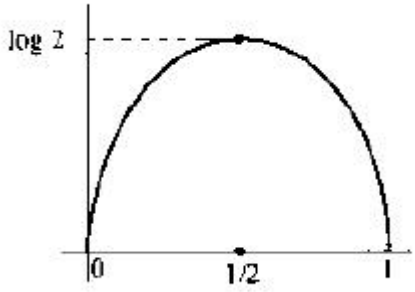
**MAX ENTROPY:**




Naïve Bayes only care about maximum likelihood, while MaxEnt care about not only likelihood but maximum entropy?
Commonly used in nlp.
熵越大混乱程度越高，不确定性越高。但是越稳定，因为他平均啊，事情总是朝着平均方向发展的。 The higher entropy, the more stable, the higher impurity.
The features for a MaxEnt model can be correlated. The model will do a good job of distributing the weight between correlated features. (This is not to say, though, that you should be indifferent to correlated features. They can make the model hard to interpret and reduce its portability.)

The maximum entropy method answers both these questions. Intuitively, the principle is simple: model all that is known and assumes nothing about that which is unknown. In other words, given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible. This is precisely the approach we took in selecting our model P at each step in the above example.

推导：$H(X|Y=y)= \sum p(x|y)\log\{1/p(x,y)\}$

$H(X|Y)=\sum p(y)H(X|Y=y)= \sum p(y)*\sum p(x|y)\log\{1/p(x/y)\}$

Average others with some constraints.

$$H_p(Y|X) = -\sum_{x\in X} p(x) \sum_{y\in Y} p(y|x) \log p(y|x)$$

Lagrange for seeking extreme：
Lagrange multiplier: In mathematical optimization, the method of Lagrange multipliers (named after Joseph Louis Lagrange) is a strategy for finding the local maxima and minima of a function subject to equality constraints.

$$\arg\max_{p \in P} H_p(Y|X)$$

$$p(y|x) = \frac{\exp(\sum_k \lambda_k f_k(y,x))}{\sum_{y' \in Y} \exp(\sum_k \lambda_k f_k(y',x))}$$

MLE, Naïve Bayes maximizes the joint probability, while MaxEnt maximizes the conditional probability? (generative vs discriminative?)

$$L_D(\Lambda) = \log \prod_{i=1}^{|D|} p_\Lambda(y^{(i)}|x^{(i)})$$

$$= \sum_{i=1}^{|D|} \log p_\Lambda(y^{(i)}|x^{(i)})$$

$$\begin{aligned} L_D(\Lambda) &= \log \prod_{i=1}^{|D|} p_\Lambda(y^{(i)}|x^{(i)}) \\ &= \sum_{i=1}^{|D|} \log p_\Lambda(y^{(i)}|x^{(i)}) \\ &= \sum_{i=1}^{|D|} \log \frac{\exp \sum_{k=1}^{n} \lambda_k f_k(y^{(i)}, x^{(i)})}{\sum_{y'} \exp \sum_{i=k}^{n} \lambda_k f_k(y', x^{(i)})} \\ &= \sum_{i=1}^{|D|} \sum_{k=1}^{n} \lambda_k f_k(y^{(i)}, x^{(i)}) - \sum_{i=1}^{|D|} \log \sum_{y'} \exp \sum_{k=1}^{n} \lambda_k f_k(y', x^{(i)}) \end{aligned}$$

This function, the conditional log-likelihood function turns out to be **convex** with a single global maximum. We can maximize this function using techniques from the field of **convex Optimization**.

Take a derivative

$$\begin{aligned} \frac{\partial L_D(\Lambda)}{\partial \lambda_k} &= \sum_{i=1}^{|D|} f_k(y^{(i)}, x^{(i)}) - \sum_{i=1}^{|D|} \sum_{y' \in Y} \frac{f_k(y', x^{(i)}) \exp \sum_k \lambda_k f_k(y', x^{(i)})}{\sum_{\hat{y} \in Y} \exp \sum_k \lambda_k f_k(\hat{y}, x^{(i)})} \\ &= \sum_{i=1}^{|D|} f_k(y^{(i)}, x^{(i)}) - \sum_{i=1}^{|D|} \sum_{y' \in Y} p_\Lambda(y'|x) f_k(y', x^{(i)}) \end{aligned}$$

- Function optimum achieved when each partial (each component of the gradient) is zero, that is, when:

$$\sum_{i=1}^{|D|} f_k(y^{(i)}, x^{(i)}) = \sum_{i=1}^{|D|} \sum_{y' \in Y} p_\Lambda(y'|x) f_k(y', x^{(i)})$$

- These addends are the empirical and model **expectations** for $k$th feature, $E[f_k] = E_\Lambda[f_k]$

**Outline for training a MaxEnt model**

1. Compute log-likelihood of dataset and its gradient with current parameters
2. Update parameters based on log-likelihood and gradient (via convex optimization)
3. If converged, stop, otherwise go to (1)

1. Transform each blog to a binary vector. Each value of the vector represents the occurrence of words in bag-of-words model. The size of each vector is 1*N, which N is the number of words in bag-of-words model. The vector would be sparse.
2. Initiate lambda(feature weight). The size of lambda is 1*2N (f(M, word), f(W, word)).
3. Implement functions for calculating log-likelihood and gradients by using matrix calculation. Here I can write a function used for calculating P(y|X) and record these posterior values. Posterior values are both useful for likelihood calculation and gradient calculation.
4. Pass log-likelihood function, gradients function and initial lambda into 'scipy.optimize.fmin_l_bfgs_b' function, which is a black box function for getting 'minimum'. Now we get the optimum value for lambda.
5. In test part, calculate likelihood of M and W. Then get the argmax_y(P(y|X)).

Optimization is like gradient descent in regression. Lambda is x in regression. Each time in iteration we change the value of lambda and let it approach optimum value. We can also use stochastic gradient.

我觉得这里和 Naïve Bayes 用 maximum likelihood 有点相同的意思啊。这个最终结论本身我们也能看出来，但是这里从数学上证明了这点。就是期望值和真实值要等才可以最大化。

MAP 和 ML 的对比，为什么 MAP 是避免 overfitting 的。因为 MAP 是 find the parameters that best agree with the data and the prior, while ML only find the parameters agree with the data?那这个 prior 是我假设的？

## Maximum a posteriori (MAP) Estimation

- A type of model *regularization*
- This is a penalty to prevent model complexity, overfitting
- From a Bayesian view, regularization is a *prior* over the possible parameter values
- Maximum a posteriori estimation involves introducing a prior over the model parameters and finding the parameters that best agree with the data *and* the prior
- This prior is a zero-mean Gaussian distribution with variance $\sigma$
- Parameters are *penalized* inversely proportional to the probability density of this distribution.

$$L'_D(\Lambda) = \sum_{i=1}^{|D|} \sum_{k=1}^{n} \lambda_k f_k(y^{(i)}, x^{(i)}) - \sum_{i=1}^{|D|} \log \sum_{y'} \exp \sum_{k=1}^{n} \lambda_k f_k(y', x^{(i)}) - \sum_k \frac{\lambda_k^2}{\sigma^2}$$

Feature functions:

A positive weight votes that this configuration is likely correct

A negative weight votes that this configuration is likely incorrect

Feature functions is an observation and the corresponding y. f(y, observation)

One good pdf:
https://web.stanford.edu/class/cs124/lec/Maximum_Entropy_Classifiers.pdf
Naïve Bayes vs Maxent
http://sentiment.christopherpotts.net/classifiers.html