Shannon and communication system design
  - Inner and outer encoders and decoders with a noisy channel in the middle
  - Remove redundancy for compression, add redundancy for coding
  - Linear algebra, calculus and probability

Entropy

- Purpose
  - A way to mathematically quantify information

- Example
  - Consider a 1 bit message that can take on 2 values $x_k = \{0, 1\}$
    - Re: Bernoulli random variable
  - A transmitter sends a message to a receiver containing 1 bit of data
  - How much information is contained in the message?
    - If $p_X(0) = 1$ and $x_k = 0$ is received? Is there any surprise?
    - If $p_X(1) = 1$ and $x_k = 1$ is received? Is there any surprise?
    - If $p_X(0) = p_X(1) = 0.5$ and $x_k = 0$ or $x_k = 1$ is received? Is there any surprise?
    - Some other $p_X(0) = 1 – p$, $p_X(1) = p$ split?

Entropy

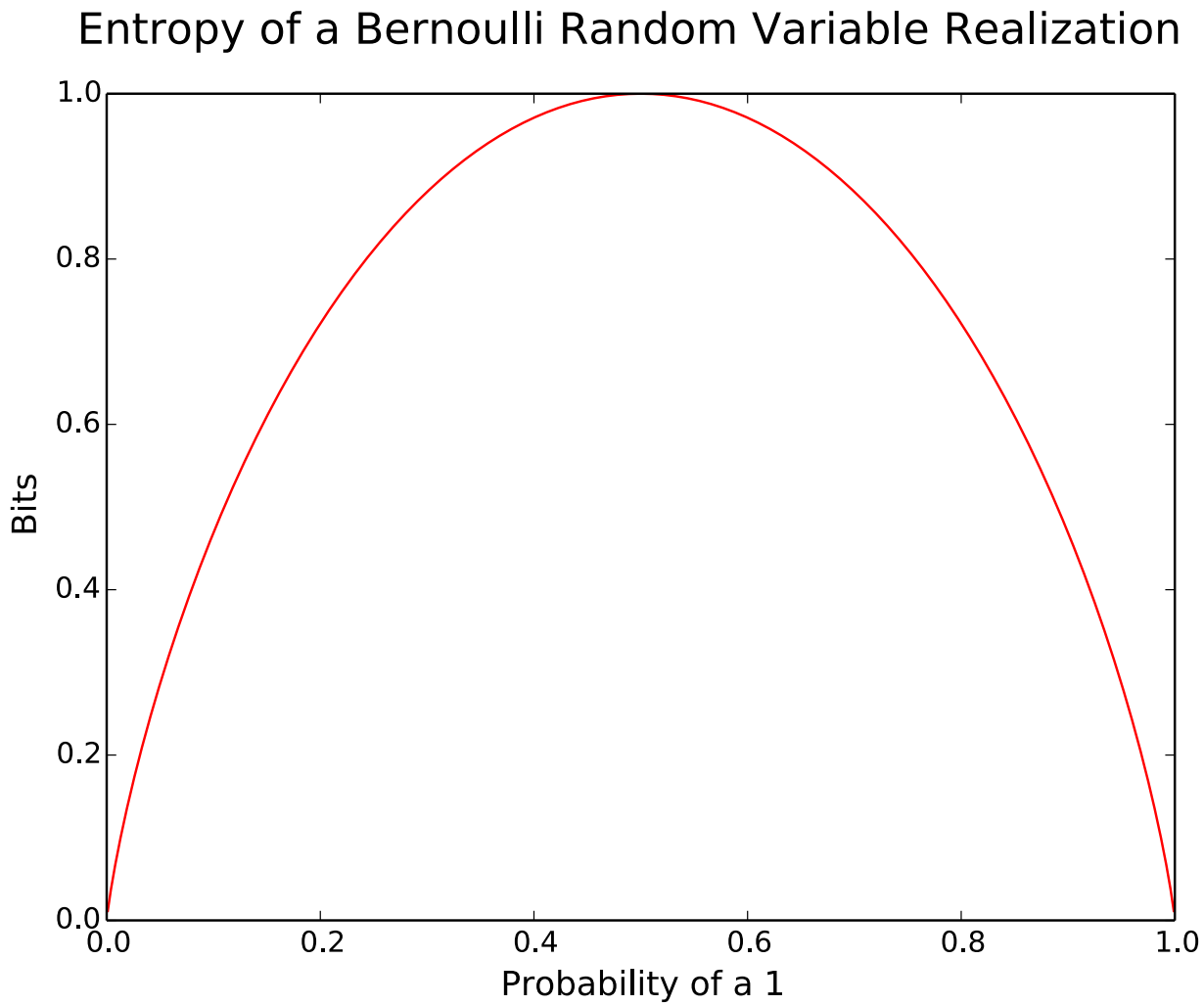
Note: this definition and most subsequent slides will consider entropy in the context of discrete random variables

- Definition
  - Informally, entropy is the information in a realization of a random variable
  - $H(X(s)) = -\Sigma_k\, p_X(x_k)\, \log_2(p_X(x_k))$
  - Units of bits because of log base 2 choice

- Revisiting the example on the previous slide
  - p: $H(X(s)) = -((1-p)\log_2(1-p)) - (p\log_2(p))$, general formula for example
  - p = 0.0: $H(X(s)) = -1\log_2(1) = 0$ bits, no surprise, no information
  - p = 1.0: $H(X(s)) = -1\log_2(1) = 0$ bits, no surprise, no information
  - p = 0.5: $H(X(s)) = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$ bit, max information

- Information and data are not the same thing
  - In the example there was always 1 bit of data
  - But the information H(X(s)) varied based on the value of p (more generally the PMF)

Entropy


Entropy of a Bernoulli Random Variable Realization
Bits
Probability of a 1
0.0
0.2
0.4
0.6
0.8
1.0

Entropy

- What distribution maximizes entropy under what constraints
  - $x_k \in \{a, a + 1, ..., b\}$: discrete uniform distribution
  - $x \in [a, b]$: continuous uniform distribution
  - $x \in (-\infty, \infty)$, $E[X(s)] = \mu_x$, $E[(X(s) - \mu_x)^2] = \sigma_x^2$: Gaussian distribution with mean μ_x and variance σ_x^2

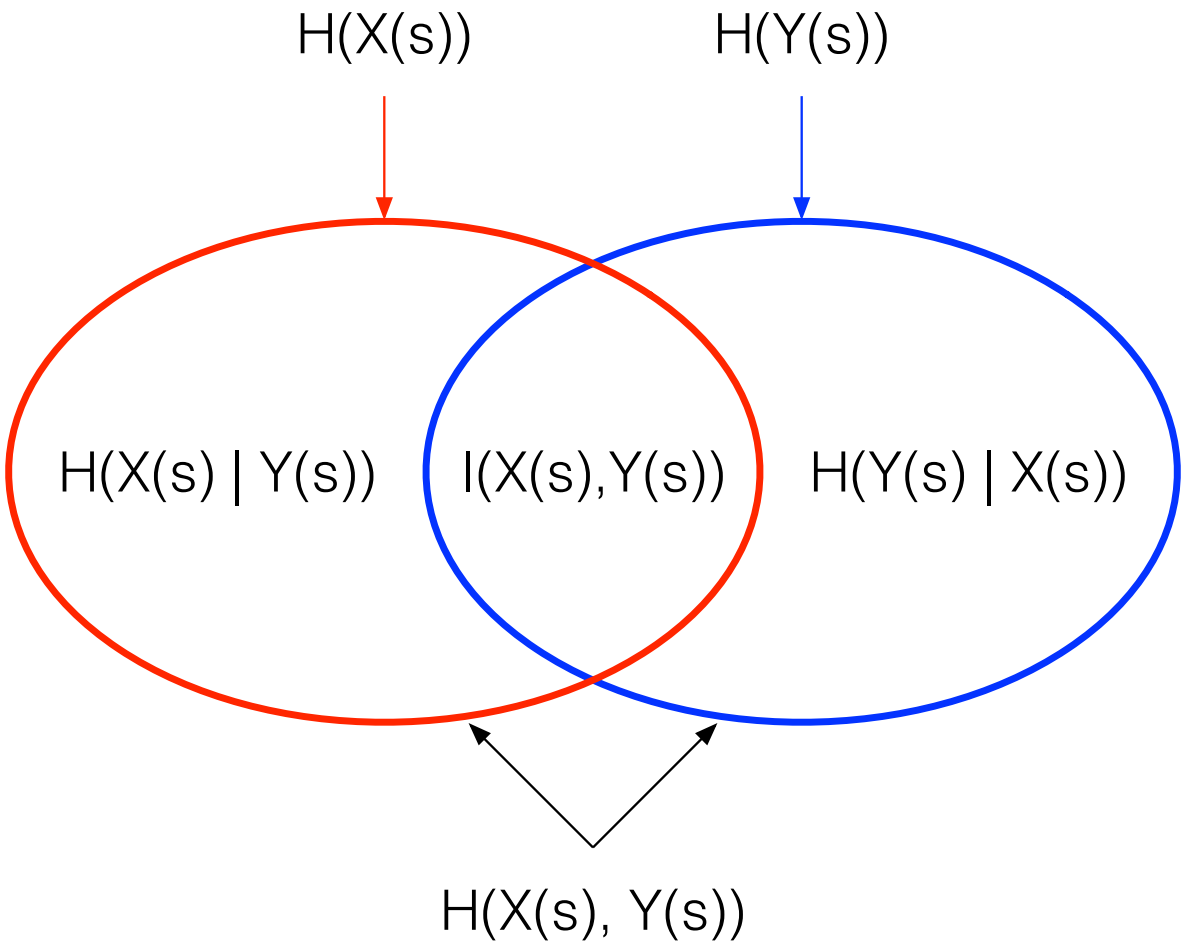
- For most success stories of CNNs, the input to the network is not an entropy maximizing distribution
  - Actually, it's just the opposite
  - And that's a good thing that's as it's implicitly exploited by the network
    - Natural images have a certain look to them
    - The human voice has a certain sound to it
    - Language has a certain structure to it
  - Think of it from the perspective of a network doing function approximation
    - Having a smaller domain to map to a finite set makes the mapping easier

Joint Entropy

- Definition
  - Informally, the information in a realization of 2 random variables
  - $H(X(s), Y(s)) = -\Sigma_j\Sigma_k\ p_{X,Y}(x_j, y_k)\ \log_2(p_{X,Y}(x_j, y_k))$

- Properties
  - Symmetry $H(X(s), Y(s)) = H(Y(s), X(s))$
  - Greater than or equal to largest $H(X(s), Y(s)) \geq \max\{H(X(s)), H(Y(s))\}$
  - Less than or equal to sum $H(X(s), Y(s)) \leq H(X(s)) + H(Y(s))$
  - Independent X(s) and Y(s) $H(X(s), Y(s)) = H(X(s)) + H(Y(s))$

Joint Entropy

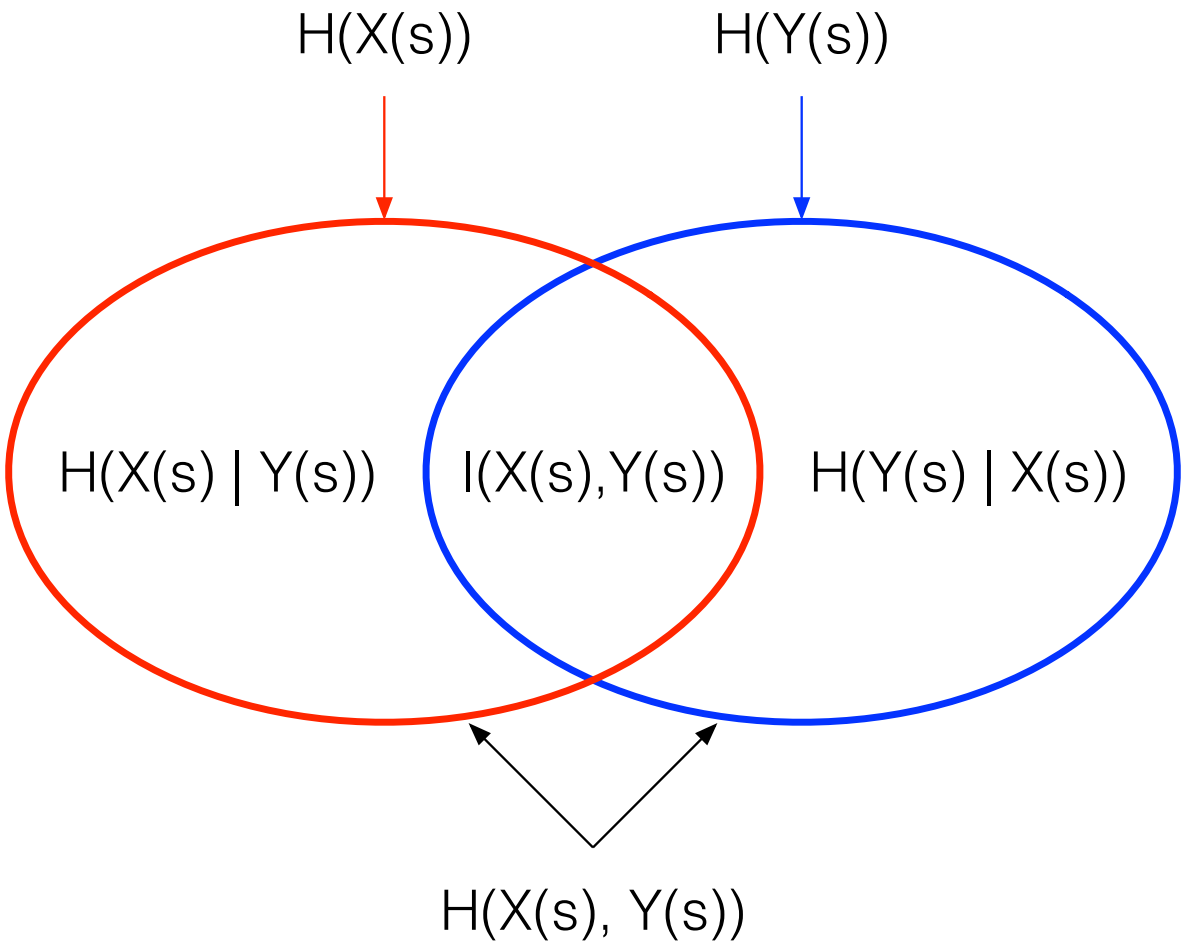


Conditional Entropy

- Definition
  - Informally, the information in the realization of 1 random variable conditioned on all possible values of another random variable
  - $H(X(s) \mid Y(s)) = \Sigma_k \, p_Y(y_k) \, H(X(s) \mid Y(s) = y_k)$
    $= -\Sigma_j \Sigma_k \, p_{X,Y}(x_j, y_k) \log_2(p_{X|Y}(x_j \mid y_k))$

- Properties
  - Information reduction — $H(X(s) \mid Y(s)) \leq H(X(s))$
  - X(s) is completely determined by Y(s) — $H(X(s) \mid Y(s)) = 0$
  - Independent X(s) and Y(s) — $H(X(s) \mid Y(s)) = H(X(s))$
  - Chain rule — $H(X(s) \mid Y(s)) = H(X(s), Y(s)) - H(Y(s))$
  - Bayes' rule — $H(X(s) \mid Y(s)) = H(Y(s) \mid X(s)) - H(Y(s)) + H(X(s))$

Conditional Entropy



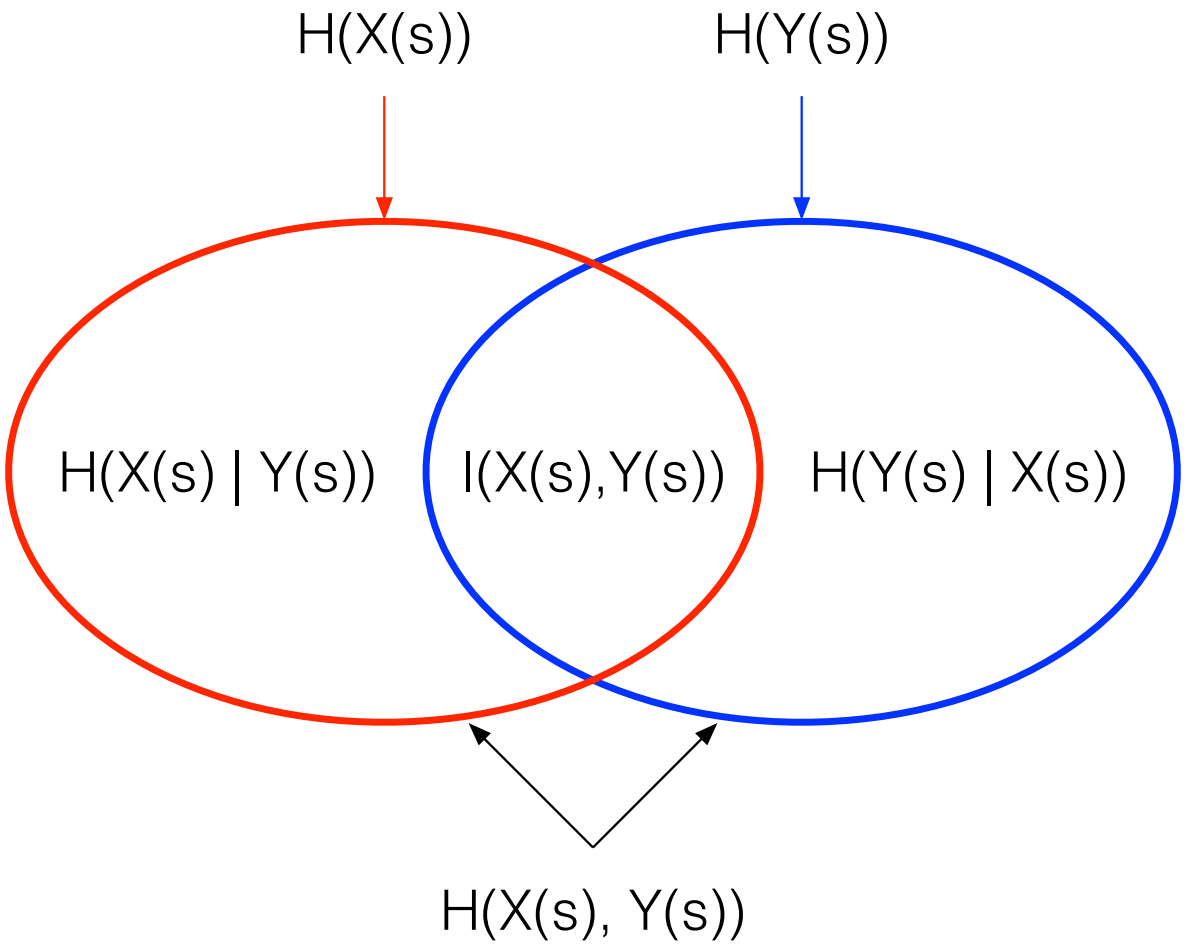
Mutual Information

- Definition
  - Informally, the information obtained about the realization of 1 random variable through the observation of the realization of another random variable; the shared information between realizations of 2 random variables
  - $I(X(s), Y(s)) = \Sigma_j \Sigma_k \, p_{X,Y}(x_j, y_k) \log_2(p_{X,Y}(x_j, y_k) / (p_X(x_j) \, p_Y(y_k)))$

- Properties
  - Self $I(X(s), X(s)) = H(X(s))$
  - Symmetry $I(X(s), Y(s)) = I(Y(s), X(s))$
  - Non negativity $I(X(s), Y(s)) \geq 0$
  - Independent X(s) and Y(s) $I(X(s), Y(s)) = 0$
  - Conditional and joint relationship

$$
\begin{aligned}
I(X(s), Y(s)) &= H(X(s)) - H(X(s) \mid Y(s)) \\
&= H(Y(s)) - H(Y(s) \mid X(s)) \\
&= H(X(s)) + H(Y(s)) - H(X(s), Y(s)) \\
&= H(X(s), Y(s)) - H(X(s) \mid Y(s)) - H(Y(s) \mid X(s))
\end{aligned}
$$

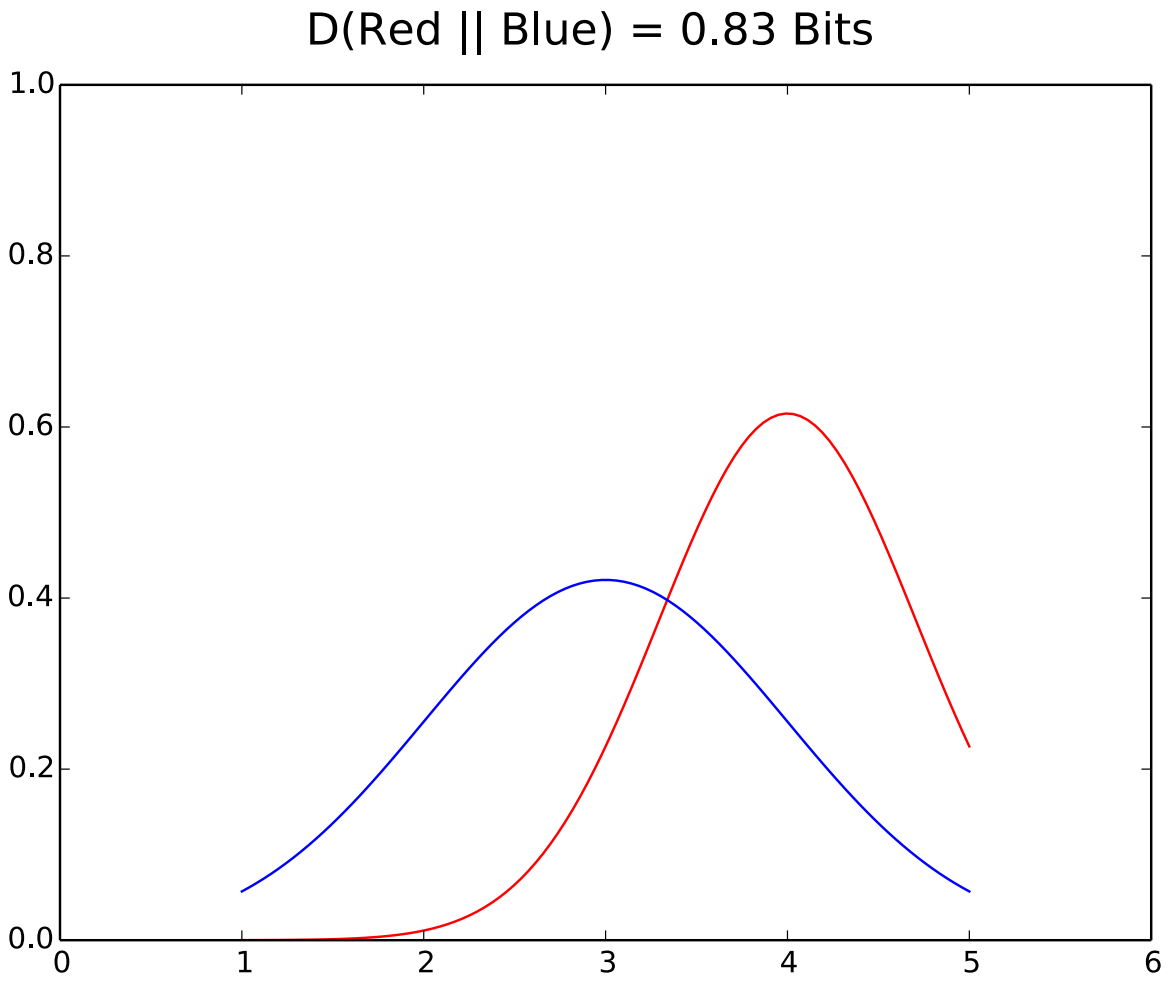
Mutual Information



Kullback Leibler (KL) Divergence

- xNN use: Error calculation for classification networks
- Definition
  - Informally, a non symmetric distance (i.e., divergence) between 2 probability distributions; the amount of information lost when 1 distribution is used to approximate another (nice for an info extracting network to minimize); the expected value of the log difference between 2 distributions
  - $D(X(s) \,||\, Y(s)) = -\sum_k p_X(x_k) \log_2(p_Y(x_k) / (p_X(x_k))),$ if $p_Y(x_k) = 0$ only when $p_X(x_k) = 0$
    $= -\sum_k p_X(x_k) (\log_2(p_Y(x_k)) - \log_2(p_X(x_k)))$
    $= -\sum_k p_X(x_k) \log_2(p_Y(x_k)) + \sum_k p_X(x_k) \log_2(p_X(x_k))$
    $= H_{ce}(X(s), Y(s)) - H(X(s)),$ $H_{ce}(X(s), Y(s))$ is cross entropy
- Notes
  - $D(X(s) \,||\, Y(s)) = 0$ iff $p_X(x_k) = p_Y(x_k)$
  - **For a 1 hot probability mass function $p_X(x_k)$, entropy $H(X(s)) = 0$ and $D(X(s) \,||\, Y(s)) = H_{ce}(X(s), Y(s))$**
  - An option for making it symmetric, define $D(X(s), Y(s)) = (D(X(s) \,||\, Y(s)) + D(Y(s) \,||\, X(s))) / 2$
  - An alternatives for comparing distributions: optimal transport

Kullback Leibler (KL) Divergence



Data Processing Inequality

- Inequality
  - Let Y(s) be a function of X(s) and Z(s) be a function of Y(s) such that X(s) → Y(s) → Z(s)
  - I(X(s), Z(s)) ≤ I(X(s), Y(s))
  - In words: Z(s) cannot have more information about X(s) than Y(s) has about X(s)
  - You never gain information by processing data (but you potentially can make the information that's already there easier to extract)

- Proof
  - $I(X(s), Z(s)) = H(X(s)) - H(X(s) \mid Z(s))$
    $\leq H(X(s)) - H(X(s) \mid Y(s), Z(s))$
    $= H(X(s)) - H(X(s) \mid Y(s))$
    $= I(X(s), Y(s))$

Data Processing Inequality

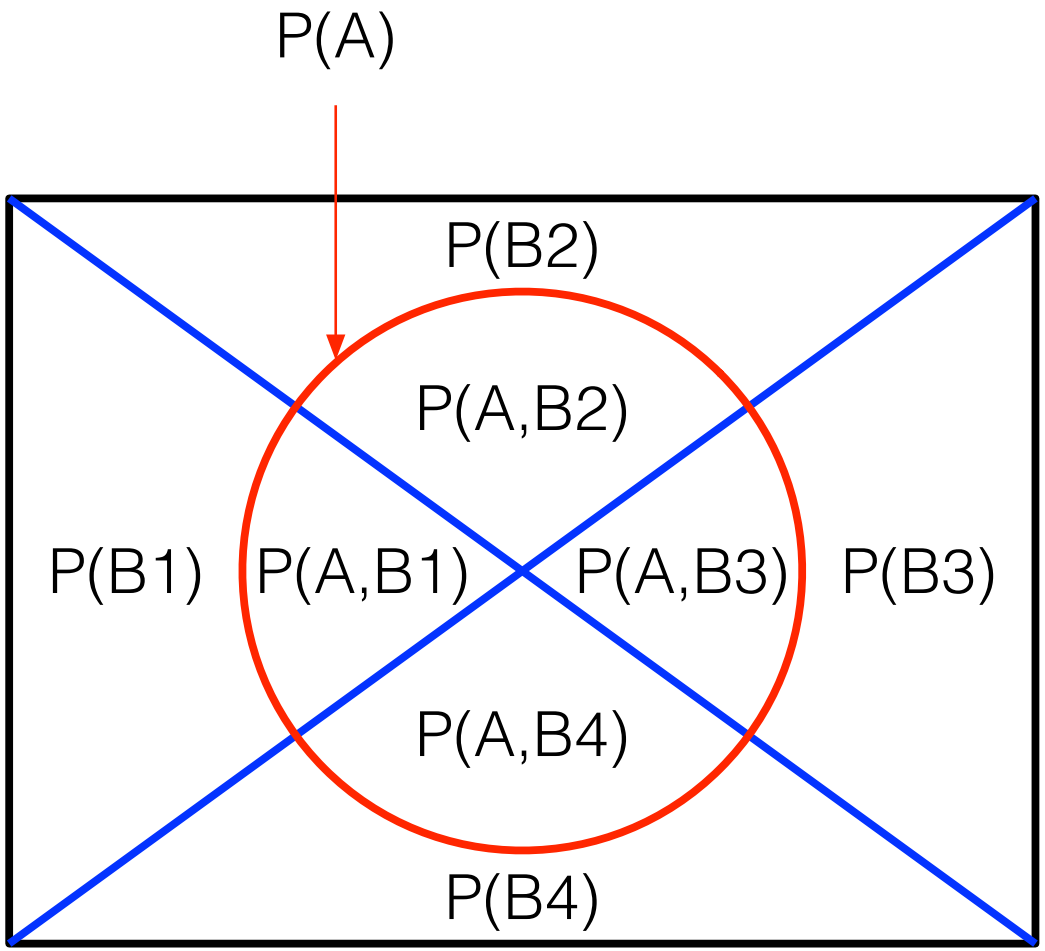
- xNN use: network design guidelines for information extraction
  - Think of a realization of a random variable as a network input containing new information
  - Think of trained filter coefficients as a network input containing past information
  - Processing the input by the network can only lose information (from the data processing inequality)
  - A key in good network design is not to create any fundamental bottlenecks of information mapping from input to output that lose significant amounts / important information (consider the extreme example of a layer zeroing out all feature maps)
  - Note that bottlenecks in residual layers are not fundamental bottlenecks because of the parallel direct path (will discuss later)

Backup

Law Of Total Probability

- Law of total probability
  - Let $\{B_0, ..., B_{K-1}\}$ be a set of disjoint events whose union is the full event space
  - Let A be an event in the same event space
  - The marginal probability of A is $P(A) = \Sigma_k P(A, B_k) = \Sigma_k P(A|B_k) P(B_k)$

Law Of Total Probability



$$P(A) = \text{sum}_k\ P(A, B_k)$$
$$= \text{sum}_k\ P(A \mid B_k)\ P(B_k)$$

Assigning Probabilities To Events

- Example methods for assigning probabilities to events
  - Relative frequency: perform an experiment N times and let N_A be the number of times A occurs
    - $P(A) = N_A / N$
  - Counting: let N_A and N_S be be the number of elements in event A and the sample space, respectively
    - $P(A) = N_A / N_S$
- Fundamental counting principle for events A and B
  - Event A has N outcomes and event B has M outcomes
  - Events A and B have NM outcomes
- Permutations
  - Number of ways to choose and arrange K objects from N objects = $N! / (N – K)!$
- Combinations
  - Number of ways to choose K objects from N objects = $N! / (K! (N – K)!)$

Continuous

- A continuous random variable is a function X with an uncountably infinite range that maps outcomes s from the sample space S to numbers $x \in R$
  - X(s) = x
  - x is a realization of X

- Note that a random variable is (still) not random and it's (still) not a variable
  - The outcome of the experiment s is random
  - The mapping X(s) = x by the random variable (function) to a real number is deterministic

Continuous

- A continuous random variable is described by it's cumulative distribution function that specifies the probability that it's value falls within an interval
  - If the cdf is absolutely continuous then it also has a probability density function (this course will assume this is true to avoid having to use the word measure and weird looking integrals)

- Probability density function
  - Single
    $\int_a^b p_X(x)\,dx = P(a \le X(s) \le b)$
    where $\int p_X(x)\,dx = 1$
  - Joint and conditional
    $p_{X,Y}(x, y) = p_{X|Y}(x|y)\,p_Y(y) = p_{Y|X}(y|x)\,p_X(x)$
  - Marginal
    $p_X(x) = \int p_{X,Y}(x, y)\,dy = \int p_{X|Y}(x|y)\,p_Y(y)\,dy$
  - Independent X and Y
    $p_{X,Y}(x, y) = p_X(x)\,p_Y(y)$
    $p_{X|Y}(x|y) = p_X(x)$

- Cumulative distribution function
  - Single
    $F_X(x) = \int^x p_X(t)\,dt$

Expected Value

- How is it possible to reduce noise?
  - If the space that the signal lives on is much smaller than the space that the noise lives on then some type of dimensionality reduction can be used to reduce noise

- Consider the problem of estimating constant a from observations y corrupted by noise v
  - $y(n) = a + v(n)$, $n = 0, ..., N - 1$
  - If v(n) is 0 mean then averaging y(n) yields an estimate for $a^{hat} = (1/N) \Sigma_n y(n)$
  - In this case averaging is a function that maps from R^N to R (a higher to lower dimensional subspace)
  - It's straightforward to extend this example to higher dimensions (e.g., the next slide)

- This concept of mapping from a higher dimensional to a lower dimensional subspace to remove noise and extract information is also realized by some xNN structures
  - E.g., a xNN based auto encoder

Expected Value

- Linear regression
  - $\mathbf{y} = \mathbf{A}\,\mathbf{x} + \mathbf{e}$, where $\mathbf{e}$ is a $\mathbf{0}$ mean vector random variable representing measurement error
  - $\mathbf{e} = \mathbf{y} - \mathbf{A}\,\mathbf{x}$
  - $\min_{\mathbf{x}} \mathbf{e}^T \mathbf{e} = (\mathbf{y} - \mathbf{A}\,\mathbf{x})^T (\mathbf{y} - \mathbf{A}\,\mathbf{x}) = \mathbf{x}^T \mathbf{A}^T \mathbf{A}\,\mathbf{x} - 2\,\mathbf{y}^T \mathbf{A}\,\mathbf{x} + \mathbf{y}^T \mathbf{y}$
  - $\mathbf{x}^{hat} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$

- Estimator mean
  - $E[\mathbf{x}^{hat}] = E[(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}]$
    $= E[(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T (\mathbf{A}\,\mathbf{x} + \mathbf{e})]$
    $= E[(\mathbf{A}^T \mathbf{A})^{-1} (\mathbf{A}^T \mathbf{A})\,\mathbf{x}] + E[(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{e}]$
    $= E[\mathbf{x}] + (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T E[\mathbf{e}]$
    $= \mathbf{x}$

- Estimator covariance
  - Substitute $\mathbf{y} = \mathbf{A}\,\mathbf{x} + \mathbf{e}$ into the formula for $\mathbf{x}^{hat}$ to get $\mathbf{x}^{hat} = \mathbf{x} + (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{e}$ or $\mathbf{x}^{hat} - \mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{e}$
  - $E[(\mathbf{x}^{hat} - \mathbf{x})(\mathbf{x}^{hat} - \mathbf{x})^T] = E[((\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{e})((\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{e})^T] = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T E[\mathbf{e}\,\mathbf{e}^T]\,\mathbf{A}\,(\mathbf{A}^T \mathbf{A})^{-1} = \sigma_e^2\,(\mathbf{A}^T \mathbf{A})^{-1}$

Examples Of Discrete PMFs

- Bernoulli
  - $p_X(x_k)$ $= 1 - p,$ $x_k = 0, p \in [0, 1]$
    $= p,$ $x_k = 1$
    $= 0,$ elsewhere
  - Mean $= p$
  - Variance $= p\,(1 - p)$

- Uniform
  - $p_X(x_k)$ $= 1 / N,$ $x_k \in \{a, a + 1, \ldots, b\}, b - a + 1 = N$
    $= 0,$ elsewhere
  - Mean $= (a + b) / 2$
  - Variance $= (N^2 - 1) / 12$

Examples Of Continuous PDFs

- Uniform
  - $p_X(x)$ = 1 / (b – a), x ∈ [a, b], a and b finite
    = 0, elsewhere
  - Mean = (a + b) / 2
  - Variance = $(b – a)^2$ / 12, side note: this leads to a famous SNR formula for quantizers

- Gaussian (or normal)
  - $p_X(x) = (1 / (2\pi\sigma^2)^{1/2}) \exp(-(x - \mu_x)^2 / 2\sigma_x^2)$
  - Mean = μ_x
  - Variance = σ_x^2

- xNN use: filter coefficient initialization
  - For initialization a uniform or truncated Gaussian distribution is frequently used

Compression

- xNN uses
  - Minimize the amount of data that needs to be moved around to improve performance (data movement can easily take more power than computation)
  - Minimize or simplify the amount of data that needs to be processed while keeping as much information as possible

- Definitions (assuming optimal decompression)
  - Lossless compression: x → compression → y → decompression → x
  - Lossy compression: x → compression → y → decompression → x + error

- Limits
  - Question: How much lossless compression of data is possible (how small can y be)?
  - Answer: The entropy (information) of the data defines the limit
  - Intuition: What remains after removing all redundancy from the data is information
    But it's not possible to throw away information and exactly recover the original data

Lossy Compression

- Frequently data type / application specific for the largest gains
  - General strategy of hiding reconstruction errors (information loss) in areas that are less noticeable to the user / consumer
  - Examples
    - Audio coding formats
    - Image coding formats
    - Video coding formats

- We've already considered some pre processing methods that can be considered data compression on the input data to the network
  - DFT and keeping L < K basis elements (throwing away the other basis elements)
  - PCA with L < K (throwing away columns)

Lossless Compression

- 2 examples of redundancy
  - Redundancy within a symbol: non uniform symbol distribution
  - Redundancy across symbols: dependencies (e.g., underlying model, correlation, ...)
- 3 examples of how to remove redundancy
  - First remove redundancy within a symbols then remove redundancy across the new symbols
  - First remove redundancy across symbols then remove redundancy within the new symbols
  - Remove redundancy within and across symbols at the same time
- Entropy codes are common for removing redundancy within a symbol
  - Huffman coding
  - Arithmetic coding
- Run length codes are common for removing redundancy across symbols
  - We'll skip this in these slides

Coding Intuition

To transmit the message x_k with the pmf and code in ex 1 it's expected to take 16*(1/16)*4 = 4 bits
To transmit the message x_k with the pmf and code in ex 2 it's expected to take 1*(1/2)*1 + 15*(1/30)*5 = 3 bits

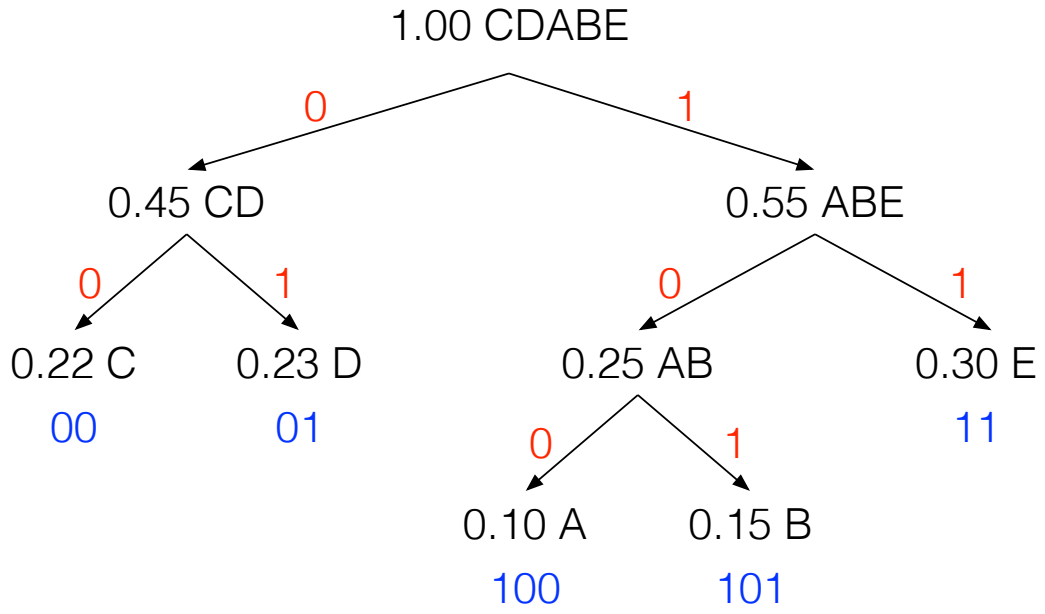
Random variable $X(s) = x_k$	Probability mass function ex 1 $p_X(X(s) = x_k)$	Binary coding ex 1 $C_X(x_k) = c_k$	Probability mass function ex 2 $p_X(X(s) = x_k)$	Binary coding ex 2 $C_X(x_k) = c_k$
0	1/16	0000	1/2	0
1	1/16	0001	1/30	10001
2	1/16	0010	1/30	10010
3	1/16	0011	1/30	10011
4	1/16	0100	1/30	10100
5	1/16	0101	1/30	10101
6	1/16	0110	1/30	10110
7	1/16	0111	1/30	10111
8	1/16	1000	1/30	11000
9	1/16	1001	1/30	11001
10	1/16	1010	1/30	11010
11	1/16	1011	1/30	11011
12	1/16	1100	1/30	11100
13	1/16	1101	1/30	11101
14	1/16	1110	1/30	11110
15	1/16	1111	1/30	11111

Huffman Coding

- Strategy
  - Record symbol probabilities
  - Build a min heap tree bottoms up (bottoms up is the key)
  - Traverse the tree top down and assign 0 / 1 to left / right branches
  - Codes for leaves = branch path are a prefix code
  - Simple table lookup for encoding and state machine for decoding
  - Close to entropy bound for many distributions of interest for independent symbols

Huffman Coding

0.10 A	0.22 C	0.25 AB	0.45 CD	1.00 CDABE
0.15 B	0.23 D	0.30 E	0.55 ABE	
0.22 C	0.25 AB	0.45 CD		
0.23 D	0.30 E			
0.30 E				



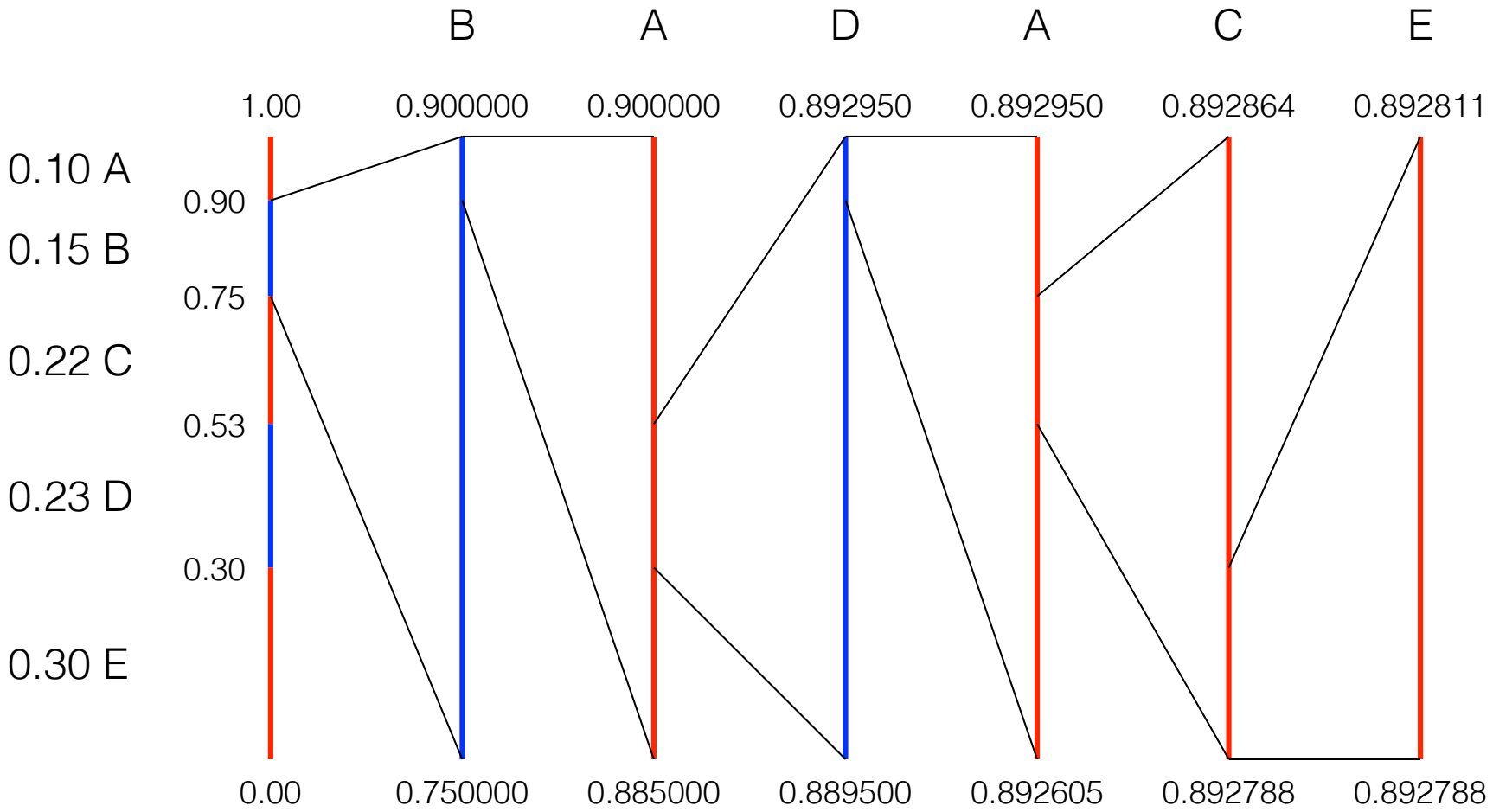
Huffman Coding

		B	A	D	A	C	E
0.10 A	100						
0.15 B	101						
0.22 C	00	101	100	01	100	00	11
0.23 D	01						
0.30 E	11						

Arithmetic Coding

- Strategy
  - Encode a complete message as a single real number
  - Start with intervals proportional to symbol probabilities
  - Rescale top and bottom limit of the interval based on symbol to encode
  - Slightly more complex arithmetic for encoding and decoding (depending on hardware)
  - Optimal in the sense that it achieves the entropy bound for independent symbols

Arithmetic Coding



A Brief Comment On Coding

- Compression and decompression
  - Compression is a function that maps data to ~ information
  - Decompression is a function that maps ~ information to data

- Idea
  - Neural networks are universal function approximators (we'll prove this in the analysis lecture)
  - Why not use them for the compression and decompression functions?

- Many papers have recently been published on this, more are likely
  - https://paperswithcode.com/task/image-compression/latest

Goals

- xNNs are universal function approximators
  - A lot of problems can be cast as a classification problem
  - In practice this is done by having a network predict a pmf over all the possible classes from the input
  - To steer the network towards this mapping a loss function is defined to adapt the parameters in the network towards this prediction during training

- Example: image classification
  - Input image $\mathbf{X}$
    - E.g., 3 x rows x cols
  - Output pmf $\mathbf{y}$
    - Num classes x 1
  - Network predicts $p(\mathbf{y} \mid \mathbf{X})$

Goals

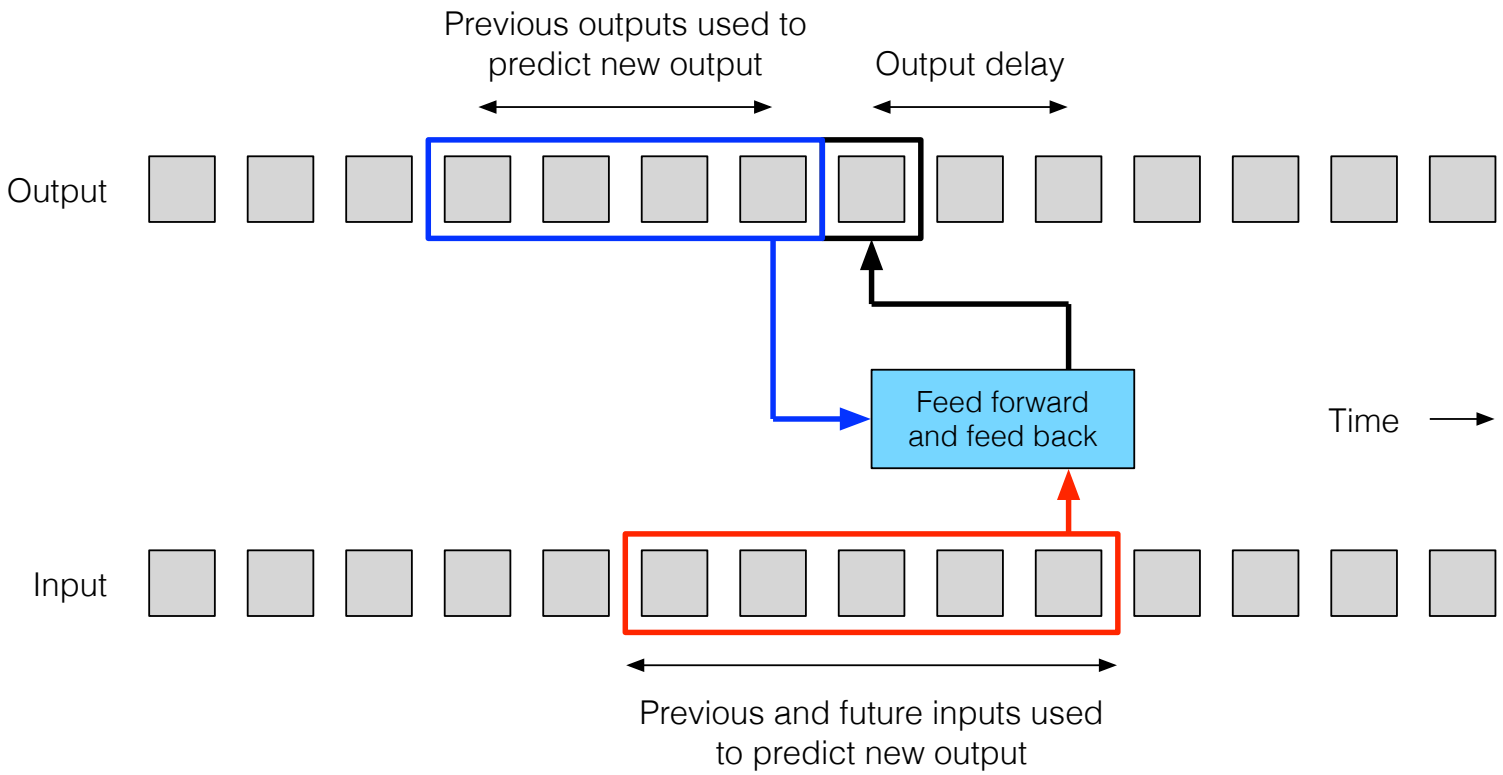
- Example: speech to text transduction
  - Can build a language model that assigns probabilities to sequences of n characters y_0, ..., y_{n-1} using the chain rule of conditional probability
    - $p(y_{n-1}, y_{n-2}, ..., y_1, y_0) = p(y_{n-1} \mid y_{n-2}, ..., y_1, y_0) ... p(y_1 \mid y_0) p(y_0)$
  - The key is next element (e.g., character) prediction from previous predictions
    - $p(y_{n-1} \mid y_{n-2}, ..., y_1, y_0)$
  - This model can be trained on a large offline text and used to generate new text on it's own
    - Standard and neural network based methods can be used for next element prediction
    - But this model doesn't generate text that correspond to a given speech signal
  - To make the model output correspond to a given speech signal the model can also be conditioned on the speech signal **X** (in addition to the previous output predictions)
    - $p(y_{n-1} \mid y_{n-2}, ..., y_1, y_0, \mathbf{X})$
  - A challenge of this type of model is that in practice the true previous outputs are replaced with their estimates which introduces the potential for error propagation (which leads to beam search, ...)
    - $p(y_{n-1} \mid y_{n-2}^{hat}, ..., y_1^{hat}, y_0^{hat}, \mathbf{X})$

Goals

- Example: text to speech transduction
  - Similar in spirit to speech to text transduction though with a model for sound conditioned on a given piece of text
- Example: language to language translation
  - Similar in spirit to speech to text transduction though with a model for text in the target language conditioned on a given piece of text in the source language
- Example: multiple object detection in a video sequence
  - Question: what's the optimal method for using past images in multiple object detection for the present image?
  - A little different than the speech and language cases as there's not a separate source to model on
  - Should the prediction be made FIR style? For accuracy? For computation?
  - Should the prediction be made DFE style? For accuracy? For computation?

Goals

- Things to consider
  - Contribution of previous outputs vs inputs to the prediction of the new output and stability in the presence of errors
  - Delaying the output prediction to allow future inputs to be used in the prediction vs stability from a controllability perspective
  - Infinite vs finite windows of input and output info used to predict the new output
  - Computation
  - Accuracy



Design

- Understanding the implications of network design on the flow of information through the network

Training

- Take the case of predicting an output pmf from an input
  - $P(y \mid x)$

- A goal of training is to maximize this probability as a function of network parameters
  - For a single input: $\arg\max_h p(y_i \mid x_i, h)$, making parameter dependence explicit
  - For all inputs: $\arg\max_h \prod_i p(y_i \mid x_i, h)$, common to maximize the product

- For practicality purposes the problem is typically transformed to minimizing a negative log likelihood
  - Take the log: $\arg\max_h \Sigma_i \log(p(y_i \mid x_i, h))$
  - Minimize the negative: $\arg\min_h -\Sigma_i \log(p(y_i \mid x_i, h))$

Training

- Weight initialization as the application of known information
- Training as the extraction of information from training data
- Error functions to quantify how well the information extraction process worked
- Stochastic variants of gradient descent to update weights
- Regularization to improve generalization to testing data

Implementation

- Simplification of network designs via quantization and other methods to reduce memory, minimize data movement and improve computation efficiency

- Compression of filter coefficients and feature maps to reduce memory, minimize data movement and improve computation efficiency

References

Probability

- Personal communication with T. Lahou
- Random
  - http://www.randomservices.org/random/index.html
- StatLect fundamentals of probability and fundamentals of statistics
  - https://www.statlect.com
- Lecture notes on probability, statistics and linear algebra
  - http://www.math.harvard.edu/~knill/teaching/math19b_2011/handouts/chapters1-19.pdf
- Mysterious stranger: a book of magic
  - https://www.amazon.com/Mysterious-Stranger-Magic-David-Blaine/dp/0812969774/
- Computational optimal transport
  - https://arxiv.org/abs/1803.00567

Information Theory

- A mathematical theory of communication
  - http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf
- Joint entropy, conditional entropy, relative entropy, and mutual information
  - http://octavia.zoology.washington.edu/teaching/429/lecturenotes/lecture3.pdf
- Visual information theory
  - http://colah.github.io/posts/2015-09-Visual-Information/

Application To xNNs

- Mathematics of deep learning
  - https://arxiv.org/abs/1712.04741
- Batch normalization: accelerating deep network training by reducing internal covariate shift
  - https://arxiv.org/abs/1502.03167
- Understanding batch normalization
  - https://papers.nips.cc/paper/7996-understanding-batch-normalization.pdf
- How does batch normalization help optimization?
  - https://arxiv.org/abs/1805.11604
- Minimizing the negative log-likelihood
  - http://willwolf.io/2017/05/18/minimizing_the_negative_log_likelihood_in_english/
- Information theory, inference, and learning algorithms
  - http://www.inference.org.uk/itprnn/book.pdf
- Entropy and mutual information in models of deep neural networks
  - https://arxiv.org/abs/1805.09785