Lecture 4. Learning Exponential Models

Prof. Alan Yuille

Spring 2014

Outline

1. Learning probability distributions by Maximum Likelihood (ML)
2. Exponential distributions, sufficient statistics, and ML learning
3. Kullback-Leibler divergence, learning "approximate" distributions
4. Advanced topics: Maximum Entropy Principle, Model Pursuit; Model Selection

This lecture deals with learning probability distributions like $p(x), p(x, y)$. This can be applied to the classification task – i.e. learn the conditional probabilities $p(x|y = 1), p(x|y = -1), p(y)$. Then apply Bayes Decision Theory to obtain a decision rule.

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. 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 and gives another perspective. (It is surprising that such a simple idea as ML leads to these rich interpretations.)

1 Learning probability distributions by ML

The basic idea of ML was introduced a century ago by Fisher.

Assume a parameterized model for the distribution of form $p(x \mid \theta)$, θ : model parameter. For example, a Gaussian distribution: $p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, $\theta = (\mu, \sigma)$.

We also assume that the data $\mathcal{X}_N = \{x_1, ..., x_N\}$ is independent identically distributed (iid) from an (unknown) distribution $p(x)$. Using the product for independence, $p(\mathcal{X}_N) = p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i)$. Now assume that $p(x)$ is of form $p(x|\theta)$ for some θ which we have to estimate.

The Maximum Likelihood Estimator 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.

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:

$\frac{\partial}{\partial \mu} \log p(x_1, \ldots, x_N \mid \mu, \sigma) = \frac{1}{\sigma^2} \sum_{i=1}^{N} (x_i - \mu)$

$\frac{\partial}{\partial \sigma} \log p(x_1, \ldots, x_N \mid \mu, \sigma) = -\frac{1}{\sigma^3} \sum_{i=1}^{N} (x_i - \mu)^2 - \frac{N}{\theta}$

The solution occurs at:

$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$

$\hat{\sigma^2} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$

Note that the Gaussian is a special case because it gives a simple analytic formula for $\hat{\mu}, \hat{\sigma}^2$.

This illustrates the ML estimator. What happens with the Maximum a Posteriori

(MAP) estimator? If we use a prior $p(\theta)$, we have:

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

where we have N data terms and 1 prior term, $\log p(\theta)$. If N is large, then the prior will have little effect, except in special cases. 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 doesn't have much effect, and what really matters is the number of heads and tails obtained.

We can also use loss functions and all the machinery of Bayes Decision Theory. In practice, loss functions are often not used when learning probability distributions – but they are used for learning classifiers (later in the course).

2 Exponential Distributions

2.1 Sufficient statistics

This section introduces exponential distributions. These 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 – see later in the course).

Example: For a Gaussian distribution in 1 dimension:
$\vec{\phi}(x) = (x, x^2) \qquad \vec{\lambda} = (\lambda_1, \lambda_2)$
$p(\vec{x}|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}]} e^{\lambda_1 x + \lambda_2 x^2}, \qquad$ compare to $\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

Translation:

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

Similar translations into exponential distribution can be made for Poisson, Beta, Dirichlet, and almost all distributions (some require hidden/latent/missing variables – see later in the course).

2.2 Learning Exponential Distributions by ML

We can learn exponential distributions by MLE. This gives a very suitable interpretation. MLE selects the parameter such that the expected statistics of the model (a function of $\vec{\lambda}$) are equal to the expected statistics of the data.

We want to maximize with respect to $\vec{\lambda}$:

$$p(\mathcal{X}_N) = p(\{\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_N\}) = \prod_{i=1}^{N} \frac{e^{\vec{\lambda}\cdot\vec{\phi}(\vec{x}_i)}}{Z[\vec{\lambda}]}$$

This has a very nice form, which occurs 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.

An important factor is that the normalization term $Z[\vec{\lambda}]$ is a function of $\vec{\lambda}$, $Z[\vec{\lambda}] = \sum_{\vec{x}} e^{\vec{\lambda}\cdot\vec{\phi}(\vec{x})}$

Claim:

$$\frac{\partial Z[\vec{\lambda}]}{\partial \vec{\lambda}} = \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}).,$$

where the notation $\sum_{\vec{x}}$ means the sum over states of a probability distribution (e.g., $\sum_{\vec{x}} \vec{x} p(\vec{x})$ is the expected value). It could also be written as an integral for the continuous case, but we will use the summation notation.

Proof:

$$\frac{\partial \log Z[\vec{\lambda}]}{\partial \vec{\lambda}} = \frac{1}{Z[\vec{\lambda}]} \frac{\partial Z[\vec{\lambda}]}{\partial \vec{\lambda}} = \frac{1}{Z[\vec{\lambda}]} \sum_{\vec{x}} \vec{\phi}(\vec{x}) e^{\vec{\lambda}\cdot\vec{\phi}(\vec{(}x)} = \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}).$$

Claim: For exponential distributions, ML corresponds to finding the value of $\vec{\lambda}$ s.t. the model statistics are equal to the data statistics. This consists in solving

$$\sum_{\vec{x}} \vec{\phi}(\vec{x}_i) p(\vec{x}_i|\vec{\lambda}) = \frac{1}{N} \sum_{\vec{x}} \vec{\phi}(\vec{x}_i)$$

E.g., for a Gaussian the model statistics are $\int d\vec{x}\vec{x}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\{-(1/2\sigma^2)(\vec{x}-\vec{\mu})^2\} = \vec{\mu}$. The data statistics are $1/N\sum_{i=1}^{N}\vec{x}_i$.

Proof: 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 is

$$F[\vec{\lambda}] = N\log Z[\vec{\lambda}] - \sum_{i=1}^{N}\vec{\lambda}\cdot\vec{\phi}(\vec{x}_i).$$

Differentiating with respect to $\vec{\lambda}$,

$$\frac{\partial F}{\partial\vec{\lambda}} = N\sum_{\vec{x}}\vec{\phi}(\vec{x}_i)p(\vec{x}_i|\vec{\lambda}) - \sum_{i=1}^{N}\vec{\phi}(\vec{x}_i)$$

Note: for some exponential distributions it is possible to compute the expected statistics of the model analytically to obtain a function $f(\lambda) = \sum_x \phi(x)p(x|\lambda)$. In this case, 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 $\sum_x \phi(x)p(x|\lambda)$. Instead we use an algorithm to minimize $F[\vec{\lambda}]$ with respect to $\vec{\lambda}$. Fortunately $F[\vec{\lambda}]$ is a convex function of $\vec{\lambda}$ and hence has only a single minimum.

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

It can be shown that $F[\vec{\lambda}]$ is a convex function which is bounded below, see figure (1). Convexity can be shown because the Hessian $\frac{\partial^2 F}{\partial\vec{\lambda}\partial\vec{\lambda}}$ is positive semi-definite (requires using the Cauchy-Schwartz inequality). This means that $F[\vec{\lambda}]$ has a unique minimum and hence there is a unique solution to MLE (for exponentials).

The convexity of $F[\vec{\lambda}]$ means that we can specify algorithms which estimate $\hat{\lambda}$ – for the cases where we cannot compute the model statistics analytically (see earlier). These algorithms require only that we can compute the expected statistics $\sum_{\vec{x}}\vec{\phi}(\vec{x})p(\vec{x}|\vec{\lambda})$ for

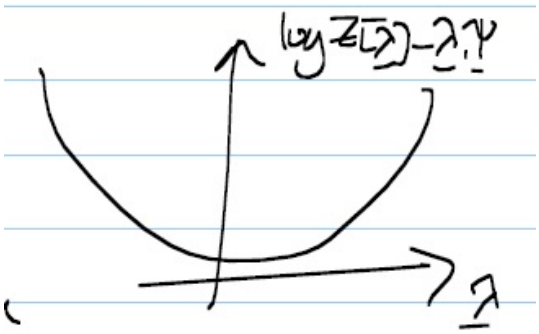

Figure 1: $F[\vec{\lambda}]/N = \log Z[\vec{\lambda}] - \vec{\lambda} \cdot \vec{\psi}$. Here $\vec{\psi} = (1/N) \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)$.

any value of $\vec{\lambda}$, which is a weaker requirement. (Also these methods can be extended to approximate techniques if this summation can only be approximated – beyond the scope of this course). These algorithms are guaranteed to converge (due to the convexity of $F[\vec{\lambda}]$).

Algorithm 1: Steepest Descent:

$$\vec{\lambda}^{t+1} = \vec{\lambda}^t - \Delta\{\sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}) - \frac{1}{N} \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)\}.$$

Here Δ is a "time step" constant. The continuous form of steepest descent is the differential equation $\frac{d\vec{\lambda}}{dt} = -\frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}}$. We approximate $\frac{d\vec{\lambda}}{dt}$ by $\frac{\vec{\lambda}^{t+1} - \vec{\lambda}^t}{\Delta}$ we compute $\frac{\partial F[\vec{\lambda}]}{\partial \vec{\lambda}} = \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}) - \frac{1}{N} \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)$. 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.

Algorithm 2: Generalized Iterative Scaling. This algorithm is similar to steepest descent, but does not need a time step parameter Δ.

$$\vec{\lambda}^{t+1} = \vec{\lambda}^t - \log \sum_{\vec{x}} \vec{\phi}(\vec{x}) p(\vec{x}|\vec{\lambda}^t) + \log \frac{1}{N} \sum_{i=1}^{N} \vec{\phi}(\vec{x}_i)$$

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.

2.4 Examples of learning Exponential Distributions

2.4.1 Gaussian distribution

The Gaussian distribution has a density function

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

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, $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}$ 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.

2.4.2 Letters example

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 $\vec{\phi}(x) = (\delta_{x,a}, \delta_{x,b}, \cdots, \delta_{x,z})$, where $\delta_{x,a} = 1$ if $x = a$, $\delta_{x,a} = 0$ otherwise. For instance, the probability of the letter c is $\vec{\phi}(c) = (0, 0, 1, 0, 0, \cdots, 0)$

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 #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 $\vec{\lambda} = (\lambda_a, \lambda_b, \cdots, \lambda_z)$. The exponential distribution representing the dataset is:

$$p(x|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}]} e^{\lambda_a \delta_{x,a} + \cdots + \lambda_z \delta_{x,z}}$$

We want the statistics of the data to be equal to the statistics of the model:

$$\sum_x p(x|\vec{\lambda}) \delta_{x,a} = \frac{1}{Z[\vec{\lambda}]} e^{\lambda_a},$$

where the right-hand side of the equation is the probability of the letter a, given the parameters of the distribution, $p(x = a|\vec{\lambda})$.

Also, $Z[\lambda] = e^{\lambda_a} + \cdots + e^{\lambda_z}$ because $\frac{1}{Z[\lambda]}(e^{\lambda_a}, e^{\lambda_b}, \cdots, e^{\lambda_z}) = \left(\frac{\#\text{a's}}{N}, \frac{\#\text{b's}}{N}, \cdots, \frac{\#\text{z's}}{N}\right)$. Hence, the solution is

$$\hat{\lambda}_a = \log \#\text{a's} - \log N, \ \hat{\lambda}_b = \log \#\text{b's} - \log N, \ \cdots, \ \hat{\lambda}_z = \log \#\text{z's} - \log N$$

$$Z[\hat{\lambda}] = \frac{\#\text{a's}}{N} + \frac{\#\text{b's}}{N} + \cdots + \frac{\#\text{z's}}{N} = 1.$$

3 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 makes sense if we have the wrong model – it gives the best approximation (we will clarify this).

3.1 Kullback-Leiber divergence

We now discuss how to justify ML as an approximation if the data is generated by a different distribution. 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
$D(f||p) \geqq 0 \quad \forall f, p$
$D(f||p) = 0, \quad$ if, and only if, $f(x) = p(\vec{x}|\vec{\lambda})$

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 $\vec{\lambda}$
$\sum_{\vec{x}} f(\vec{x}) \log p(\vec{x}|\vec{\lambda})$ depends on $\vec{\lambda}$

Hence, minimizing $D(f||p)$ with respect to $\vec{\lambda}$ corresponds to minimizing $-\sum_{\vec{x}} f(\vec{x}) \log p(\vec{x}|\vec{\lambda})$.

3.2 Geometric interpretation

Information geometry (Shun'ichi Amari, 1980) applies methods of differential geometry to probability distributions. $p(\vec{x}|\vec{\theta}) = \dfrac{e^{\vec{\lambda}\cdot\vec{\psi}(\vec{x})}}{Z[\vec{\lambda}]}$ defines a sub-manifold of distributions, the $\vec{\lambda}$'s being 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 (see Figure (2).

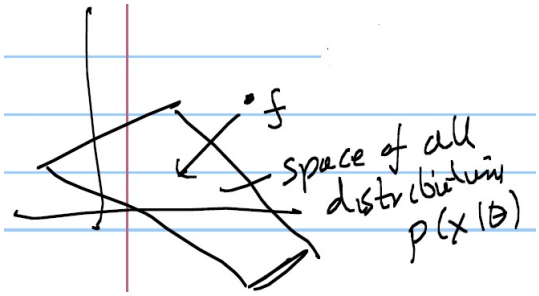

Figure 2: Space of all distribution $p(\vec{x}|\vec{\lambda})$. MLE corresponds to finding the best projection of the empirical distribution $f(\vec{x})$ onto the space of distributions of form $p(\vec{x}|\vec{\lambda}) = (1/Z[\vec{\lambda}]) \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x})\}$.

3.3 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 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.

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.
Model pursuit is beyond the scope of this course.

3.4 Letters example, alternative model

In Subsection 2.4.2 we had a dataset of single letters. In this example, 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-Liebler 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 because of the pairwise regularities. E.g, in English, qu is frequent, qz is impossible.

Note: if we have more letters, e.g, $\mathcal{X} = \{\text{brown, smith, loves, hates, ghost,}\ldots,\}$, then higher order statistics are best. But higher order statistics require more parameters – M-letters requires 26^M parameters – so we don't usually have enough data. Claude Shannon fit models like these to estimate the entropy of English.

4 Maximum Entropy

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 (solution is not unique), so we have to make some assumptions.

4.1 Entropy

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 $p(x) = \dfrac{1}{Z[\vec{\lambda}]} e^{\vec{\lambda}\cdot\phi(\vec{x})}$? And how to justify using ML to get $\vec{\lambda}$?

Entropy of a distribution $p(\vec{x})$:

$$H[p] = -\sum_x p(\vec{x}) \log p(\vec{x})$$

It is a measure of the amount of information obtained by observing a sample $\vec{x}$ from a distribution $p(\vec{x})$.

Shannon – Information Theory: Encode a signal $\vec{x}$ by a code of length $-\log p(\vec{x})$ – so that frequent signals ($p(\vec{x})$ big) have short codes and infrequent signals ($p(\vec{x})$ small) have long codes. Then the expected code length is $-\sum_{\vec{x}} p(\vec{x}) \log p(\vec{x})$. Alternatively, the entropy is the amount of information we expect to get from a signal $\vec{x}$ before we observe it – but we know that the signal has been sampled from a distribution $p(\vec{x})$.

Entropy is a concept discovered 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.

Example 1 : Suppose $\vec{x}$ can take N states: $\vec{\alpha}_1, \vec{\alpha}_2, ..., \vec{\alpha}_N$

Let: $p(\vec{x} = \vec{\alpha}_1) = 1 \;\; p(\vec{x} = \vec{\alpha}_j) = 0, \qquad j = 2, ..., N.$

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 $-0 \log 0 + (N-1)\{1log1\}$, and $0 \log 0 = 0$ 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. (Figure (3).

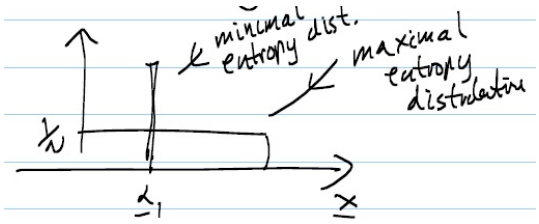

Figure 3: Example of maximum entropy and minimum entropy distributions.

4.2 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).

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

μ, λ: lagrange multipliers impose constraints on $p(\vec{x})$:
$\frac{\delta}{\delta p(\vec{x})}$ $-\log p(\vec{x}) - 1 + \mu + \vec{\lambda} \cdot \vec{\phi}(\vec{x}) = 0$

Solution, $p(\vec{x}|\vec{\lambda}) = \frac{\exp^{\vec{\lambda} \cdot \vec{\phi}(\vec{x})}}{Z[\vec{\lambda}]}$, where $\vec{\lambda}, Z[\vec{\lambda}]$ 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. $\sum_{\vec{x}} p(\vec{x}|\vec{\lambda})\phi(\vec{x}) = \vec{\psi}$

The maximum entropy principle recovers exponential distribution!

4.3 Entropy and Maximum Likelihood

Suppose we have data $\{\vec{x}_i : i = 1..N\}$, and fit a probability distribution $p(\vec{x}|\vec{\lambda})$ by ML to get the parameters $\hat{\vec{\lambda}}$. The probability of the data, using $p(\vec{x}|\hat{\vec{\lambda}})$ with the best estimate $(\hat{\vec{\lambda}})$ is

$$\prod_{i=1}^{N} P(\vec{x}_i|\hat{\vec{\lambda}}) = \exp\left\{\hat{\vec{\lambda}}_i \sum_{i=1}^{N} \phi(\vec{x}_i) - N \log Z[\vec{\lambda}]\right\}$$

The entropy of $p(\vec{x}|\hat{\vec{\lambda}})$ is

$$-\sum_{vecx} p(\vec{x}|\hat{\vec{\lambda}}) \log p(\vec{x}|\hat{\vec{\lambda}}) = \log \vec{Z}[\hat{\vec{\lambda}}] - \sum_{\vec{x}} \hat{\vec{\lambda}}\vec{\phi}(\vec{x}) p(\vec{x}|\hat{\vec{\lambda}}) = \log \vec{Z}[\hat{\vec{\lambda}}] - \frac{1}{N}\sum_{i=1}^{N} \hat{\vec{\lambda}}\vec{\phi}(\vec{x}_i)$$

Hence, the probability of data given $p(\vec{x}|\hat{\vec{\lambda}})$ is $\exp\{-N\mathcal{H}[p(\vec{x}|\hat{\vec{\lambda}})]\}$. So if the entropy of $p(\vec{x}|\hat{\vec{\lambda}})$ is small, then it does not describe the data well – it cannot predict it and there is a lot of uncertainty. Two related measures of the model are its entropy and the probability of the data given the model. This motivates Shannon's search for the entropy of English. It is an example of *model pursuit*.

Shannon starts with unary statistics – frequency of letters – to fit the data of English texts, and estimates the entropy. Then he uses more complex statistics – pairwise frequencies – and entropy decreases, which means a better fit. For example, what is the next letter of "ryth_"? The entropy here is very low.