This is the title

Author One
Author Two

29 February 2004

1 Energy-Based Model (EBM)

EBM mengaitkan sebuah energi skalar pada setiap konfigurasi variable yang diinginkan. Proses learning bertujuan untuk memodifikasi fungsi energi sehingga bentuknya memiliki sifat yang diinginkan. Sebagai contoh, misalnya diinginkan sebuah bentuk konfigurasi yang memiliki energi yang rendah, maka model probabilistik dari EBM didifinisikan sebagi distribusi probabilitas melalui fungsi energi sebagi berikut:

$$p(x) = \frac{e^{-E(x)}}{Z}. \tag{1}$$

Z adalah faktor normalisasi yang disebut sebagai fungsi partisi untuk menganalogikan dengan sistem fisika.

$$Z = \sum_x e^{-E(x)}$$

EBM bisa dilatih dengan cara melakukan (stochastic) gradient descent pada negative log-likelihood (NLL)-nya secara empiris pada data training. Adapun untuk logistic regression akan didifinisikan terlebih dahulu log-likelihood $\mathcal{L}(\theta, \mathcal{D})$ dan fungsi loss-nya sebagai NLL $\ell(\theta, \mathcal{D})$ sebagai berikut:

$$\mathcal{L}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x^{(i)} \in \mathcal{D}} \log\ p(x^{(i)})$$
$$\ell(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D})$$

Menggunakan stochastic gradient $-\frac{\partial \log p(x^{(i)})}{\partial \theta}$, dimana θ adalah parameter dari modelnya.

1.1 EBM dengan Hidden Units

Pada banyak kasus, sampel x biasanya tidak terobservasi secara penuh, atau akan ditambahkan varibel yang tidak teobservasi secara langsung yang disebut dengan hidden unit, dimana hal ini berguna untuk meningkatkan ekspresivitas dari model. Sehingga dikenalkan bagian yang terobservasi disini dilambangkan dengan x, dan sebuah bagian yang tersembunyi h. Sehingga bisa ditulis sebagai:

$$P(x) = \sum_h P(x, h) = \sum_h \frac{e^{-E(x,h)}}{Z}. \tag{2}$$

Pada kasus ini, untuk melakukan pemetaan rumus yang mirip dengan rumus (1), akan dikenalkan notasi (yang merupakan inspirasi dari fisika) yaitu free energy $\mathcal{F}(x)$, yang didifinisikan sebagai berikut:

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)} \tag{3}$$

Sehingga bisa diturunkan sebagai :

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \text{ dengan } Z = \sum_x e^{-\mathcal{F}(x)}.$$

Data dari gradien NLL kemudian memiliki bentuk yang menarik yaitu:

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial \mathcal{F}(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}. \tag{4}$$

Gradien tiatas memiliki dua istilah, dimana hal tersebut mereferensikan pada fase positif dan fase negatif. Istilah positif dan negatif ini tidak merujuk pada tanda (positif/negatif) persamaan, akan tetapi merefleksikan efek pada kepadatan probabilitas yang didefinisikan oleh model. Istilah pertama, menambah probabilitas data training (dengan cara mengurangi free energy yg berhubungan), sedangkan istilah kedua mengurangi probabilitias sampel yang digenerasi oleh model.

Biasanya sulit untuk menentukan gradien secara analitis, oleh karena berhubungan dengan komputasi dari $E_P[\frac{\partial \mathcal{F}(x)}{\partial \theta}]$. Dikarenakan hal ini merupakan ekspektasi semua kemungkinan konfigurasi input x (pada distribusi P

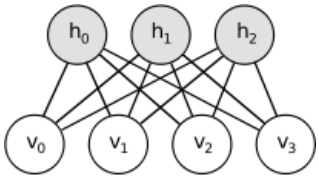

Figure 1: Grafik yang Menggambarkan RBM

yang dibentuk oleh model).
Oleh karena itu, langkah pertama agar bisa dikomputasi secara analitis maka dilakukan estimasi ekspektasi menggunakan jumlah yang pasti dari sampel pada model. Sampel digunakan untuk mengestimasi gradien dari fase negatif yang direferensikan sebagai partikel negatif, dimana disimbolkan sebagai $\mathcal{N}$. Kemudian, gradien bisa ditulis sebagai :

$$-\frac{\partial \log p(x)}{\partial \theta} \approx \frac{\partial \mathcal{F}(x)}{\partial \theta} - \frac{1}{|\mathcal{N}|} \sum_{\tilde{x} \in \mathcal{N}} \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}. \tag{5}$$

Dimana secara ideal, elemen seperti $\tilde{x}$ dari $\mathcal{N}$ disampel menurut P (sebagai contoh adalah menggunakan teknik sampling Monte-Carlo). Dengan rumus diatas, secara praktis hampir bisa melakukan algoritma stochastic, hanya saja partikel negatif $\mathcal{N}$ belum bisa diekstraksi. Oleh karena itu, pada literatur dengan metode Markov Chain Monte Carlo, sangat bagus digunakan pada model Restricted Boltzmann Machine (RBM) yang merupakan bentuk spesifik dari model EBM.

2 Restricted Boltzmann Machines (RBM)

Boltzmann Machines (BMs) adalah bentuk khusus dari log-linear Markov Random Field (MRF), dengan kata lain, dimana fungsi energi adalah linear pada parameter bebasnya. Agar membuat BM cukup bisa merepresentasikan distribusi yang kompleks(dengan kata lain, berangkat dari setting parameter yang terbatas kepada non paramter), diasumsikan bahwa beberapa variabel tidak terobserbasi sehingga disebut hidden. Dengan memiliki variabel hidden, bisa dilakukan peningkatan kapasitas model dari BM. RBM, selanjutnya membuat BM yang terbatas pada variabel tanpa koneksi visibel-visibel dan hidden-hidden. Seperti pada gambar 1

Fungsi energi $E(v, h)$ pada RBM didefinisikan sebagai persamaan 6.

$$E(v, h) = -b'v - c'h - h'Wv \tag{6}$$

Dimana W merepresentasikan bobot yang terkoneksi antara unit hidden dan visible dan b, c adalah bias dari visible dan hidden secara berurutan.

Hal ini bisa diterjemahkan dalam bentuk persamaan energi bebas $\mathcal{F}(v)$ seperti dibawah:

$$\mathcal{F}(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)}.$$

Dikarenakan struktur RBM yang spesifik, visibel dan hidden adalah independen secara bersyarat antara satu dengan lainnya. Dengan menggunakan sifat tersebut, maka dapat dituliskan :

$$p(h|v) = \prod_i p(h_i|v)$$

$$p(v|h) = \prod_j p(v_j|h).$$

2.1 RBMs yang Menggunakan Unit Biner

Kasus umum jika menggunakan unit biner (dimana v_j dan $h_i \in \{0, 1\}$), yang didapat dari persamaan (6) dan (2), versi probabilistik dari fungsi aktivasi neuron adalah sebagai berikut:

$$P(h_i = 1|v) = sigm(c_i + W_i v) \tag{7}$$

$$P(v_j = 1|h) = sigm(b_j + W'_j h) \tag{8}$$

Selanjutnya, energi bebas dari RBM dengan unit biner, disederhanakan menjadi persamaan:

$$\mathcal{F}(v) = -b'v - \sum_i \log(1 + e^{(c_i + W_i v)}). \tag{9}$$

2.2 Update Persamaan dengan Unit Biner

Menghubungkan persamaan (5) dengan (9), didapatkan gradien log-likelihood untuk RBM dengan unit biner sebagai berikut:

$$-\frac{\partial \log p(v)}{\partial W_{ij}} = E_v[p(h_i|v) \cdot v_j] - v_j^{(i)} \cdot sigm(W_i \cdot v^{(i)} + c_i) \tag{10}$$

$$-\frac{\partial \log p(v)}{\partial c_i} = E_v[p(h_i|v)] - sigm(W_i \cdot v^{(i)})$$

$$-\frac{\partial \log p(v)}{\partial b_j} = E_v[p(v_j|h)] - v_j^{(i)}$$

3 Sampling pada RBM

Sampel dari $p(x)$ bisa didapat dengan menjalankan Markov chain sampai konvergen dengan menggunakan gibbs samping sebagai operator transisi.

Gibbs sampling dari join variable random sebanyak N dari $S = (S_1, ..., S_N)$ merupakan urutan sebanyak N sampling dari sub-steps dalam bentuk $S_i \sim p(S_i|S_{-i})$ dimana S_{-i} berisi $N-1$ variabel random lain didalam S tetapi diluar S_i.

Untuk RBM, S berisi himpunan dari visible dan hidden unitnya. Akan tetapi, dikarenakan unit ini dipenden secara kondisional, maka salah satunya bisa dilakukan gibbs sampling. Pada setting disini, unit visible disampel secara simultan given nilai fix dari hidden unitnya. Demikian sebaliknya, hidden unitnya disampel secara simultan given unit visibelnya.Sehingga satu langkah Markov chain adalah sebagai berikut:

$$h^{(n+1)} \sim sigm(W'v^{(n)} + c)$$
$$v^{(n+1)} \sim sigm(Wh^{(n+1)} + b),$$

Dimana $h^{(n)}$ menunjik pada himpunan semua hidden unit pada nilai yang ke-n langkah dari Markov chain. Yang artinya adalah sebagai contoh, $h_i^{(n+1)}$ adalah secara random dipilih antara 1 (versus 0) dengan nilai probabilitas $sigm(W_i'v^{(n)} + c_i)$, demikian juga, $v_j^{(n+1)}$ adalah dipilih secara random antara 1 (versus 0) dengan probabilitas $sigm(W_{.j}h^{(n+1)} + b_j)$.

Hal ini seperti digambarkan pada gambar 2

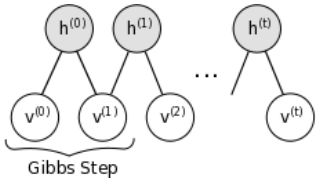

Figure 2: Gibbs Sampling

Oleh karena $t \to \infty$, maka sampel $(v^{(t)}, h^{(t)})$ bisa dipastikan akan akurat dalam mensampel $p(v, h)$.

Secara teori, tiap parameter diupdate pada proses learning dibutuhkan satu rantai tersebut untuk konvergen. Akan tetapi hal ini sangat mahal komputasinya. Sehingga banyak diajukan algoritma untuk melatih RBM agar sampel $p(v, h)$ efisien, disaat proses learningnya.

4 Contrastive Divergence (CD-k)

Contrastive Divergence(CD) menggunakan trik untuk mempercepat proses sampling:

Contrastive Divergence uses two tricks to speed up the sampling process: Dikarenakan yang diinginkan adalah $p(v) \approx p_{train}(v)$ (distribusi data yang asli), initialisasi Markov chain dengan contoh data training (dimana, berasal dari distribusi yang mendekati p, pada distribusi final dari p). CD tidak menunggu rantai untuk konvergen. Sampel didapatkan setalah langkah ke-k dari Gibbs sampling. Pada prakteknya, $k = 1$ secara mengejutkan sudah menghasilkan hasil yang baik.

5 Persistent CD

Persistent CD (P-CD) [Tieleman08] menggunakan pendekatan lain untuk mensampling $p(v, h)$. Hal ini bergantung hanya pada Markov chain tunggal, yang memiliki kondisi yang persisten (dimana, tidak melakukan restart chain pada setiap sampel yang terobservasi). Pada setiap umpdate parameter, akan di ekstraksi sampel baru dengan penjalankan chain pada langkah ke-k. Kondisi chain akan dipertahankan pada update selanjutnya.

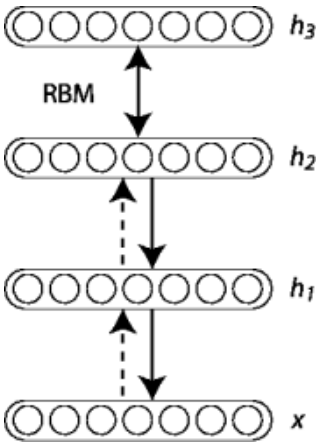

Figure 3: Arsitektur Deep Believe Network (DBN) yang merupakan gabungan dari RBM

Intuisinya adalah jika update parameternya cukup kecil dibaningkan dengan rate campuran dari Markov Chain, maka hal ini bisa mengejar perubahan modelnya.

6 Deep Belief Networks

Hinton menunjukkan bahwa RBM bisa dijajar dan dilatih secara greedy untuk membentuk sebuah jaringan yang dinamakan dengan Deep Belief Network (DBN). DBN adalah model grafis dimana bisa melakukan learning untuk mengekstraksi representasi hirarki yang mendalam (deep) dari data training. Hal ini memodelkan distribusi gabungan antara vektor x sebagai observer dan ℓ layer hidden h^k sebagai berikut:

$$P(x, h^1, \ldots, h^\ell) = \left(\prod_{k=0}^{\ell-2} P(h^k | h^{k+1}) \right) P(h^{\ell-1}, h^\ell) \tag{11}$$

Dimana $x = h^0, P(h^{k-1}|h^k)$ adalah distribusi kondisional untuk unit visible dikondisikan pada unit hidden pada level k dan $P(h^{\ell-1}, h^\ell)$ adalah distribusi gabungan visible-hidden pada level teratas dar RBM. Seperti diilustrasikan pada gambar 3.

Prinsip dari greedy layer-wise unsupervised training bisa di aplikasikan pada DBN dengan RBM sebagai bagian pada tiap layernya [hinton] [bengio].

Pada prinsipnya prosesnya adalah sebagai berikut:

1. Latih layer pertama ssebagai RBM yang memodelkan input $x = h^{(0)}$ sebagai visible layernya.

2. Gunakan layer pertama untuk mendapatkan representasi input yang digunakan sebagai data untuk layer kedua. Ada dua solusi yang sama. Representasi ini bisa dipilih sebagai rata-rata dari aktivasi $p(h^{(1)} = 1|h^{(0)})$ atau sampel dari $p(h^{(1)}|h^{(0)})$.

3. Train layer kedua sebagai RBM dengan mengambil data transformasi (sampel atau rata-rata aktivasi) sebagai training (untuk layer visible dari RBM tersebut).

4. Iterasikan (2 dan 3) untuk semua layer yang diinginkan, setiap waktu dengan mempropagasikan keatas antara sampel atau nilai rata-ratanya.

5. Fine-tune semua parameter dari arsitektur dengan log-likelihood DBN atau dengan kriteria secara supervised setelah menambahkan layer supervised untuk memprediksikan kelas, sebagai contoh misalnya layer logistic regression.

Pada kasus ini, akan difokuskan pada fine-tuning dengan melakukan gradien descent menggunakan klassifier logistic regression dimana digunakan untuk mengklasifikasikan input x berdasar pada output dari hidden layer $h^{(l)}$ dari DBN. Fine-tune kemudian dilakukan melalui gradien descent dari NLL fungsi costnya. Dikarenakan gradien secara supervised adalah hanya non-null untuk bobot dan bias pada hidden layer pada tiap-tiap layer, maka prosedur ini serupa dengan menerapkan initialisasi parameter dari arsitektur MLP yang deep dengan bobot dan bias dari hidden layer yang didapat pada proses training unsupervised diatas.

7 Alasan Melakukan Training Secara Greedy Layer-Wise

Algoritma training deep learning secara greedy layer-wise terbukti bisa bekerja dengan baik, sebagai contoh 2 layer DBN dengan hidden layer $h^{(1)}$ dan

$h^{(2)}$ dengan parameter bobot berurutan adalah $W^{(1)}$ dan $W^{(2)}$, (**?**) maka $\log p(x)$ bisa ditulis sebagai:

$$\begin{aligned}\log p(x) =& KL(Q(h^{(1)}|x)||p(h^{(1)}|x)) + H_{Q(h^{(1)}|x)} + \\ & \sum_h Q(h^{(1)}|x)(\log p(h^{(1)}) + \log p(x|h^{(1)})).\end{aligned} \tag{12}$$

$KL(Q(h^{(1)}|x)||p(h^{(1)}|x))$ merepresentasikan KL divergence antara posterior $Q(h^{(1)}|x)$ dari RBM pertama jika hal ini sendirian, dan probabilitas $p(h^{(1)}|x)$ untuk layer sayng sama tapi didifinisikan oleh keseluruhan DBN (sebagai contoh, perhitungan prior $p(h^{(1)}, h^{(2)})$ didefinisikan sebagai top-level RBM). $H_{Q(h^{(1)}|x)}$ adalah entropy dari distribusi $Q(h^{(1)}|x)$.
Hal ini bisa ditunjukkan bahwa jika diinitialisasi kedua layer hidden sehingga $W^{(2)} = {W^{(1)}}^T, Q(h^{(1)}|x) = p(h^{(1)}|x)$ dan KL divergence nya adalah null. Maka jika di lakukan learning pada level awal RBM dan kemudian parameter $W^{(1)}$ dibuat tetap, kemudian dilakukan optimasi pada persamaan 12 terhadap $W^{(2)}$ bisa meningkatkan likelihood dari $p(x)$. Jika diisolasi hanya pada $W^{(2)}$ sehinggi didapatkan:

$$\sum_h Q(h^{(1)}|x)p(h^{(1)})$$

Melakukan optimasi persamaan ini dengan memperhatikan jumlah $W^{(2)}$ training pada tingkat RBM selanjutnya, menggunakan output dari $Q(h^{(1)}|x)$ sebagai distribusi training untuk RBM yang pertama.