Markov Chain Monte Carlo



Slides from Geoffrey Hinton and Iain Murray CSC411: Machine Learning and Data Mining, Winter 2017

Michael Guerzhoy

Motivation

- Want to estimate $\sum_{W'} net_{W'}(x) P(W'|data)$
- Strategy: pick W' randomly according to P(W'|data), and average all the $net_{W'}(x)$ we get
- Sampling W' according to P(W'|data) is hard!
 - For most W', P(W'|data) is basically equal to 0
 - Hard to find the W' for which P(W'|data) is not equal to 0
 - Those are the W' we should pick!

Metropolis Algorithm

- Goal: obtain samples from $P(\theta)$
 - I.e., obtain $\theta^{(s+1)}$, $\theta^{(s+2)}$, ..., $\hat{\theta}^{(s+m)}$ that are distributed according to $P(\theta)$

Metropolis Algorithm

- $\theta' \sim q(\theta'; \theta^{(s)})$ (Obtain a proposed θ')
 - Simplest q: normal distribution around $\theta^{(s)}$. q must be symmetric: $q(\theta';\theta^{(s)})=q(\theta^{(s)};\theta')$
- if accept:
 - $\theta^{(s+1)} \leftarrow \theta'$
- else:
 - $\theta^{(s+1)} \leftarrow \theta^{(s)}$
- With $Prob(accept) = \min\left(1, \frac{P^*(\theta')}{P^*(\theta^{(s)})}\right)$
 - $P^*(\theta) \propto P(\theta)$
 - For Neural Networks, this is proportional to $P(W|data)\,$ -- can ignore the denominator

Metropolis Intuition

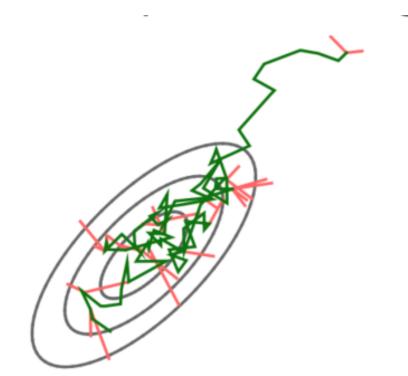
$$Prob(accept) = \min\left(1, \frac{P^*(\theta')}{P^*(\theta^{(s)})}\right)$$

- Tries to perturb $\theta^{(s)}$ and see if the new θ' isn't more likely $\frac{P^*(\theta')}{P^*(\theta^{(s)})}$. If it is, accept. If it's not, accept with a lower probability
- Makes sure that if $\theta^{(s)}$ is sampled according to $P(\theta)$, $\theta^{(s+1)}$ is as well

Metropolis Intuition

- For large s (i.e., after many steps), $P(\theta^{(s)})$ is likely large
- In fact $\theta^{(s+1)}$, ..., $\theta^{(s+m)}$ looks like it's sampled according to $P(\theta^{(s)})$
- The Metropolis Algorithm is an example of a Monte Carlo Markov Chain (MCMC) algorithm

Illustration



$$\theta' \sim q(\theta'; \theta^{(s)})$$

if accept:

$$\theta^{(s+1)} \leftarrow \theta'$$

else:

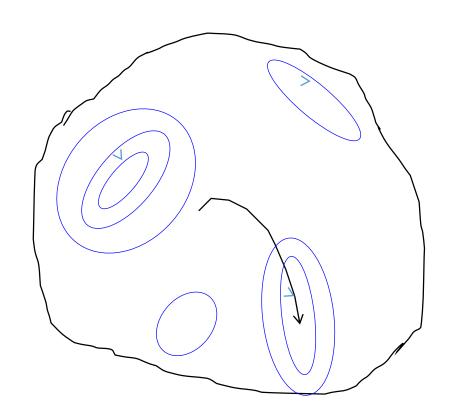
$$\theta^{(s+1)} \leftarrow \theta^{(s)}$$

$$P(\text{accept}) = \min \left(1, \frac{\pi^*(\theta') \, q(\theta^{(s)}; \theta')}{\pi^*(\theta^{(s)}) \, q(\theta'; \theta^{(s)})} \right)$$

Adding the q is a generalization

Sampling weight vectors

- In standard backpropagation we keep moving the weights in the direction that decreases the cost.
 - i.e. the direction that increases the log likelihood plus the log prior, summed over all training cases.
 - Eventually, the weights settle into a local minimum or get stuck on a plateau or just move so slowly that we run out of patience.



One method for sampling weight vectors

- Suppose we add some Gaussian noise to the weight vector after each update.
 - So the weight vector never settles down.
 - It keeps wandering around, but it tends to prefer low cost regions of the weight space.
 - Can we say anything about how often it will visit each possible setting of the weights?

