Exponential Models

Alan Yuille

Feb 5 2024

Learning Exponential Distributions

- This lecture describes how to represent probability distributions by exponential models and hoe to learn their parameters by Maximum Likelihood (M) estimation. Next lecture extends this to exponential models with hidden variables which includes famous models like Hidden Markov Models (HMMs) and the Boltzmann Machine as special cases.
- Exponential distributions are a large and important family of parametric probability distributions which are specified by,sufficient statistics, The distributions are derived from statistics by the Maximum Entropy Principle. Most well known probability distributions like Gaussians are members of this family.
- Maximum Likelihood (ML) is used to estimate their parameters from data. ML can be interpreted in terms of finding the distribution which is closest to the empirical distribution of the data. Hence ML finds the best approximation to the data, given the statistics used by the model, and justifies model pursuit which increases the statistics to find distributions which approximate the data better.
- The lecture deals with distributions $P(x)$ where x is typically multivariate $x = (x_1, ..., x_m)$. The distributions are defined over a graph with nodes $1, ..., m$. Often this has a Markov Random Field structure, i.e., with neighborhoods, and the distributions can have closed loops (as in the previous lecture).

Learn Exponential Distributions Overview

- Firstly we describe basic ML for a parametric probability model.
- Secondly we introduce *exponential models* and *sufficient statistics* which give a general form for representing probability models. ML has a very simple and intuitive interpretation for this case (and yields a simple decision rule for binary classification based on the log-likelihood rule).
- Thirdly, we re-interpret ML in terms of making the "best" approximation to the data as measured by the Kullback-Leibler divergence. This has a nice interpretation within *information geometry*. This explains why ML makes sense if you have the wrong model. It also leads to a strategy where you "pursue" the probability model within a space of probability models.
- Finally, as an advanced topic, we describe the maximum entropy principle which enables us to derive the probability models from their statistics giving an alternative perspective.

Learning probability distributions by ML (1)

- Assume a parameterized model for the distribution of form $p(x \mid \theta)$, where θ is the parameter. For example, a Gaussian distribution is specified by $p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ with parameters $\theta = (\mu, \sigma)$.
- We also assume that the training data $\mathcal{X}_N = \{x_1, ..., x_N\}$ is independent identically distributed (iid) from an (unknown) distribution $p(x)$. Then the probability of $\mathcal{X}_N$ is given by $p(\mathcal{X}_N) = p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i)$.Our task is to select a parameterized distribution $p(x|\theta)$ and estimate its parameters θ from $\mathcal{X}_N$.
- *The Maximum Likelihood Estimator* of θ is:

$$\hat{\theta} = \arg\max_{\theta} p(x_1, \ldots, x_N \mid \theta)$$

$$= \arg\min_{\theta} \{-\log p(x_1, \ldots, x_N \mid \theta)\}.$$

Equivalently, $p(x_1, \ldots, x_N \mid \hat{\theta}) \geq p(x_1, \ldots, x_N \mid \theta)$, for all θ.

Learning by ML example

- *Example:* Gaussian distribution.
  $-\log p(x_1, \ldots, x_N \mid \mu, \sigma) = -\sum_{i=1}^{N} \log p(x_i \mid \mu, \sigma)$
  $= \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2} + \sum_{i=1}^{N} \log \sqrt{2\pi}\sigma$
- To estimate the parameters $\theta = (\hat{\mu}, \hat{\sigma})$ we differentiate w.r.t. μ, σ, i.e, maximize $\log p(\mathcal{X}_N|\mu, \sigma)$, which gives:
- $\frac{\partial}{\partial \mu} \log p(x_1, \ldots, x_N \mid \mu, \sigma) = \frac{1}{\sigma^2} \sum_{i=1}^{N}(x_i - \mu)$
- $\frac{\partial}{\partial \sigma} \log p(x_1, \ldots, x_N \mid \mu, \sigma) = \frac{1}{\sigma^3} \sum_{i=1}^{N} (x_i - \mu)^2 - \frac{N}{\sigma}$
- The solution occurs at: $\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$, $\hat{\sigma^2} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$. This is a special case because the ML estimate is given by an analytic formula but for more complex distributions an algorithm will be needed.

Learning by ML

- Instead of ML we can use the Maximum a Posteriori (MAP) estimator if we know a prior $p(\theta)$ for θ. This gives

$$\begin{aligned} \hat{\theta}_{MAP} &= \arg\max_{\theta} \frac{p(\mathcal{X}_N|\theta)p(\theta)}{p(\mathcal{X}_N)} \qquad \text{(note: } p(\mathcal{X}_N) \text{ is independent of } \theta) \\ &= \arg\max_{\theta} \{\log p(\mathcal{X}_N|\theta) + \log p(\theta)\} \\ &= \arg\max_{\theta} \{\sum_{i=1}^{N} \log p(x_i|\theta) + \log p(\theta)\}, \end{aligned} \tag{1}$$

- If N is large, then the prior will have little effect because we have N data terms $\log p(x_i|\theta)$and only one 1 prior term, $\log p(\theta)$. For example, if we are tossing a coin, we may start with a prior (fair coin, or some other), but after a large number of tosses the prior has very little effect and only the data matters,
- Cognitive scientists have examples where humans learn from limited data by exploiting prior knowledge, but these are not relevant to this course. More generally, we can formulate learning by Bayes Decision Theory and include loss functions, particularly if we can to estimate conditional distributions, and we will discuss this later in the course.

Exponential Distributions: Sufficient Statistics

- Exponential distributions are a general way for representing probability distributions $p(.)$ in terms of *sufficient statistics* $\vec{\phi}(.)$ and parameters $\vec{\lambda}$.
- The general form of an exponential distribution is: $p(\vec{x}|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}]} \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x})\}$, where $Z[\vec{\lambda}]$ is the *normalization factor*, $\vec{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_M)$ are the *parameters* and $\vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_M(\vec{x}))$ are the *statistics*. The distribution depends on the data $\vec{x}$ only by the function $\vec{\phi}(\vec{x})$, hence $\vec{\phi}(.)$ is called the sufficient statistics.
- Almost every named distribution can be expressed as an exponential distribution, particularly if you allow hidden variables (next lecture). For example: a Gaussian distribution in 1 dimension can be re-expressed as an exponential model: $\vec{\phi}(x) = (x, x^2) \quad \vec{\lambda} = (\lambda_1, \lambda_2)$, $p(\vec{x}|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}]} e^{\lambda_1 x + \lambda_2 x^2}$.

  This can be translated back to standard Gaussian form $\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$.

$$\begin{cases} \lambda_2 = -\frac{1}{2\sigma^2} \\ \lambda_1 = \frac{\mu}{\sigma^2} \\ Z[\vec{\lambda}] = \sqrt{2\pi}\sigma \exp \frac{\mu^2}{2\sigma^2} \end{cases}$$

Learning Exponential Distributions by ML (1)

- We can learn exponential distributions by MLE. This gives a very intuitive interpretation. *MLE* selects the parameter θ such that the expected statistics $\vec{\phi}(.)$ of the model are equal to the expected statistics of the data.
- We want to maximize with respect to $\vec{\lambda}$:
  $p(\mathcal{X}_N) = p(\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N\}|\lambda) = \prod_{i=1}^{N} \frac{e^{\vec{\lambda}\cdot\vec{\phi}(\vec{x}_i)}}{Z[\vec{\lambda}]}$. This has a simple form because the exponential distribution depends on the data $\vec{x}_i$ only in terms of the function $\vec{\phi}(\vec{x}_i)$, that is, the sufficient statistics.
- ML minimizes
  $$-\log \prod_{i=1}^{N} p(\vec{x}_i|\vec{\lambda}) = -\sum_{i=1}^{N} \log p(\vec{x}_i|\vec{\lambda}).$$
  For exponential distributions this corresponds to minimizing
  $$F[\vec{\lambda}] = N \log Z[\vec{\lambda}] - \sum_{i=1}^{N} \vec{\lambda}\cdot\vec{\phi}(\vec{x}_i).$$
  $F[\vec{\lambda}]$ is a convex function of $\vec{\lambda}$ so its minimum can be found by differentiation.

Learning Exponential Distributions by ML (2)

- Differentiating $F[\vec{\lambda}]$ with respect to $\vec{\lambda}$ gives $\frac{\partial F}{\partial \vec{\lambda}} = N \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}) - \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)$. The minimum occurs at $\{\hat{\vec{\lambda}}\}$ such that the expected statistics $\sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\hat{\vec{\lambda}})$ is equal to the data statistics $(1/N)\vec{\phi}(\vec{x}_i)$.
- This result requires differentiating the log of the normalization function $Z[\vec{\lambda}] = \sum_{\vec{x}} e^{\vec{\lambda}\cdot\vec{\phi}(\vec{x})}$. This yields $\frac{\partial \log Z[\vec{\lambda}]}{\partial \vec{\lambda}} = \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda})$.
- Proof: $\frac{\partial \log Z[\vec{\lambda}]}{\partial \vec{\lambda}} = \frac{1}{Z[\vec{\lambda}]} \frac{\partial Z[\vec{\lambda}]}{\partial \vec{\lambda}} = \frac{1}{Z[\vec{\lambda}]} \sum_{\vec{x}} \vec{\phi}(\vec{x}) e^{\vec{\lambda}\cdot\vec{\phi}((\vec{x})} = \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda})$.
- For example, for a Gaussian the model statistics $\vec{x}$ are $\int d\vec{x}\vec{x} \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\{-(1/2\sigma^2)(\vec{x}-\vec{\mu})^2\} = \vec{\mu}$. The data statistics are $1/N \sum_{i=1}^{N} \vec{x}_i$.

Learning Exponential Distributions by ML (3)

- For simple exponential distributions, like the Gaussian, the expected statistics of the model can be computed analytically giving a function $f(\lambda) = \sum_x \phi(x)p(x|\lambda)$. Then MLE reduces to solving the equation

$$\lambda = f^{-1}(\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)).$$

- But for other exponential distributions we cannot compute $\sum_x \phi(x)p(x|\lambda)$ as a function of λ. Instead we require an algorithm to minimize $F[\vec{\lambda}]$ with respect to $\vec{\lambda}$.
- $F[\vec{\lambda}]$ is a convex function of $\vec{\lambda}$ (its Hessian $\frac{\partial^2 F}{\partial\vec{\lambda}\partial\vec{\lambda}}$ can be computed and shown to be positive semi-definite using the Cauchy-Schwarz inequality). Hence has only a single minimum which can be found by steepest descent, or variants, using the gradient $\frac{\partial F}{\partial \vec{\lambda}}$.
- Unfortunately, at each iteration step, this requires computing the expectation of the statistics $\sum_{\vec{x}} \vec{\phi}(\vec{x})p(\vec{x}|\vec{\lambda})$ which may not be possible (if the distribution is defined over a graph with closed loops) and will need to be approximated.

Convexity of $F[\vec{\lambda}]$, Uniqueness of MLE, and Iterative Algorithms (2)

- Algorithm 1: Steepest Descent:
- $\vec{\lambda}^{t+1} = \vec{\lambda}^t - \Delta\{\sum_{\vec{x}} \vec{\phi}(\vec{x})p(\vec{x}|\vec{\lambda}^t) - \frac{1}{N}\sum_{i=1}^N \vec{\phi}(\vec{x}_i)\}$. Here Δ is a "time step" constant.
- The continuous form of steepest descent is the differential equation $\frac{d\vec{\lambda}}{dt} = -\frac{1}{N}\frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}}$. We approximate $\frac{d\vec{\lambda}}{dt}$ by $\frac{\vec{\lambda}^{t+1} - \vec{\lambda}^t}{\Delta}$ we compute $\frac{1}{N}\frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}} = \sum_{\vec{x}} \vec{\phi}(\vec{x})p(\vec{x}|\vec{\lambda}) - \frac{1}{N}\sum_{i=1}^N \vec{\phi}(\vec{x}_i)$. Convergence of "differential steepest descent" follow by $\frac{dF}{dt} = \frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}}\frac{d\vec{\lambda}}{dt}$ (the chain rule) which yields $\frac{dF}{dt} = -\frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}} \cdot \frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}} \leq 0$. So the algorithm converges to the unique (by convexity) value of λ where $\frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}} = 0$. The choice of Δ is important. If Δ is too large, then the discrete equation may poorly approximate the continuous version, and so convergence may not occur. But if Δ is too small, then the algorithm can be very slow.

Convexity of $F[\vec{\lambda}]$, Uniqueness of MLE, and Iterative Algorithms (3)

- Algorithm 2: Generalized Iterative Scaling. This algorithm is similar to steepest descent, but does not need a time step parameter Δ. This can be derived from variational bounding and CCCP.
- $\vec{\lambda}^{t+1} = \vec{\lambda}^{t} - \log \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}^{t}) + \log \frac{1}{N} \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)$
- Both algorithms are guaranteed to converge to the correct solution independent of the starting point λ^0 (provided Δ is sufficiently small).
- Both algorithms require computing the quantity: $\sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x} \mid \vec{\lambda}^{t})$ for each iteration step, which is difficult to perform numerically for some distributions. In that case, stochastic sampling methods like Markov Chain Monte Carlo (MCMC) may be used. We will return to this issue in later lectures.

Examples of learning Exponential Distributions: Gaussian Distribution

- The Gaussian distribution has a density function

$$p(x|\vec{\lambda}) = \frac{1}{\sqrt{2\pi\sigma}} e^{\dfrac{-(x-\mu)^2}{2\sigma^2}}.$$

  Let its statistics be $\vec{\phi}(x) = (x, x^2)$. Note: in the general case with N dimensions we would have N-dimensional vectors $\vec{x}$ and statistics $\vec{\phi}(\vec{x}) = (\vec{x}, \vec{x}\vec{x}^T)$.

- The model statistics have to be equal to the data statistics:

$$\sum_x p(x|\vec{\lambda})(x, x^2) = \frac{1}{N} \sum_{i=1}^{N} (x_i, x_i^2).$$

  Note: Really should be $\int p(\vec{x}|\vec{\lambda}) dx$ for Gaussian.

- Left-hand side of the equation: $\int p(x|\vec{\lambda})x = \mu$ and $\int p(x|\vec{\lambda})x^2 = \mu^2 + \sigma^2$. Hence, $\hat{\mu} = \frac{1}{N}\sum_{i=1}^N x_i$ and $\hat{\mu}^2 + \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^N x_i^2$, so $\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^N (x_i - \hat{\mu})^2$, which are the estimators for mean and variance.

Examples of learning Exponential Distributions: Letter Distribution

- Let x be a letter of the alphabet, $x \in \mathcal{A} = \{a, b, c, d, \cdots, y, z\}$. The probability of each letter can be represented by an exponential distribution $p(x) = (1/Z(\lambda)) \exp\{\vec{\lambda} \cdot \vec{\phi}(x)$ where $\vec{\phi}(x) = (\delta_{x,a}, \delta_{x,b}, \cdots, \delta_{x,z})$, with $\delta_{x,a} = 1$ if $x = a$, $\delta_{x,a} = 0$ otherwise. For instance, if the letter is c then $\vec{\phi}(c) = (0, 0, 1, 0, 0, \cdots, 0)$.
- For a given dataset of letters $\mathcal{X}_N = \{x_1, \cdots, x_N\}$, the data statistics are: $\frac{1}{N}\sum_{i=1}^N \vec{\phi}(x_i) = \left(\frac{\#\text{a's}}{N}, \frac{\#\text{b's}}{N}, \cdots, \frac{\#\text{z's}}{N}\right)$, where $\#\text{a's} = \sum_{i=1}^N \delta_{x_i,a}$ is the number of a's in the dataset. The parameters of the distribution are $\vec{\lambda} = (\lambda_a, \lambda_b, \cdots, \lambda_z)$.
- Hence the exponential distribution representing the dataset is of form: $p(x|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}]} e^{\lambda_a \delta_{x,a} + \cdots + \lambda_z \delta_{x,z}}$, where $Z(\vec{\lambda}) = e^{\lambda_a} + \ldots + e^{\lambda_z}$.
- The expected value of the statistics can be computed to be: $\sum_x p(x|\vec{\lambda})\delta_{x,a} = \frac{1}{Z[\vec{\lambda}]} e^{\lambda_a}$. Hence the ML estimator, is obtained by solving equations: $e^{\lambda_a}/(e^{\lambda_a} + \ldots + e^{\lambda_z}) = \frac{\#\text{a's}}{N}, \ldots, e^{\lambda_z}/(e^{\lambda_a} + \ldots + e^{\lambda_z}) = \frac{\#\text{z's}}{N}$. Notice that there is an ambiguity in the λ's (i.e. we can send $(\lambda_a, \ldots, \lambda_z) \mapsto (\lambda_a + K, \ldots, \lambda_z + K)$ where K is a constant without altering the solution. Hence we can resolve the ambiguity by setting $\hat{\lambda}_a = \log \#\text{a's} - \log N$, $\hat{\lambda}_b = \log \#\text{b's} - \log N$, $\cdots$, $\hat{\lambda}_z = \log \#\text{z's} - \log N$, with $Z[\hat{\lambda}] = \frac{\#\text{a's}}{N} + \frac{\#\text{b's}}{N} + \cdots + \frac{\#\text{z's}}{N} = 1$.

Kullback-Leibler and Approximate Distributions

- Here is an alternative viewpoint on ML learning of distributions which gives a deeper understanding. In particular, it shows that MLE gives the "best" approximation even if it is not the correct model for the data.
- First we define the Kullback-Leibler (KL) divergence $D(f(.)||p(.|\vec{\lambda}))$ between distributions $f(.)$ and $p(.|\lambda)$ defined by: $D(f(.)||p(.|\vec{\lambda})) = \sum_{\vec{x}} f(\vec{x}) \log \frac{f(\vec{x})}{p(\vec{x}|\vec{\lambda})}$.
- KL has the property that $D(f||p) \geqq 0 \quad \forall f, p \; D(f||p) = 0,$ if, and only if, $f(x) = p(\vec{x}|\vec{\lambda})$. In general the larger $D(f||p)$ the bigger the difference between $f(.)$ and $p(.)$. For small $D(f||p)$ it can be shown that $D(f||p) \approx (1/2)\sum_x (f(x) - p(x))^2$. (This involves setting $p(x) = f(x) + \epsilon(x)$ and doing a Taylor series expansion of $D(F||p)$).
- So, $D(f||p)$ is a measure of the similarity between $f(\vec{x})$ and $p(\vec{x}|\vec{\lambda})$ (not exactly, because it is not symmetric).
- We can write, $D(f||p) = \sum_{\vec{x}} f(\vec{x}) \log f(\vec{x}) - \sum_{\vec{x}} f(\vec{x}) \log p(\vec{x}|\vec{\lambda})$, where: $\sum_{\vec{x}} f(\vec{x}) \log f(\vec{x})$ is independent of $\vec{\lambda}$ so minimizing $D(f||p)$ with respect to $\vec{\lambda}$ corresponds to minimizing $-\sum_{\vec{x}} f(\vec{x}) \log p(\vec{x}|\vec{\lambda})$.

Relation to ML

- Recall that MLE minimizes $-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i|\lambda)$. Next, we define the *empirical distribution* of the data $\{\vec{x}_i : i = 1..N\}$. (This is a special case of Parzen windows, later next lecture). $f(x) = \frac{1}{N}\sum_{i=1}^{N} I(x = x_i)$. Here $I(x = x_i)$ is the indicator function ($= 1$ if $x = x_i$, $= 0$ otherwise).
- In this, minimizing the KL divergence corresponds to minimizing:

$$-\sum_{\vec{x}} f(\vec{x})\log p(\vec{x}|\vec{\lambda}) = -\sum_{\vec{x}}\frac{1}{N}\sum_{i=1}^{N} I(\vec{x} = \vec{x}_i)\log p(\vec{x}|\vec{\lambda}) = -\frac{1}{N}\sum_{i=1}^{N}\log p(\vec{x}_i|\vec{\lambda}),$$

  which is the same criterion as ML. This proves the claim:
- Claim: ML estimation of $\vec{\lambda}$ is equivalent to minimizing $D(f||p(\vec{x}|\vec{\lambda}))$ w.r.t. $\vec{\lambda}$, where $f(\vec{x})$ is the empirical distribution of the data. Hence, we can justify ML (for exponential distributions) as obtaining the distribution of form $\frac{1}{Z[\vec{\lambda}]}e^{\vec{\lambda}\cdot\phi(\vec{x})}$, which is the best approximation of the data. ML is meaningful even if the model is not the correct one, but only an approximation.

A Geometric Interpretation: Information Geometry

- Information geometry (Shun'ichi Amari, 1980) applies methods of differential geometry to probability distributions. $p(\vec{x}|\vec{\theta}) = \frac{e^{\vec{\lambda}\cdot\vec{\psi}(\vec{x})}}{Z[\vec{\lambda}]}$ defines a sub-manifold of distributions, where $\vec{\lambda}$'s being are coordinates in the manifold. Minimizing $D(f||p)$ w.r.t. $\vec{\lambda}$ is finding a distribution p closest to f in the sub-manifold.
- This also motivates the idea of *model pursuit* as a way to obtain better approximations to the true distribution: (1) Start by doing ML on an exponential distribution with statistic $\vec{\phi}(\vec{x})$. Get the best approximation. (2) Get a better approximation by using more complex statistics, e.g, $\vec{\phi}_1(\vec{x})$, $\vec{\phi}_2(\vec{x})$ with parameters $\vec{\lambda}_1, \vec{\lambda}_2$. (3) Proceed by using incrementally complex statistics. (See Della Pietra et al, S-C Zhu et al.)
- From Amari's perspective this corresponds to selecting submanifolds, which are increasing closer to the data, by using better statistics. This will be illustrated later by Shannon's work on learning the probability of sequences of letters.

Letters example, Shannon's pursuit of letter sequences I

- Earlier, we discussed a dataset of single letters. In this subsection, let us consider data which consists of pairs of letters: $\mathcal{X}_N = \{(x_1^1, x_2^1), (x_1^2, x_2^2), \cdots, (x_1^N, x_2^N)\}$. Let us define a first model which assumes independence between letters: $p(x_1, x_2) = p(x_1)p(x_2)$. The model is exponential, as before: $p(x) = \frac{1}{Z[\vec{x}]} e^{\vec{\lambda}\vec{\phi}(x)}$.
- This gives best fit – in the Kullback-Leibler sense – to data, using statistics $\vec{\phi}_1(\vec{x}_1, \vec{x}_2) = \vec{\phi}(\vec{x}_1) + \vec{\phi}(\vec{x}_2)$. But we can use a better statistic $\vec{\phi}_2(x_1, x_2)$ which considers the pairwise frequencies of letters:

$$\vec{\phi}(x_1, x_2) = \begin{pmatrix} \delta_{x_1,a}\delta_{x_2,a} & \delta_{x_1,a}\delta_{x_2,b} & \cdots & \delta_{x_1,a}\delta_{x_2,z} \\ \delta_{x_1,b}\delta_{x_2,a} & \delta_{x_1,b}\delta_{x_2,b} & \cdots & \delta_{x_1,b}\delta_{x_2,z} \\ \cdots & & & \\ \delta_{x_1,z}\delta_{x_2,a} & \delta_{x_1,z}\delta_{x_2,b} & \cdots & \delta_{x_1,z}\delta_{x_2,z} \end{pmatrix}$$

This second model, with $\vec{\phi}_1(\vec{x}_1, \vec{x}_2)$, gives a better fit to the data the first model because it is better at capturing the pairwise regularities which exist in English. E.g, in English "qu" is frequent but "qz" is impossible.

Letters example, Shannon's pursuit of letter sequences II

- Shannon – the founder of information theory and a pioneer of AI – tried to learn a probability model of sequences of letters. His first model used only the frequency of letters. This is a poor model for text sequences and random sampling yields sequences which differ greatly from typical text sequences in natural sentences. A probability model using the statistics of letter pairs (see last slide) does better but is till unrealistic. Models with third order and higher statistics give increasingly realistic text sequences.
- Shannon evaluated these probability models by comparing them to humans. To do this he used the models $P(x_1, x_2, .., x_m)$ to predict the last letter x_m by sampling from $P(x_m|x_1, ..., x_{m-1})$. He compared these predictions to those made by human subjects. More specifically, he compared the entropy of the human predictions to the entropy of the model (see next slide).
- His approach cannot be extended to learn models of the long sequences required by LLMs (GPTs). For letters, we have 26^T statistics for sequences of T letters and hence need to learn 26^T parameters, which is impossible. So LLMs have to use very different types of probability distributions which will be described later in the course.

Letters example, Shannon's pursuit of letter sequences III

- The entropy of a distribution $p(\vec{x})$ is a measure of the randomness of the distribution:
$$H[p] = -\sum_{x} p(\vec{x}) \log p(\vec{x})$$
The most random distribution, with highest entropy, is the uniform distribution. The distribution with lowest entropy is $P(\vec{x};\vec{x}_0) = \delta(\vec{x}-\vec{x}_0)$ and has probability 1 for one state $\vec{x}_0$ and probability 0 for the others.
- Entropy measures the information we expect to obtain by obtaining a sample $\vec{x}$ from a distribution $p(\vec{x})$. WE get a lot of information from a sample if the distribution is uniform but we get no information is the distribution has zero entropy because we know what the sample will be before we have seen it.
- Information Theory, developed by Shannon, says we should encode a signal $\vec{x}$ by the negative logarithm of their probability $-\log p(\vec{x})$. This means that frequent signals ($p(\vec{x})$ big) have short codes and infrequent signals ($p(\vec{x})$ small) have long codes. The expected code length is the entropy $-\sum_{\vec{x}} p(\vec{x}) \log p(\vec{x})$. But this is a separate subject.

The Probability of the Data and the Entropy of the Distribution

- If we have learnt a distribution $p(\vec{x}|\hat{\vec{\lambda}})$ by ML from data $\{\vec{x}_i : i = 1..N\}$, then we compute the probability of the data to be $\prod_{i=1}^{N} P(\vec{x}_i|\hat{\vec{\lambda}}) = \exp\left\{\hat{\vec{\lambda}} \cdot \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i) - N \log Z[\hat{\vec{\lambda}}]\right\}$
- The entropy of $p(\vec{x}|\hat{\vec{\lambda}})$ is $-\sum_{\vec{x}} p(\vec{x}|\hat{\vec{\lambda}}) \log p(\vec{x}|\hat{\vec{\lambda}}) =$ $\log \vec{Z}[\hat{\vec{\lambda}}] - \sum_{\vec{x}} \hat{\vec{\lambda}}\vec{\phi}(\vec{x})p(\vec{x}|\hat{\vec{\lambda}}) = \log \vec{Z}[\hat{\vec{\lambda}}] - \frac{1}{N}\sum_{i=1}^{N} \hat{\vec{\lambda}}\vec{\phi}(\vec{x}_i)$.
- The probability of the data, given the estimated parameter $\hat{\vec{\lambda}}$ is $\prod_{i=1}^{N} p(\vec{x}_i|\hat{\vec{\lambda}}) = \exp\{-N\mathcal{H}[p(\vec{x}|\hat{\vec{\lambda}})]\}$ and hence depends only on the entropy of the models. If the entropy of $p(\vec{x}|\hat{\vec{\lambda}})$ is large then we cannot predict the data well. But this depends on two issues: (i) is our model a good fit to this data?, and (ii) is the data inherently unpredictable?
- Shannon measures the inherent unpredictability of the data by asking humans to predict the next letters and computing the entropy of their predictions. He compares this to predictions of models learnt with higher order statistics. He show that the entropy of probability models decreases when the order of the statistics increases since the models predict the data better. His work is an example of model pursuit because he increases the complexity of the statistics and shows that they predict the data better.

Maximum Entropy (1)

- An alternative perspective of learning, motivated by the question – how to get to distributions from statistics? Where do exponential distributions come from? E.T. Jaynes claimed (1957) that exponential distributions come from a maximum entropy principle. Suppose we measure some statistics $\vec{\phi}(\vec{x})$, what distribution does it correspond to? This is an ill-posed problem (the solution is not unique), so we have to make some assumptions.
- We have data $\{\vec{x}_1, \cdots, \vec{x}_N\}$ and we have statistics $\vec{\phi}(\vec{x})$ of the data. How to justify a distribution like $p(x) = \frac{1}{Z[\vec{\lambda}]} e^{\vec{\lambda} \cdot \phi(\vec{x})}$? And how to justify using ML to get $\vec{\lambda}$?
- Entropy was discovered, or invented, by physicists. It can be shown that the entropy of a physical system always increases (with plausible assumptions). This is called the Second Law of Thermodynamics. It explains why a cup can break into many pieces (if you drop it), but a cup can never be created by its pieces suddenly joining together. Thermodynamics was discovered in the early 19^{th} century, and shows that it is impossible to design an engine that can create energy.

Maximum Entropy (2)

- Example 1 : Suppose $\vec{x}$ can take N states: $\vec{\alpha}_1, \vec{\alpha}_2, ..., \vec{\alpha}_N$. Let $p(\vec{x} = \vec{\alpha}_1) = 1 \quad p(\vec{x} = \vec{\alpha}_j) = 0, \qquad j = 2, ..., N$.
- Then the entropy of this distribution is zero, because we know that $\underline{x}$ has to take value $\vec{\alpha}$, before we observe it. The entropy is $-0 \log 0 + (N-1)\{1 log 1\} = 0$, because $0 \log 0 = 0$ and $1 \log 1 = 0$ (take the limit of $x \log x$ as $x \mapsto 0$ and $x \mapsto 1$). No information is gained by observing the sample, because we know it can only be $\vec{\alpha}$.
- Example 2: $p(\vec{x} = \vec{\alpha}_j) = \frac{1}{N}, \qquad j = 1, ..., N$. Then $H(p) = -N \times \frac{1}{N} \log(\frac{1}{N}) = \log N$. This is the maximum entropy distribution. Note that the maximum entropy distribution is *uniform* – all states x are equally likely.

Maximum Entropy (3)

- The Maximum Entropy Principle. Given statistics $\phi(\vec{x})$ with observed value $\vec{\psi} = \frac{1}{N}\sum_{i=1}^{N}\phi(\vec{x}_i)$, choose the distribution $p(\vec{x})$ to maximize the entropy subject to constraints (Jaynes, 1957):

$$-\sum_{\vec{x}} p(\vec{x})\log p(\vec{x}) + \mu\{\sum_{\vec{x}} p(\vec{x}) - 1\} + \vec{\lambda}\cdot\{\sum_{\vec{x}} p(\vec{x})\phi(\vec{x}) - \vec{\psi}\}$$

  where μ, λ are lagrange multipliers which impose the constraints on $p(\vec{x})$:
- We differentiate with respect to $p(\vec{x})$, $\frac{\delta}{\delta p(\vec{x})}$, and obtain:

$$-\log p(\vec{x}) - 1 + \mu + \vec{\lambda}\cdot\vec{\phi}(\vec{x}) = 0.$$

- This gives a solution of form $p(\vec{x}|\vec{\lambda}) = \frac{\exp^{\vec{\lambda}\cdot\vec{\phi}(\vec{x})}}{Z[\vec{\lambda}]}$, where the parameters $\vec{\lambda}, Z[\vec{\lambda}]$ are chosen to satisfy the constraints:
  $\sum_{\vec{x}} p(\vec{x}) = 1, \Rightarrow Z[\vec{\lambda}] = \sum_{\vec{x}} \exp^{\vec{\lambda}\cdot\vec{\phi}(\vec{x})}$
  $\sum_{\vec{x}} p(\vec{x})\phi(\vec{x}) = \vec{\psi}, \Rightarrow \vec{\lambda}$ is chosen s.t. $\sum_{\vec{x}} p(\vec{x}|\vec{\lambda})\phi(\vec{x}) = \vec{\psi}$
- Hence the maximum entropy principle obtains exponential distributions from statistics and estimates their parameters consistent with maximum likelihood. So the maximum entropy principle is equivalent to choosing an exponential family of distributions and estimating the λ parameters by maximum likelihood.

Back to Vision

- S-C Zhu had a couple of papers in the 1990's which used exponential distributions. One paper (handout) modelled texture where the ststistics were the histograms of Gabor filters and model pursuit was used to select which Gabors to use (i.e. which statistics to use). This worked fairly well for homogeneous texture (but simpler image-patch models by Efros and Freeman were as effective).
- Another paper showed that the weak membrane model (the weak prior) could be obtained from the measured statistics of the first order derivatives of the image. This was conceptually very interesting. But it also showed limitations of the weak membrane model since it did not capture the statistics of the higher order derivatives (which are also very similar between images).
- In both papers, learning the parameters of the exponential models required using MCMC algorithms to compute the expected statistics of the models in order to match them to the statistics of the data (because the models were two dimensional). The Della Pietra paper (handout) also did model pursuit but their application was language so they could use dynamic programming algorithms for inference.