**LECTURE NOTES**

**Selected Topics in Computational Intelligence I**

**Session 8**

**Supervised Neural Networks**

**LEARNING OUTCOMES**

**Tujuan Instruksional Umum :**

1. *Mahasiswa mampu menjelaskan tentang supervised learning pada neural network*

**Tujuan Instruksional Khusus :**

1. *Mahasiswa dapat menjelaskan perbedaan antara supervised learning*
2. *Mahasiswa dapat membuat program supervised learning*

**8.1 Supervised Learning**

Supervised Learning membutuhkan training set yang terdiri dari vektor input dan target vektor yang terkait dengan setiap vektor input. Pembelajar NN menggunakan target vektor untuk menentukan seberapa baik ANN telah belajar, dan untuk membimbing penyesuaian terhadap nilai-nilai bobot (weight) untuk mengurangi error secara keseluruhan. Learning terdiri dari menyesuaikan bobot sampai kesalahan empiris diterima telah tercapai. NN yang disupervisi antara lain standard multilayer NNs (feedforward), functional link NNs, simple recurrent NNs, time-delay NNs, product unit NN, dan cascade networks. **Time-delay neural network** (TDNN) dikenal sebagai backpropagation through- time, merupakan jaringan temporal dimana bentuk input secara suksesif ditunda terhadap waktu. Ada dua jenis algoritma supervised learning, didasarkan pada saat bobot diperbarui:

• **Stochastic / online learning**, di mana bobot disesuaikan setelah setiap pola presentasi. Dalam hal ini pola input berikutnya dipilih secara acak dari training set, untuk mencegah bias yang mungkin terjadi karena urutan pola terjadi pada training set.   
• **Batch / offline learning,** di mana perubahan berat badan diakumulasikan dan digunakan untuk menyesuaikan bobot hanya setelah semua pola pelatihan telah disajikan.

**8.2 Feed- Forward Network**

Jika seluruh koneksi dari *output* suatu layer ke *input* layer berikutnya, dan tidak ada koneksi di dalam layer yang sama atau koneksi dari sebuah layer ke layer sebelumnya, ini disebut *network feed-forward*. *Network Feed forward* digunakan untuk tipe neural network yang paling mudah. *Feed forward* ini diusulkan oleh Rosenblatt pada tahun 1962, namun hanya mempunyai 2 layer, yaitu layer *input* dan *output* yang disebut *perceptron*.



**Gambar 3.1 Feed forward network**

Output dari FFNN untuk pattern input dihitung dengan sebuah forward pass melalui network. Untuk tiap unit output :

=)



Dimana dan fungsi aktivasi untuk unit output dan hidden unit ; adalah bobot diantara unit output dan hiden unit ; nilai dari unit input dari pattern input . (I+1)-th merupakan unit input dan (J+1)-th hidden unit merupakan unit bias merepresentasikan nilai threshold neuron di layer berikutnya.

**8.3 Optimisasi Gradient Descent**

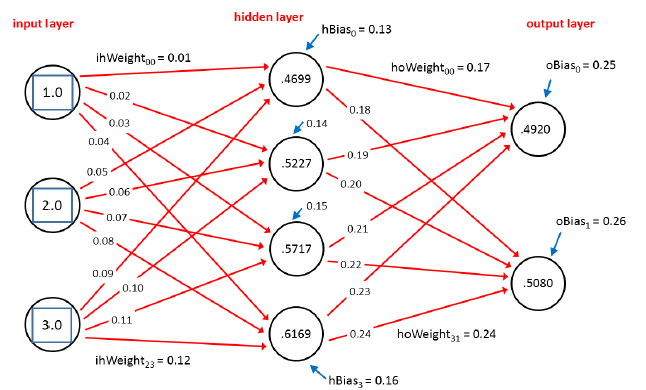
Optimasi ***Gradient descent*** (GD) telah menghasilkan salah satu algoritma pembelajaran yang paling populer, yaitu backpropagation, dipopulerkan oleh Werbos. Learning iteration (satu iterasi belajar disebut sebagai epoch) terdiri dari dua tahap:

1. **Feedforward pass**, yang hanya menghitung nilai output (s) dari NN untuk setiap pola pelatihan).

2.**Backward pass**, yang menyebar nilai error kembali dari lapisan output ke lapisan input.

**Contoh Aplikasi :** FeedForward Network

Aplikasi d ibawah menampilkan beberapa nilai intermediate selama perhitungan output. Nilai-nilai menengah, pre-activation hidden sum, hidden output , dan pre-activation output sums biasanya tidak akan ditampilkan dan yang ada hanya untuk membantu Anda memahami mekanisme feedforward.



**Gambar 3.2 Rancangan feedforward**

Ada tiga input dummy dengan nilai 1.0, 2.0, dan 3.0. Setiap garis merah yang menghubungkan satu node ke yang lain merupakan konstanta weight. Paling atas weight dengan nilai 0.01 diberi label ihWeight [0] [0], yang berarti berat dari node input 0 sampai hidden node 0. Langkah pertama dalam mekanisme feedforward adalah untuk menghitung nilai-nilai yang ditampilkan dalam lapisan hidden node. Nilai hidden node 0 adalah 0,4699 dan dihitung sebagai berikut. Pertama, produk dari setiap nilai masukan dan bobot yang terkait dijumlahkan:

**hSums[0] = (1.0)(0.01) + (2.0)(0.05) + (3.0)(0.09) = 0.01 + 0.10 + 0.27 = 0.38**

Berikutnya, nilai bias yang sesuai ditambahkan:

**hSums[0] = 0.38 + 0.13 = 0.51**

Anda dapat melihat ini ditampilkan sebagai **pre-activation sum** yang pertama. Selanjutnya, yang disebut fungsi aktivasi diterapkan untuk sum(jumlah). Fungsi aktivasi akan dijelaskan secara rinci nanti, tapi untuk saat ini cukup untuk mengatakan bahwa fungsi aktivasi adalah fungsi tangen hiperbolik, yang biasanya disingkat **tanh**.

**hOutputs[0] = tanh(0.51) = 0.4699**

Dapat dilihat bahwa.4699 ditampilkan sebagai hidden output pertama, juga sebagai nilai dari top-most hidden node, dilanjutkan perhitungan sisanya sebagai berikut:

**hOutputs[1] = tanh( (1.0)(0.02) + (2.0)(0.06) + (3.0)(0.10) + 0.14 ) = tanh(0.58) = 0.5227**

**hOutputs[2] = tanh( (1.0)(0.03) + (2.0)(0.07) + (3.0)(0.11) + 0.15 ) = tanh(0.65) = 0.5717**

**hOutputs[3] = tanh( (1.0)(0.04) + (2.0)(0.08) + (3.0)(0.12) + 0.16 ) = tanh(0.72) = 0.6169**

Setelah semua nilai-nilai output hidden node telah dihitung, nilai-nilai ini digunakan sebagai masukan ke lapisan output. Nilai node output dihitung sedikit berbeda dari hidden node. Jumlah yang keluaran awal, sebelum aktivasi, untuk node keluaran 0 dan 1 dihitung dengan cara yang sama seperti awal jumlah hidden node, dengan menjumlahkan produk dari input dan bobot dan kemudian menambahkan nilai Bias:

**oSums[0] = (0.4699)(0.17) + (0.5227)(0.19) + (0.5717)(0.21) + (0.6169)(0.23) + 0.25 = 0.6911**

**oSums[1] = (0.4699)(0.18) + (0.5227)(0.20) + (0.5717)(0.22) + (0.6169)(0.24) + 0.26 = 0.7229**

Fungsi aktivasi untuk lapisan output disebut fungsi softmax. Hal ini akan dijelaskan kemudian, tapi untuk saat ini softmax setiap jumlah awal(preliminary sum) dijelaskan dengan contoh:

**output[0] = exp(0.6911) / (exp(0.6911) + exp(0.7229)) = 1.9959 / (1.9959 + 2.0604) = 0.4920**

**output[1] = exp(0.7229) / (exp(0.6911) + exp(0.7229)) = 2.0604 / (1.9959 + 2.0604) = 0.5080**

Jika Anda melihat gambar, Anda akan melihat ini adalah nilai-nilai hasil akhir dihitung dengan demo NN. Di sini, fungsi exp dari beberapa nilai x adalah matematika konstan e = 2,71828. . . pangkat xth. Perhatikan dua nilai output berjumlah 1,0, yang bukan kebetulan dan merupakan titik menggunakan fungsi **softmax**. Untuk meringkas, jaringan saraf pada dasarnya adalah sebuah fungsi matematika yang output ditentukan oleh nilai input, sejumlah node tersembunyi, satu set bobot dan nilai bias, dan dua fungsi aktivasi.

**8.4 Bias Values as Special Weights**

Anda harus perhatikan bagaimana nilai-nilai bias ditangani. Pada bagian sebelumnya, nilai bias diperlakukan sebagai konstanta yang berbeda. Namun, sebagian besar penelitian NN memperlakukan bias bobot khusus yang terkait dengan nilai masukan dummy 1.0. Misalnya, pada gambar, nilai bias untuk hidden node 0 adalah 0,13. Pra-aktivasi sum untuk tersembunyi node 0, jika bias diperlakukan sebagai bias benar, dihitung sebagai:

**hSum [0] = (1.0) (0.01) + (2.0) (0.05) + (3.0) (0.09) + 0.13 = 0.51**

Jika Anda membayangkan dummy input node dengan nilai 1.0 di samping tersembunyi node 0, dan mempertimbangkan bias sebagai special weight, jumlah pre-aktivasi akan dihitung sebagai:

**hSum [0] = (1.0) (0.01) + (2.0) (0.05) + (3.0) (0.09) + (1,0) (0.13) = 0.51**

Dengan kata lain, hasilnya adalah sama. Point dari memperlakukan bias sebagai special weight adalah bahwa hal tersebut menyederhanakan banyak bukti matematika dan turunan dalam penelitian. Program demo menggunakan 26 bobot dummy dan nilai-nilai bias dan kemudian menempatkan mereka ke dalam NN menggunakan metode SetWeights.

Program demo menggunakan fungsi tangen hiperbolik untuk dan fungsi softmax untuk lapisan output simpul aktif(output layer node activation). Ada fungsi aktivasi umum ketiga disebut fungsi

sigmoid logistik. Ketiga fungsi aktivasi ini berkaitan erat dengan matematis. Meskipun ada beberapa pengecualian, secara umum fungsi tangen hiperbolik adalah **pilihan terbaik** **untuk hidden layer activation**. Untuk lapisan aktivasi output, jika jaringan saraf Anda berkinerja klasifikasi di mana variabel dependen yang diprediksi memiliki tiga atau lebih nilai (misalnya, memprediksi kecenderungan politik seseorang yang dapat "liberal", "moderat", atau "konservatif"), aktivasi softmax adalah pilihan terbaik.

Fungsi sigmoid logistik, juga disebut log-sigmoid, mirip dengan fungsi tangen hiperbolik. Fungsi sigmoid logistik menerima apapun nilai numerik dari infinity negatif ke positif tak terhingga, dan mengembalikan nilai antara 0,0 dan 1,0. Berbeda dengan tangen hiperbolik, kebanyakan bahasa pemrograman tidak memiliki fungsi sigmoid logistik built-in. Definisi matematika dari fungsi sigmoid logistik adalah: f (x) = 1 / (1 + e-x).

Kebanyakan bahasa pemrograman, termasuk C #, memiliki fungsi built-in, biasanya bernama Exp. Jadi, kode fungsi sigmoid logistik adalah:

**private static double LogSigmoid(double x) {**

if (x < -45.0) return 0.0;

else if (x > 45.0) return 1.0;

else return 1.0 / (1.0 + Math.Exp(-x));

}

**8.5 Training**

Tujuan utama dari NN adalah memprediksi pola yang tepat terhadap sutu masalah. Dalam rangka memprediksi, jaringan saraf pertama harus dilatih. Pelatihan jaringan saraf adalah proses menemukan satu set bobot yang baik dan nilai bias sehingga output beberapa data pelatihan sesuai dengan output yang dihitung dengan menggunakan bobot dan nilai bias. Model kemudian dapat digunakan untuk memprediksi output untuk input sebelumnya tanpa memerlukan nilai output.

Salah satu contoh demo ANN sederhana adalah memprediksi spesies dari bunga iris dari empat atribut numerik bunga: panjang kelopak, lebar kelopak, panjang sepal, dan lebar sepal. Kelopak adalah apa yang kebanyakan orang akan mempertimbangkan bagian bunga sebenarnya dari iris. Sepal adalah struktur hijau yang dapat Anda pikirkan sebagai daun khusus. Ada tiga spesies yang mungkin: Iris Setosa, Iris versicolor, dan Iris virginica. Kumpulan data pertama kali diterbitkan pada tahun 1936 oleh matematikawan Ronald Fisher dan sehingga sering disebut data Iris Fisher. Kumpulan data terdiri dari total 150 item, dengan data untuk 50 dari masing-masing tiga jenis iris. Karena jaringan saraf hanya memahami data numerik, demo harus menggunakan data mana tiga kemungkinan y-nilai telah dikodekan sebagai angka. Berikut spesies Iris sestosa dikodekan sebagai (0, 0, 1), Iris versicolor dikodekan sebagai (0, 1, 0), dan spesies Iris virginica dikodekan sebagai (1, 0, 0).

Demo mengambil kumpulan data 150-item dan membagi secara acak ke dalam subset 120-item (80%) yang akan digunakan untuk pelatihan dan subset 30-item (20%) yang akan digunakan untuk pengujian(testing data), yaitu, untuk digunakan untuk memperkirakan probabilitas klasifikasi yang benar pada data yang belum pernah terlihat sebelumnya. Demo melatih 4-7-2 jaringan saraf yang terhubung sepenuhnya menggunakan algoritma back-propagasi dalam hubungannya dengan teknik yang disebut pelatihan tambahan.

Setelah pelatihan selesai, akurasi model yang dihasilkan dihitung pada training set dan test set. Model dengan benar memprediksi 97,50% dari item pelatihan (117 dari 120) dan 96,67% dari item tes (29 dari 30). Oleh karena itu, jika disajikan dengan data untuk bunga iris baru yang bukan bagian dari data pelatihan atau tes, Anda bisa memperkirakan bahwa model dengan benar akan mengklasifikasikan bunga dengan sekitar probabilitas 0,9667.

**Incremental Training**

Ada dua pendekatan utama untuk pelatihan NN. Seperti biasa, kedua pendekatan memiliki beberapa nama yang berbeda tapi dua istilah yang paling umum adalah incremental dan batch. Dalam tingkat tinggi pseudo-code, pelatihan batch:

**loop until done**

**for each training item**

**compute error and accumulate total error**

**end for**

**use total error to update weights and bias values**

**end loop**

Inti dari pelatihan batch bahwa metrik kesalahan untuk seluruh training set dihitung dan kemudian bahwa nilai error tunggal digunakan untuk memperbarui bobot semua jaringan saraf dan bias.

**loop until done**

**for each training item**

**compute error for current item**

**use item's error to update weights and bias values**

**end for**

**end loop**

Dalam **incremental training**, bobot dan bias diperbarui untuk setiap item pelatihan. Berdasarkan pengalaman, bagi kebanyakan orang, batch training tampaknya sedikit lebih logis. Namun, beberapa bukti penelitian menunjukkan bahwa pelatihan tambahan (juga disebut online training) sering memberikan hasil yang lebih baik (berarti menghasilkan model yang memprediksi lebih baik) daripada batch training. Ada pendekatan ketiga untuk pelatihan NN, yang merupakan kombinasi dari batch dan incremental training. Teknik ini sering disebut pelatihan mini-batch. Dalam pseudo-code:

**loop until done**

**loop n times**

**compute error term for current item**

**accumulate error**

**end loop**

**use current accumulated error to update weights and bias values**

**reset mini accumulated error to 0**

**end loop**

**SIMPULAN**

*Supervised learning menggunakan data pelatihan yang berisi nilai input dan output yang diinginkan.*

**DAFTAR PUSTAKA**

* Adries P. Engelbrect. (2007), ***Computational Intelligence An Introduction***. 2nd ed. John Wiley & Sons. USA.
* James McCaffrey (2014), ***Neural Network using C# Succintly***, Syncfusion Publisher.