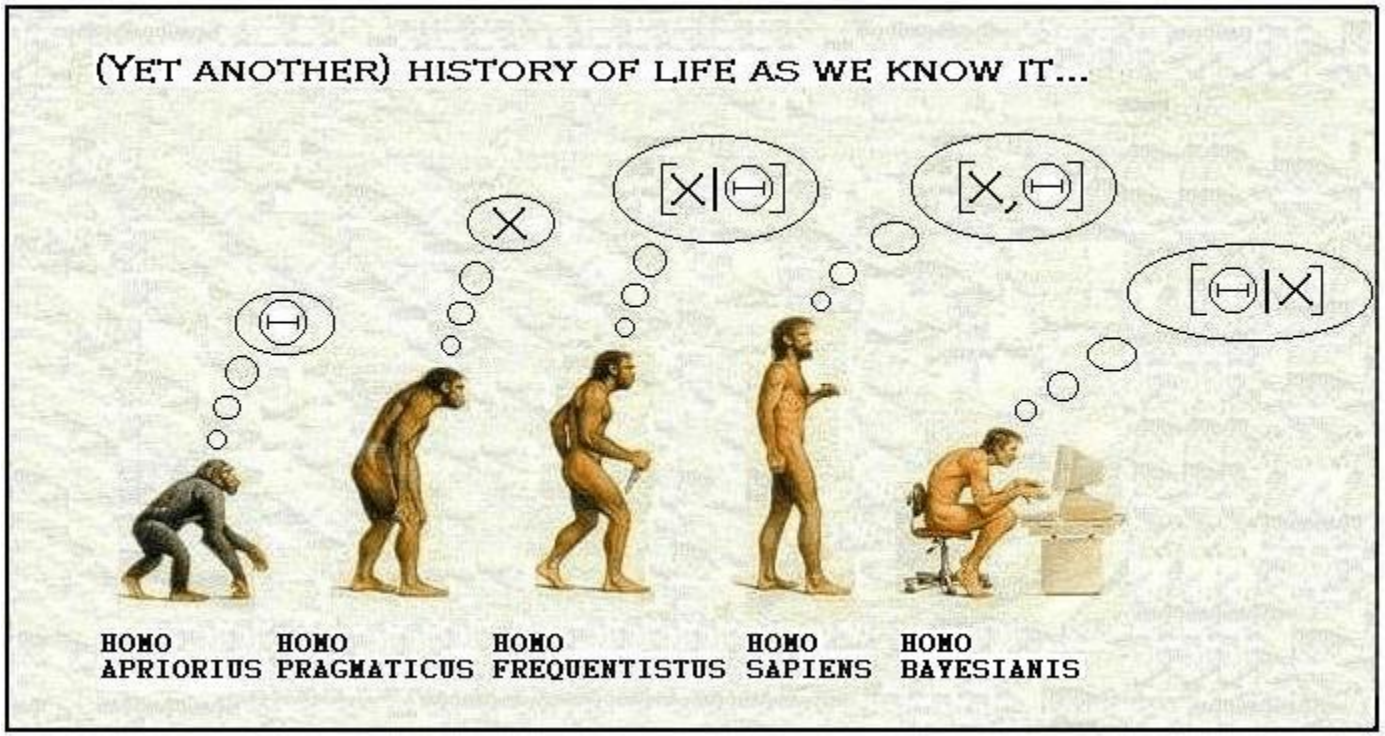
Bayesian Inference and an Intro to Monte Carlo Methods



Slides from Geoffrey Hinton and Iain Murray

CSC321: Intro to Machine Learning and Neural Networks, Winter 2016

Michael Guerzhoy

Reminder: Bayesian Inference

- Model: $\mathrm{y}_i = h_\theta(x_i)$
- Data: $\{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$
- Want: $P(\theta|data)$
  - I.e., we want the probability/degree of belief for every value of the parameter θ
    - E.g., the probability for a certain set of weights+biases of a neural network
    - For linear regression, the probability that $(a = a_0, b = b_0)$ for $h_{(a,b)}(x) = ax + b$, given the data that we observe

Use Bayes' Rule

- $P(\theta|data) = \frac{P(data,\theta)}{P(data)} = \frac{P(data|\theta)P(\theta)}{\sum_{\theta'} P(data|\theta')P(\theta')}$
- $-\log P(\theta|data) = -\log\big(P(data|\theta)\big) - \log\big(P(\theta)\big) + const$
  - $-\log P(data|\theta)$: can be thought of as the cost function without the penalty. The lower $-\log P(data|\theta)$, the higher $P(data|\theta)$, so the better the data fit the model with the parameter θ
  - $-\log P(\theta)$ can be thought of as the weight penalty
  - If we minize $-\log P(\theta|data)$, we can find the "best" parameters for the data

Full Bayesian Inference

- Instead of maximizing $P(\theta|data)$, simply compute it for every possible value of θ
- In a sense, this will give us the best possible information about what the "true" θ might be
  - The "true" θ is the θ that is used to generate the training set. E.g.:
    - $y = \theta_0 + \theta_1 x + 0.1N(0,1)$ for some specific values of θ_0 and θ_1 that we don't know
- What happens to $\mathrm{P}(\theta)$ as the training set gets larger?

Neural Network Example

- $P\left(y^{(i)}, x^{(i)} \middle| W\right) = Gaussian_{\sigma_y^2}\left(net_W\left(x^{(i)}\right) - y^{(i)}\right)$
  - I.e., $y^{(i)} = net_W\left(x^{(i)}\right) + \mathrm{N}(0, \sigma^2)$
    - I.e., the observed output of the network is what the network computes using the weights *W*, plus some Gaussian noise
- $P(W) = \Pi_j Gaussian_{\sigma_W^2}(W_j)$(or = const)
  - I.e., we believe W's that are closer to the 0 more (or we believe all weights equally)

Neural Network Example: Inference

- Can now compute $P(W|data)$ for any set of weights W
- Can compute $net_W(x)$ for any W
- For an input x, compute

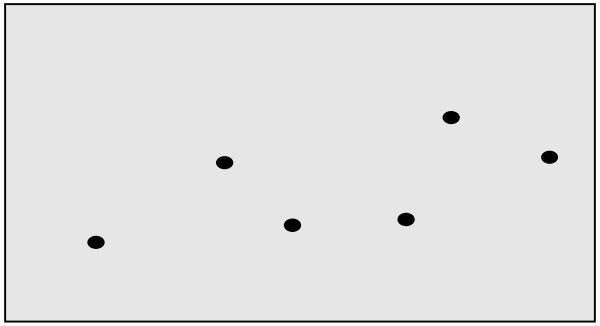
$$E_{data}[net_W(x)] = \sum_{W'} net_{W'}(x) P(W'|data)$$

- I.e., compute the weighted average of the predictions for all the possible W', weighting by how plausible the W' is

“All the possible W’s”?

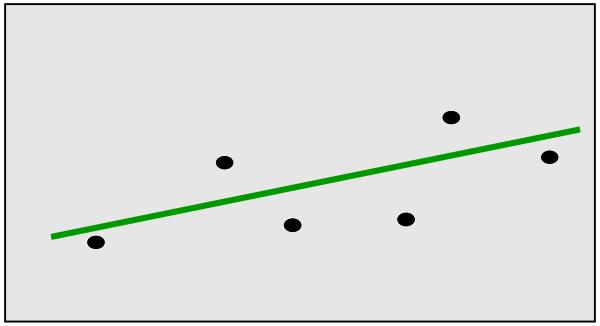
- W is a set of weights (real numbers)
- Can’t compute the probability for all the possible W’s
- Instead allow each W_j to have a finite set of values (e.g., [-10.0, -9.9, -9.8, …. 9.8, 9.9, 10.0]), and compute the probabilities for all the possible *discretized* W’s
- For 30 weights, the above would still be 200^{30}computations!
  - Not really practical except for very small networks.

Why bother?



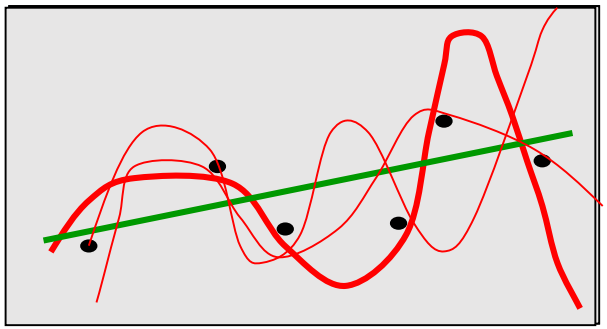
- We happen to believe that the data was generate by a degree 5 polynomial (+noise)

Why bother?



- Fitting a single polynomial would overfit horribly: much better to just fit a straight line

Why bother?



- Fitting multiple polynomials and averaging would do the trick: no overfitting on the one hand, and no underfitting (because the data doesn't *really* lie a long a straight line) on the other

How to make it practical?

$$\sum_{W'} net_{W'}(x) P(W'|data)$$

- Instead of computing this weighted sum for *all* W', only compute it for *some* W'
- Pick the W' uniformly at random
- Or: pick the W' according to P(W'|data), and compute the average $net_{W'}(x)$ that we get

Monte Carlo Methods

- Picking the possible parameters (e.g. W') and computing some functions of them (e.g. $net_{W'}(x)$) and then computing the averageto estimate the true value (e.g., $net_W(x)$ for the true W) is known as a Monte Carlo method
  - After the casino in Monaco

Enrico Fermi and Insomnia

- “Enrico Fermi (1901–1954) took great delight in astonishing his colleagues with his remarkably accurate predictions of experimental results. . . . . his “guesses” were really derived from the statistical sampling techniques that he used to calculate with whenever insomnia struck!”

  —*The beginning of the Monte Carlo method*, N. Metropolis