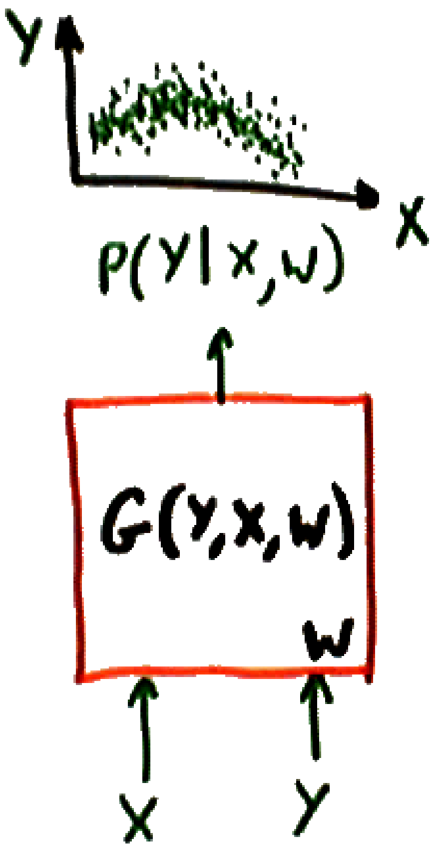
MACHINE LEARNING AND PATTERN RECOGNITION

Spring 2004, Lecture 3: Probabilistic Learning: Maximum Likelihood Estimation, MAP

Yann LeCun
The Courant Institute,
New York University
http://yann.lecun.com

Probabilistic Framework for Learning

Given a set of observations $\mathcal{S} = [(X^1, Y^1), (X^2, Y^2)...(X^P, Y^P)]$ (training set), we want to produce a model that predict Y from X. More precisely, we want to estimate a function that computes the conditional distribution $P(Y|X)$ for any given X, in a way that is consistent with the observations $\mathcal{S}$. We write this function $P(Y|X, \mathcal{S})$

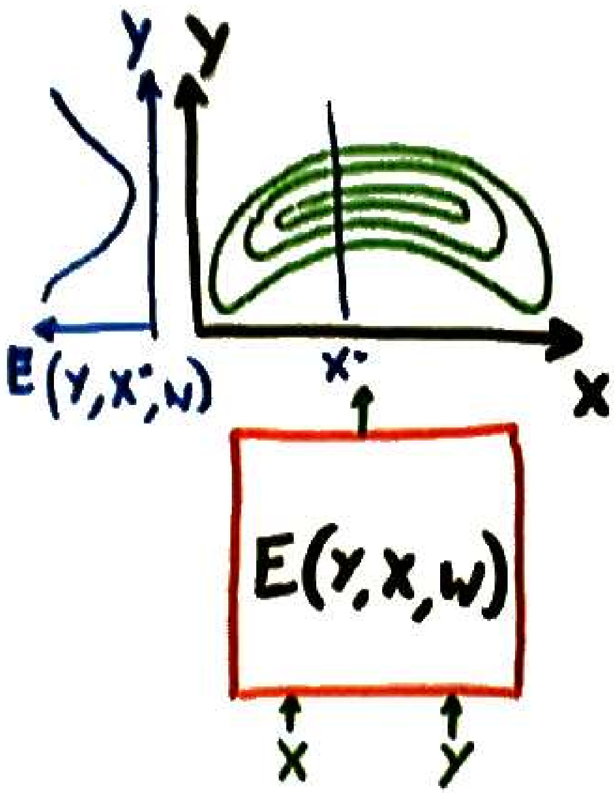


- **STEP 1**, architecture and parameterization: We assume that $P(Y|X, \mathcal{S})$ can be decomposed into two parts:

$$P(Y|X, \mathcal{S}) = \int P(Y|X, W)P(W|\mathcal{S})dW$$

- Our estimate of $P(Y|X, W)$ will taken among a *family of functions* $\{G(Y, X, W), \forall W\}$, parameterized by a parameter vector W.
- This family of functions, which we will call the *architecture* of the learning machine, must be chosen carefully for each particular problem. Examples of architectures include logistic regressors, neural networks, Hidden Markov Models, and literally hundreds of others.

Energy Function



- **STEP 2**, energy function: for convenience we will often define $G(Y,X,W)$ as the normalized exponential of an *energy function* $E(X,Y,W)$:

$$P(Y|X,W) \approx G(Y,X,W) = \frac{\exp(-\beta E(X,Y,W))}{Z_E(X,W,\beta)}$$

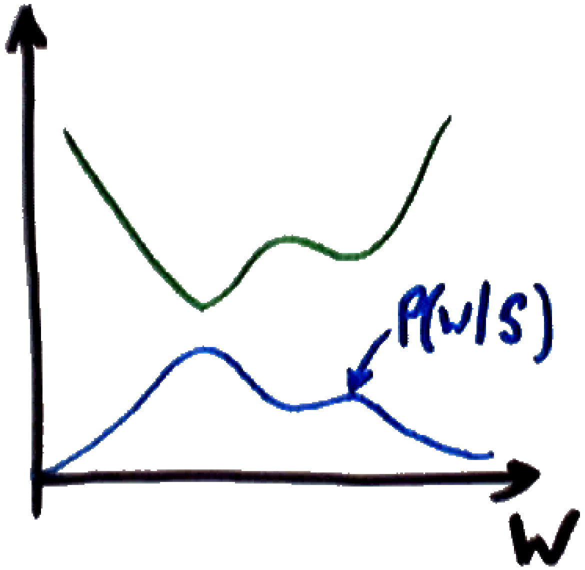
where β is an arbitrary positive constant, and $Z_E(X,W,\beta)$ is a normalization term

$$Z_E(X,W,\beta) = \int \exp(-\beta E(X,Y,W))dY$$

(often called the *partition function*) which ensures that our estimate of $P(Y|X,W)$ is normalized.

Probabilistic Framework for Learning

$$P(Y|X,\mathcal{S}) = \int P(Y|X,W)P(W|\mathcal{S})dW$$



- **STEP 3**, learning: $P(W|\mathcal{S})$ is the result of a *learning procedure* that assigns a probability (or an energy) to each possible value of W as a function of the training set.
- Intuitively, the learning procedure will assign high probabilities to values of W that assign high combined probability (low combined energy) to the observed data.

Assigning Probs to Parameter Values

STEP 3.1, likelihood of observations: Let us define

$$\mathcal{X} = (X^1, X^2, \ldots X^p), \text{ and } \mathcal{Y} = (Y^1, Y^2, \ldots Y^p)$$

Using Bayes inversion while keeping all the terms conditioned on $\mathcal{X}$, we can write:

$$P(W|\mathcal{S}) = P(W|\mathcal{Y}, \mathcal{X}) = \frac{P(\mathcal{Y}|\mathcal{X}, W)P(W|\mathcal{X})}{P(\mathcal{Y}|\mathcal{X})}$$

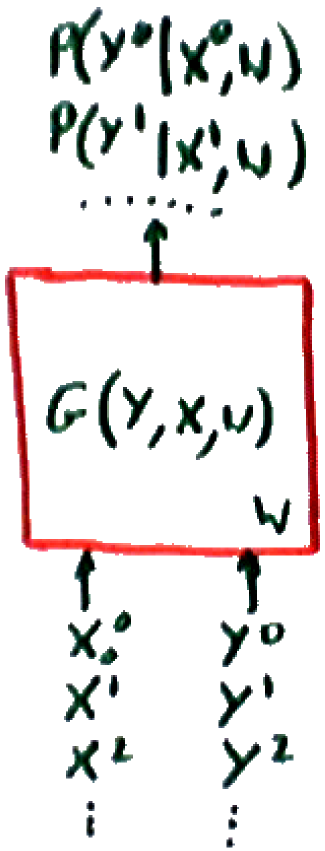
where the denominator is a normalization term:

$$P(\mathcal{Y}|\mathcal{X}) = \int P(\mathcal{Y}|\mathcal{X}, W)P(W|\mathcal{X})dW$$

that ensures that $\int P(W|\mathcal{S})dW = 1$.

Likelihood of the Observations

STEP 3.2, sample independence: To compute $P(\mathcal{Y}|\mathcal{X}, W)$, we use the assumption that it can be written as a product of terms that each depend on a single training sample. In other words, we see the drawing of the samples (X^i, Y^i)'s as independent events:



$$P(\mathcal{Y}|\mathcal{X}, W) = \prod_i P(Y^i|X^i, W) = \prod_i G(Y^i, X^i, W)$$

Using the definition for G:

$$P(\mathcal{Y}|\mathcal{X}, W) = \exp(-\beta \sum_i [E(Y^i, X^i, W) + \frac{1}{\beta} \log Z_E(X^i, W, \beta)])$$

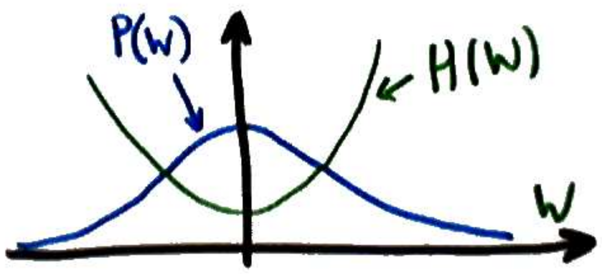
with $Z_E(X, W, \beta) = \int \exp(-\beta E(X, Y, W)) dY$.

Priors, Regularizers, Penalty terms

$$P(W|\mathcal{S}) = P(W|\mathcal{Y},\mathcal{X}) = \frac{P(\mathcal{Y}|\mathcal{X},W)P(W|\mathcal{X})}{P(\mathcal{Y}|\mathcal{X})}$$

STEP 3.3, choosing a regularizer: The term $P(W|\mathcal{X})$ is an arbitrary prior distribution over the values of W that we can choose freely. In the following, we will drop the dependency on $\mathcal{X}$. We will often represent this prior as the normalized exponential of a *penalty term* or *regularizer* $H(W)$:

$$P(W) = \frac{1}{Z_H}\exp(-\beta H(W))$$



- Parameters that produce low values of the regularizer will be favored over parameters that produce large values. Therefore our choice or regularizer will determine what we consider "good" models (e.g. simple, smooth, well behaved) for which the regularizer is small, and "bad" models for which the regularizer is large.

Probability of a Parameter

- **STEP 3.4**, Putting it all together: The probability of a particular parameter value W given the observations $\mathcal{S}$ is:

$$P(W|\mathcal{S}) = \frac{\exp(-\beta\{\sum_i [E(Y^i, X^i, W) + \frac{1}{\beta} \log Z_E(X^i, W, \beta)] + H(W)\})}{Z_W(\mathcal{S}, \beta)}$$

- $E(Y, X, W)$ is our energy function. We can give it any form we like. Considerable effort should be spent designing appropriate forms of $E(Y, X, W)$ (good architectures) for particular problems. Many specific architectures will be discussed in the rest of the course.
- $H(W)$ is the regularizer that contains our preferences for “good” models over “bad” ones. Our choice of $H(W)$ is somewhat arbitrary, but some work better than others for particular applications.

Probability of a Parameter (continued)

$$P(W|\mathcal{S}) = \frac{\exp(-\beta\{\sum_i [E(Y^i, X^i, W) + \frac{1}{\beta}\log Z_E(X^i, W, \beta)] + H(W)\})}{Z_W(\mathcal{S}, \beta)}$$

- $Z_W(\mathcal{S}, \beta)$ is the normalization term that ensures that the integral of $P(W|\mathcal{S})$ over W is 1: $Z_W(\mathcal{S}, \beta)$ is the integral over W of the numerator.
- $Z_E(X^i, W, \beta)$ are the normalization terms (one for each sample) that ensure that the integral $P(Y|X^i, W)$ over Y is 1:
  $Z_E(X^i, W, \beta) = \int exp(-\beta E(Y, X^i, W))dY$.
- β is a positive constant that we are free to choose as we like or that we can estimate. It reflects the "reliability" of the data. Low values should be used to get probability estimates with noisy data. Large values should be used to get good discrimination. We can estimate β through learning too (we can fold it into E, as a component of W).

Intractability of Bayesian Learning

- Recall that the formula for our predictor was:

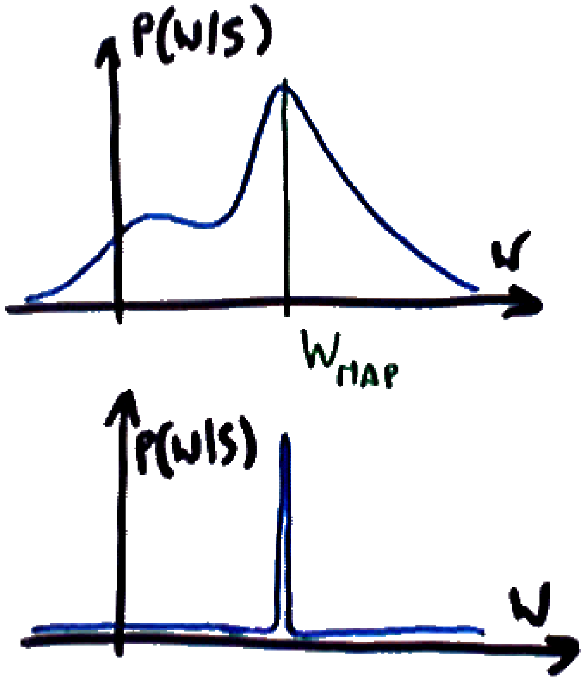
$$P(Y|X,\mathcal{S}) = \int G(Y,X,W)P(W|\mathcal{S})dW$$

- To compute the distribution of Y for a particular input X, we are supposed to integrate the product of two complicated functions over all possible values of W.
- This is totally intractable in general.
- There are special classes of functions for G for which the integral is tractable, but that class is fairly restricted.
- So, we need to take a shortcut.
- There is a number of popular shortcuts....

Making Bayesian Learning Tractable

- First shortcut: *Maximum A Posteriori Estimation*. simply replace the distribution $P(W|\mathcal{S})$ by a Dirac delta function centered on its mode (maximum).
- Second shortcut: *Maximum Likelihood Estimation*. Same as above, but drop the regularizer.
- Third Shortcut: *Restricted Class of function*. Simply restrict yourself to special forms of $G(Y, X, W)$ for which the integral can be computed analytically (e.g. Gaussians). CAUTION: This is a perfect example of looking for your lost keys under the street light.
- Fourth shortcut: *Sampling*. Draw a a bunch of samples of W from the distribution $P(W|\mathcal{S})$, and replace the integral by a sum over those samples.
- Fifth Shortcut: *Local Approximations*. compute a Taylor series of $P(W|\mathcal{S})$ around its maximum and integrate with the resulting (multivariate) polynomial.

Maximum A Posteriori Estimation



- **Maximum A Posteriori Estimation**: assume that the mode (maximum) of $P(W|\mathcal{S})$ is so much larger than all other values that we can view $P(W|\mathcal{S})$ as a Dirac delta function centered around its maximum

$$P_{\text{MAP}}(W|\mathcal{S}) \approx \delta(W - W_{\text{MAP}})$$

$$W_{\text{MAP}} = \text{argmax}_W P(W|\mathcal{S})$$

with this approximation, we get simply:

$$P(Y|X, \mathcal{S}) = P(Y|X, W_{\text{MAP}})$$

If we take the limit $\beta \to \infty$, $P(W|\mathcal{S})$ does converge to a delta function around its maximum. So the MAP approximation is simply the large β limit.

Question: How do we compute W_{MAP}?

Computing W_{MAP}

Now, here is the cool thing $W_{\mathrm{MAP}} = \mathrm{argmax}_W P(W|\mathcal{S}) =$

$$
\begin{aligned}
= & \quad \mathrm{argmax}_W \tfrac{1}{Z_W(\mathcal{S},\beta)} \exp(-\beta\{\textstyle\sum_i [E(Y^i, X^i, W) + \frac{1}{\beta}\log Z_E(X^i, W, \beta)] + H(W)\}) \\
= & \quad \mathrm{argmax}_W \exp(-\beta\{\textstyle\sum_i [E(Y^i, X^i, W) + \frac{1}{\beta}\log Z_E(X^i, W, \beta)] + H(W)\}) \\
= & \quad \mathrm{argmin}_W \textstyle\sum_i [E(Y^i, X^i, W) + \frac{1}{\beta}\log Z_E(X^i, W, \beta)] + H(W) \\
= & \quad \mathrm{argmin}_W \textstyle\sum_i [E(Y^i, X^i, W) + \frac{1}{\beta}\log \int \exp(-\beta E(Y, X^i, W))dY] + H(W)
\end{aligned}
$$

We can drop the normalizer $\frac{1}{Z_W(\mathcal{S},\beta)}$ because it does not depend on W. We can take the log because log is monotonic.

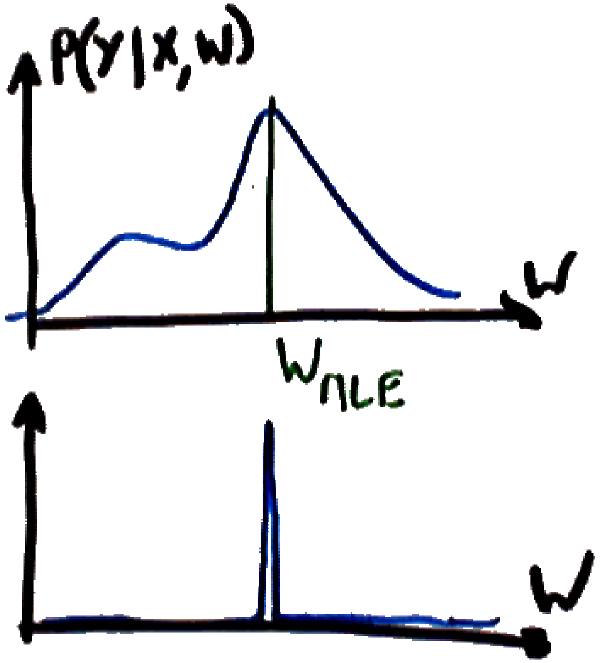
To find the MAP parameter estimate, we simply need to find the value of W that minimizes:

$$L_{\mathrm{MAP}}(W) = \sum_i [E(Y^i, X^i, W) + \frac{1}{\beta}\log \int \exp(-\beta E(Y, X^i, W))dY] + H(W)$$

Most learning algorithms are of that form

Maximum Likelihood Estimation

Maximum Likelihood Estimation: same as MAP, but ignore $H(W)$ altogether. This is equivalent to finding the W that maximizes $P(\mathcal{Y}|\mathcal{X}, W)$ (the likelihood of the data) instead of $P(\mathcal{Y}|\mathcal{X}, W)P(W|\mathcal{X})$ (the un-normalized posterior of the parameter).



- We assume that the mode (maximum) of $P(W|\mathcal{S})$ is so much larger than all other values that we can view $P(W|\mathcal{S})$ as a Dirac delta function centered around its maximum, and we assume that the prior $P(W)$ has no influence on the result:

$$P(W|\mathcal{S}) \approx \delta(W - W_{\text{MLE}}) \quad W_{\text{MLE}} = \text{argmax}_W P(\mathcal{Y}|\mathcal{X}, W)$$

- with this approximation, we get simply:

$$P(Y|X, \mathcal{S}) = P(Y|X, W_{\text{MLE}})$$

- How do we compute W_{MLE}?

Computing $W_{\rm MLE}$

Same result as with $W_{\rm MAP}$, except that $H(W)$ disappears: $W_{\rm MLE} =$

$$
\begin{aligned}
&= \text{argmax}_W P(W|\mathcal{S}) && (1)\\
&= \text{argmax}_W \frac{1}{Z_W(\mathcal{S},\beta)} \exp(-\beta\{\textstyle\sum_i [E(Y^i,X^i,W) + \frac{1}{\beta}\log Z_E(X^i,W,\beta)]\}) && (2)\\
&= \text{argmax}_W \exp(-\beta\{\textstyle\sum_i [E(Y^i,X^i,W) + \frac{1}{\beta}\log Z_E(X^i,W,\beta)]\}) && (3)\\
&= \text{argmin}_W \textstyle\sum_i [E(Y^i,X^i,W) + \frac{1}{\beta}\log Z_E(X^i,W,\beta)] && (4)\\
&= \text{argmin}_W \textstyle\sum_i [E(Y^i,X^i,W) + \frac{1}{\beta}\log \int \exp(-\beta E(Y,X^i,W))dY] && (5)\\
& && (6)
\end{aligned}
$$

to find the MLE parameter estimate, we simply need to find the value of W that minimizes:

$$L_{\rm MLE}(W) = \sum_i [E(Y^i,X^i,W) + \frac{1}{\beta}\log \int \exp(-\beta E(Y,X^i,W))dY]$$

A Little Digression

$$L_{\mathrm{MAP}}(W) = \sum_i [E(Y^i, X^i, W) + \frac{1}{\beta} \log \int \exp(-\beta E(Y, X^i, W))dY] + H(W)$$

- All of the terms in this equation have analogs and interpretations in *Statistical Physics* and *Thermodynamics*.
- $\sum_i [E(Y^i, X^i, W)$ is analogous to the *Average Energy* of a thermodynamical system where each sample is analogous to a particle in an ideal gas.
- $\sum_i \frac{1}{\beta} \log \int \exp(-\beta E(Y, X^i, W))dY$ is analogous to the *Helmoltz Free Energy* of a thermodynamical system.
- $1/\beta$ is analogous to the *Temperature* of the system.
- L_{MAP} is analogous to the product of the *Entropy* by the Temperature.
- The above equation is a form of the well-known thermodynamic equation $< \mathrm{Temperature} > \times < \mathrm{Entropy} >=< \mathrm{AverageEnergy} > - < \mathrm{FreeEnergy} >$.
- MAP/MLE estimation is like entropy minimization.

A Few Remarks on $\log \int \exp()$

The operator:

$$\text{logmean}_i^\beta(V_i) = -\frac{1}{\beta} \log \frac{1}{p} \sum_{i=1}^{p} exp(-\beta V_i)$$

has many interesting properties. Algebraically, it is to addition what addition is to multiplication:

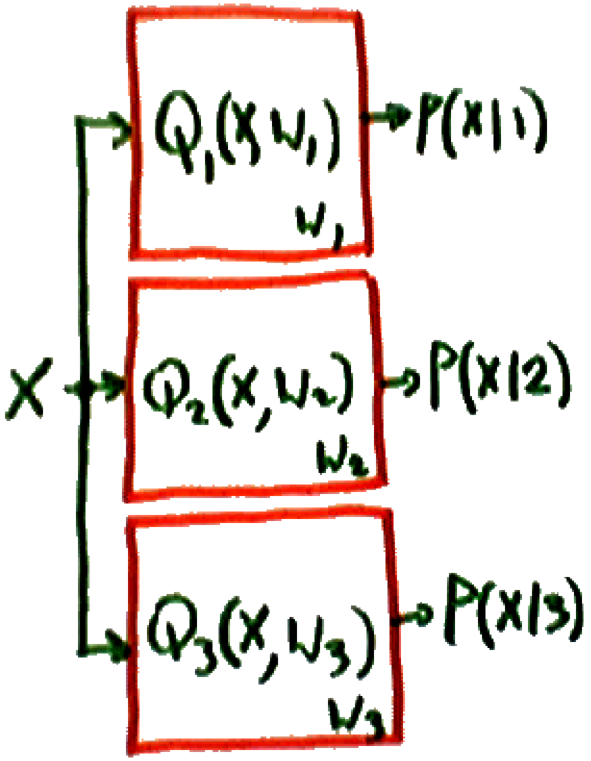
$$\begin{aligned}
\text{logmean}_i^\beta(V_i) &= V^* + \text{logmean}_i^\beta(V_i - V^*) \\
\lim_{\beta \to 0} \text{logmean}_i^\beta(V_i) &= \frac{1}{p} \sum_{i=1}^{p} V_i \\
\lim_{\beta \to \infty} \text{logmean}_i^\beta(V_i) &= V_{\min} \\
\lim_{\beta \to -\infty} \text{logmean}_i^\beta(V_i) &= V_{\max}
\end{aligned}$$

Generative and Discriminative Models

Early on, we assumed that $P(Y|X)$ was directly modeled by a parameterized family of functions $G(Y, X, W), \forall W$. However, we can Bayes-invert $P(Y|X)$ to get:

$$P(Y|X) = P(X|Y)P(Y)/P(X) = P(X|Y)P(Y)/\int_P (X|Y)P(Y)dY$$

and parameterize $P(X|Y)$ for each Y by a family of functions $Q_Y(X, W_y)\ \forall W_y$



- This seems like a very silly thing to do (and indeed it is in many cases): why should we submit ourselves to estimating such a horrendous object as $P(X|Y)$?
- $P(X|Y$ is the probability density (in input space) for one particular value of Y (e.g. a category). It's horrendous, because it must be normalized over the set of all possible X (which is **big**).
- Models that estimate $P(X|Y)$ are called **generative** because they can be used to *generate* input vectors X by sampling from $P(X|Y)$.

Generative/Discriminative Pros and Cons

- :-) in some cases, it is simpler to independently estimate a separate function $P(X|Y, W)$ for each class Y. This allows us to add classes simply by estimating $P(X|Y, W)$ for the new class, without revisiting the models of all the other classes.
- :-) Sometimes, it is easier to come up with a good architecture for a generative model by simply implementing a process that would *synthesize* the objects we are trying to recognize/classify (a process known as *analysis by synthesis*.
- :-) Training of generative models is generally faster.
- :-(Generative models solve a considerably more complex and more ill-conditioned problem than necessary.
- :-(Generative models spend considerable resource getting the whole distribution right when getting the class boundary right may be sufficient.
- :-(Because the model must be normalized over X, we are restricted to using simple "normalizable" density models. We can't use non-normalizable models like linear classifiers or neural nets.
- :-(Generative models almost always require much more memory and CPU time than discriminative one.

Unsupervised Learning

- The probabilistic form of unsupervised learning is *density estimation*: finding a function $G(Y, W)$ that best approximates the distribution $P(Y)$.
- The entire derivation of MAP and MLE estimation can be transposed to the case of unsupervised learning by simply omitting X from all the equations.
- The unsupervised objective function to be minimnized for unsupervised MAP is:

$$L_{\text{MAP}}(W) = \sum_i [E(Y^i, W) + \frac{1}{\beta} \log \int \exp(-\beta E(Y, W)) dY] + H(W)$$

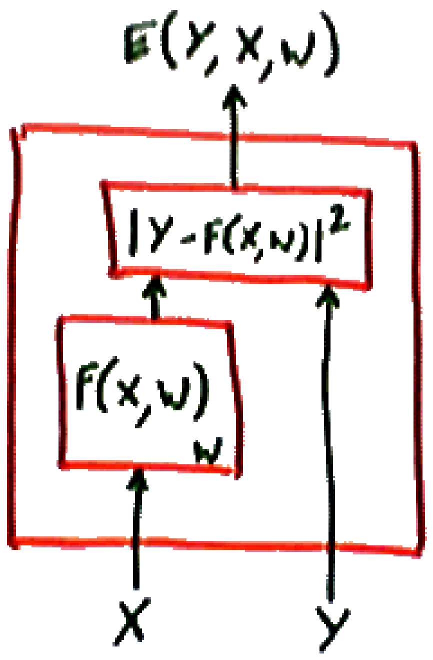
- The unsupervised objective function to be minimnized for unsupervised MLE is:

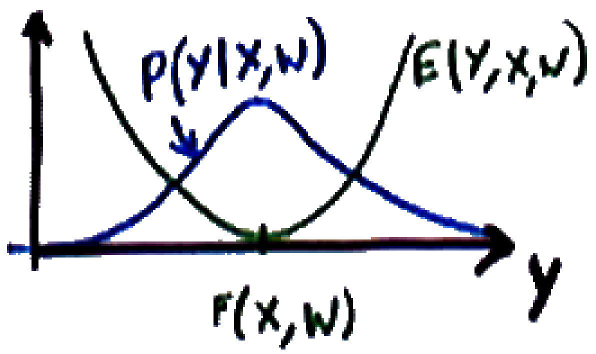
$$L_{\text{MLE}}(W) = \sum_i [E(Y^i, W) + \frac{1}{\beta} \log \int \exp(-\beta E(Y, W)) dY]$$

Energy-Based Models

- As we have seen, most reasonable probabilistic learning algorithms can be seen as simply minimizing an appropriately defined *Energy Function*.
- It is tempting, and often justified, to drop the probabilistic framework altogether, and simply manipulate energy functions...
- ...but sometimes, we need to worry about the "Free Energy" terms which ensure that everything is well normalized.
- sometimes we can drop the free energy term or transform it beyond recognition with no adverse effect (in fact with beneficial effects).
- By doing so, we may loose the probabilistic interpretation but we may improve the classification performance.

MAP/MLE Example: Least Square = Gaussian





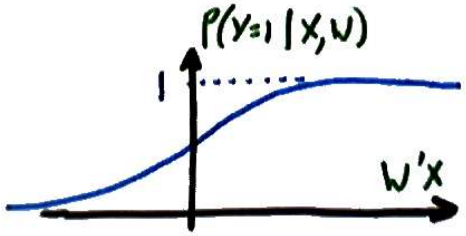
- MAP estimation of Gaussian models gives Least-Square Regression:

$$
\begin{aligned}
P(Y|X,W) &= \frac{1}{Z}\exp(-\frac{1}{2v}(Y - F(X,W))^2) \\
L_{\mathrm{MAP}}(W) &= \sum_i (Y^i - F(X^i,W))^2 + H(W)
\end{aligned}
$$

  up to a constant which does not affect the minimum.

- note: $F(X,W)$ can be non-linear: which leads to a non-linear least square optimization problem.
- if $H(W) = \lambda||W||^2$ this is *Ridge Regression*.

MAP/MLE Example: Logistic Regression



MAP estimation of binomial distribution gives Logistic Regression. Y is binary (0 or 1) with a linear parametrization of the likelihood ratio between the two classes:

$$\log \frac{P(Y=1|X,W)}{P(Y=0|X,W)} = W'X$$

$$\begin{aligned}
P(Y=1|X,W) &= \frac{1}{1+\exp(-W'X)} \\
G(Y,X,W) &= P(y=1|X,W)^Y (1-P(y=1|X,W))^{(1-Y)} \\
E(Y,X,W) &= Y \log \frac{1}{1+\exp(-W'X)} + (1-Y) \log \frac{1}{1+\exp(W'X)} \\
L_{\mathrm{MAP}}(W) &= \sum_i Y^i \log \frac{1}{1+\exp(-W'X^i)} + (1-Y^i) \log \frac{1}{1+\exp(W'X^i)} + H(W)
\end{aligned}$$

The Subjectivity of Learning

$$L_{\text{MAP}}(W) = \sum_i [E(Y^i, X^i, W) + \frac{1}{\beta} \log \int \exp(-\beta E(Y, X^i, W)) dY] + H(W)$$

- Regularizers, Penalty functions, Prior probabilities, Measures of Randomness, Capacity of a family of functions: these are different names for the same thing.
- It is hopeless to try to find a "universally good" functional form for E or for H. Different architctures and regularizers are good for different problems. Which one you pick is up to you. No theory will tell you how to best design your E and H for a particular problem.
- No Free Lunch Theorem: if your family of function and your prior is good for particular problems, it must be bad for others. No family/prior is good for all problems.
- The best theoretical framework to explain all this is the Vapnik-Chervonenkis (VC) theory.