Generalisation

Ali Akbar Septiandri

Universitas Al Azhar Indonesia

October 14, 2018

Fungsi Softmax

“Bagaimana jika terdapat lebih dari 2 kelas?”

MNIST Digit Recognition

Gambar: Mendeteksi angka dari tulisan

Multi-class Networks

- Gunakan ide one-of-k encoding
- Setiap kelas menggunakan fungsi sigmoid
- Apa masalahnya?

Multi-class Networks

- Gunakan ide one-of-k encoding
- Setiap kelas menggunakan fungsi sigmoid
- Apa masalahnya?
- $\sum_k P(y_k = 1|\mathbf{x}) \neq 1$

Softmax

$$\hat{y}_k = \frac{\exp(a_k)}{\sum_{c=1}^{K} \exp(a_c)}$$

$$a_k = \sum_{j=1}^{D} w_{kj} x_j$$

- Setiap keluaran bernilai $[0, 1]$
- Penyebutnya memastikan bahwa jumlah keluaran dari setiap kelas bernilai 1

Melatih Softmax

- Kita dapat mengembangkan fungsi error *cross-entropy* menjadi multikelas

$$E(\mathbf{w}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

- Gunakan aturan turunan berantai
- Hasil akhirnya adalah

$$\frac{\partial E}{w_{kj}} = (\hat{y}_k - y_k)x_j$$

Delta rule!

Generalisasi ke Data Baru

Generalisasi

- Berapa hidden units yang kita perlukan?
- Berapa hidden layers yang kita perlukan?
- Konfigurasi seperti apa yang dapat meminimalkan error pada data uji?

Mengestimasi Error

- Error dapat terjadi karena model terlalu fleksibel atau terlalu kaku

Mengestimasi Error

- Error dapat terjadi karena model terlalu fleksibel atau terlalu kaku
- Dalam perbandingan model, akan lebih baik jika jumlah parameter yang dibandingkan sama

Mengestimasi Error

- Error dapat terjadi karena model terlalu fleksibel atau terlalu kaku
- Dalam perbandingan model, akan lebih baik jika jumlah parameter yang dibandingkan sama
- Optimasi pada data latih tidak sama dengan optimasi pada data uji

Data Latih / Validasi / Uji

- **Data latih**: Membentuk model, mencari nilai parameter
- **Data validasi**: Mencari konfigurasi hyperparameter, data uji "sementara"
- **Data uji**: Data yang belum pernah dilihat, hanya digunakan satu kali

Mengukur Generalisasi

- Training error

$$E_{train} = -\sum_{\mathcal{D}_{train}} \sum_{k=1}^{K} y_k \log \hat{y}_k$$

- Validation error

$$E_{val} = -\sum_{\mathcal{D}_{val}} \sum_{k=1}^{K} y_k \log \hat{y}_k$$

Berhati-hatilah pada kasus overfitting!

Menghindari Overfitting

- Pilih model yang bekerja terbaik, i.e. error minimal, pada data validasi

Menghindari Overfitting

- Pilih model yang bekerja terbaik, i.e. error minimal, pada data validasi
- Menambah parameter $\sim$ menambah fleksibilitas

Menghindari Overfitting

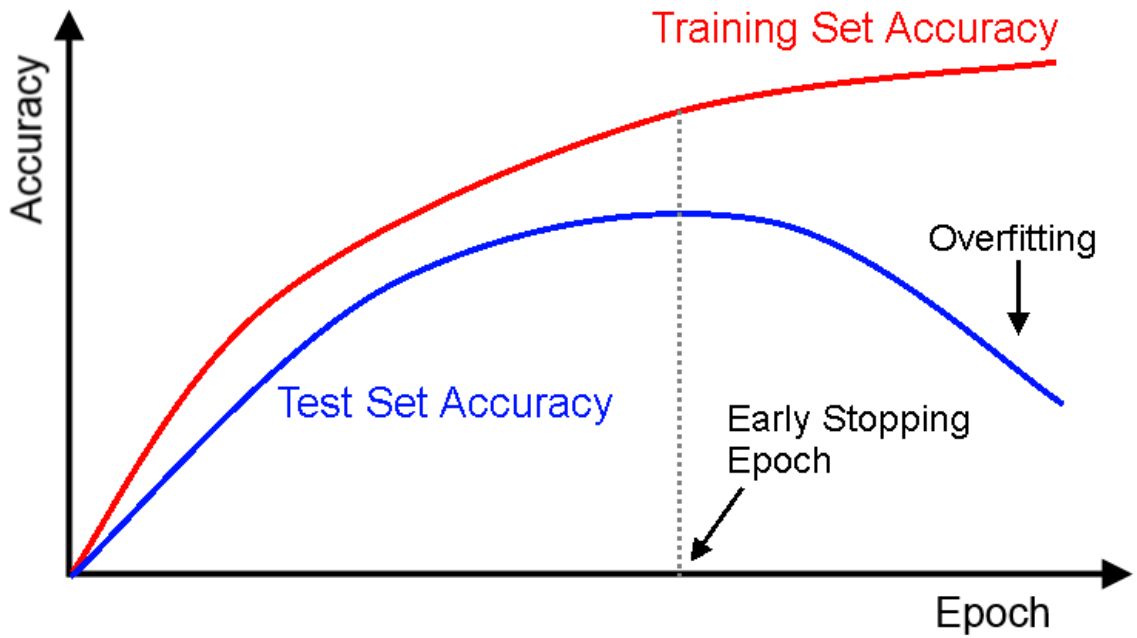
- Pilih model yang bekerja terbaik, i.e. error minimal, pada data validasi
- Menambah parameter $\sim$ menambah fleksibilitas
- Ide: Mulai dari yang fleksibel, lalu berikan batasan

Regularisasi

Epoch

- Dalam kasus *neural networks*, lebih sering digunakan iterasi terbatas
- Setiap iterasi dikenal dengan nama epoch
- Jumlah epoch sangat tergantung kasus

Early Stopping



Gambar: Menentukan jumlah epoch yang menghasilkan model terbaik secara umum

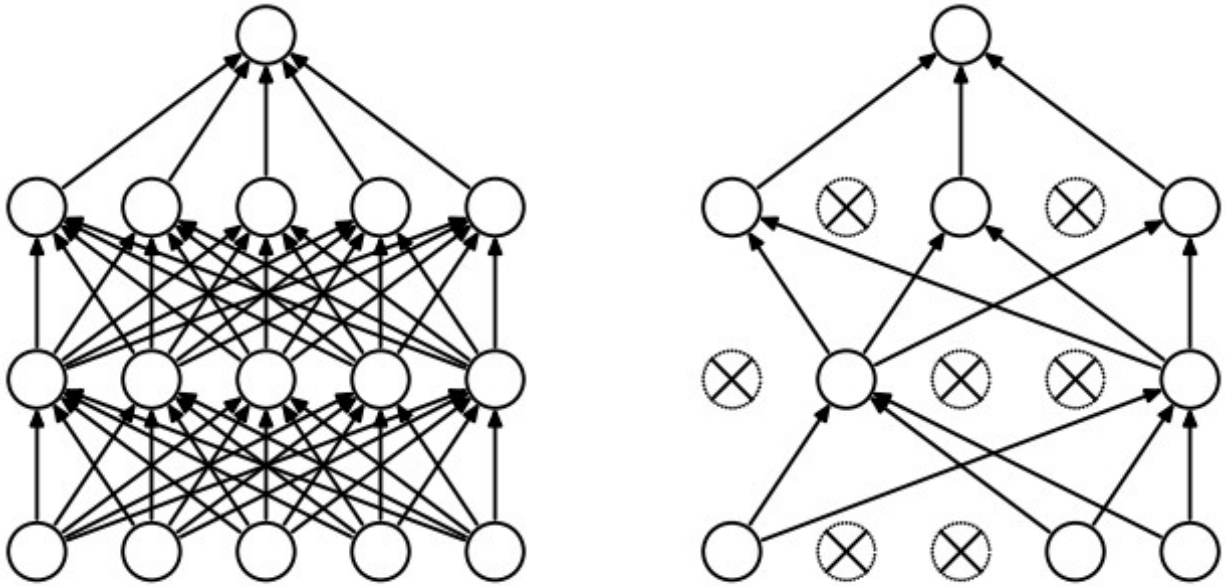
Early Stopping

- Gunakan data validasi
- Berhubungan dengan *bias-variance tradeoff*
- Berhubungan dengan fleksibilitas model

Dropout

- Cara lain untuk menghindari *overfitting*
- Ide: Matikan proses pelatihan pada *hidden units* secara acak
- Merupakan *hyperparameter* yang harus diatur nilainya
- Lihat [Srivastava et al., 2014] dan [Karpathy, 2018]

Dropout



(a) Standard Neural Net (b) After applying dropout.

Gambar: Membuat nilai masukan ke *hidden units* menjadi 0 dengan probabilitas $1 - p$ [Srivastava et al., 2014]

Referensi

N. Srivastava et al. (2015)

Dropout: a simple way to prevent neural networks from overfitting

The Journal of Machine Learning Research, 15(1), pp.1929-1958.

Andrej Karpathy (2018)

Neural Networks Part 2: Setting up the Data and the Loss

`http://cs231n.github.io/neural-networks-2/`

Terima kasih