**PENGGUNAAN OPTIMIZER NESTEROV MOMENTUM, ADAM, DAN RMSPROP PADA KLASIFIKASI ANGKA**

Dibuat untuk memenuhi tugas PR 3 mata kuliah MA4072 Pembelajaran Mendalam

Tahun Akademik 2019/2020

****

Disusun oleh :

Marcello Susanto – 10116017

Suparjo – 10116031

**PROGRAM STUDI MATEMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

1. **Pendahuluan**

**Latar Belakang**

Pada tugas sebelumnya telah dilakukan beberapa eksperimen yang mengubah komponen-komponen dari *deep learning* baik dalam data maupun parameter model yang digunakan. Komponen-komponen yang telah dieksperimenkan berupa jumlah layer pada *neural network*, fungsi aktivasi, *lost* atau *cost function*, *learning rate*, *scaling* pada data, dan lain-lain. Pada tugas kali ini, akan dieksperimenkan *optimizer* atau modifikasi dari *gradient decent* pada *update* beban dan bias dari model *neural network.*

Optimizer berperan penting dalam model neural network. *Optimizer* yang akan digunakan adalah nesterov momentum, adam, dan RMSProp. Dari ketiga *optimizer* ini, akan dicari *optimizer* yang dapat mencapai keoptimalan dari model dengan efektif.

**Tujuan**

Tujuan dari ekperimen ini adalah menerapkan beberapa variasi dari *optimizer* pada kasus klasifikasi angka, PR 1, dan akan dicari *optimizer* yang dapat mencapai keoptimalan dari model dengan efektif.

1. **Dasar Teori**

*Optimizer* adalah suatu metode untuk memodifikasi gradient decent yang diperoleh agar mencapai nilai optimal lebih cepat dengan fungsi objektif *loss function*. Akan dijelaskan beberapa *optimizer*.

**Gradient descent**

*Optimizer* yang paling sederhana karena menggunakan prinsip dari kalkulus, yaitu naik dan turun tercepat pada fungsi bervariabel banyak. Arah gerak agar naik tercepat adalah searah dengan gradient fungsi dan *vise-versa*. Pembuktian sebagai berikut

Dengan,

, Turunan berarah dari fungsi di posisi

, sudut bentukan dan

Agar meniliki turunan terbesar haruslah bernilai 1 dan agar memiliki turunan terkecil haruslah bernilai -1 atau ekivalen dengan pernyataan di atas

Sehingga, dengan menggunakan gradient decent biasa, akan diperoleh *update* titik sebagai berikut

Dengan,

, *Learning rate*

, Iterasi

Keunggulan dan kelemahan dari gradient decent adalah:

* Komputasi yang mudah
* Implementasi yang mudah
* Bisa terjebak dalam ekstema lokal
* Boros dalam *memory*

**Nesterov Momentum**

Nesterov Momentum adalah modifikasi dari gradient descent yang memiliki efek momentum dan perbaikian nilai update dengan memperhatikan loss dari titik gradient descent biasa. Efek momentum adalah suatu pembobotan yang masih memperhatikan nilai-nilai sebelumnya.

Dengan,   
, *Learning rate*

, Faktor momentum

, *Loss function*

, *Learning rate*

, bobot *update* pada epoch ke-t

Dari persamaan, dapat dilihat bahwa, loss function dioperasikan dua kali. Pertama untuk menentukan gradient descent pertama dan yang kedua untuk calon update bobot. Pembaharuan dari dikontrol oleh gradient descent dan sebelumnya. Biasanya, dipilih 0.9 agar optimal.

Keunggulan dan kelemahan dari Nesterov Momentum adalah adalah:

* Bobot pembaharuan mengecil jika semakin mendekati nilai optimal
* Membutuhkan *hyperparameter*, parameter yang ditentukan oleh user, lainnya
* Komputasi forward dilakukan dua kali

**Adam**

Adam, Adaptive Moment Estimation, adalah modifikasi dari gradient descent yang menggunakan momentum orde pertama dan kedua. Momen pertama adalah rataan dan kedua adalah variansi.

Dengan,   
, Learning rate

, Faktor momentum orde ke -i

, nilai agar akar tidak nol pada persamaan kedua biasanya nilai yang kecil

, Loss function

, bobot update pada *epoch*

Keunggulan dan kelemahan dari RMSprop adalah adalah:

* Dapat konvergen dengan sangat cepat
* Banyak *hyperparameter*
* Komputasi yang banyak

**RMSprop**

RMSprop adalah modifikasi dari gradient descent yang mempertahan bobot sebelumnya dengan menggunakan *exponential moving average*,EMA, untuk mengatur besar gradient descent yang akan diperbaharui.

Dengan,   
, *Learning rate*

, Faktor momentum

, nilai agar akar tidak nol pada persamaan kedua biasanya nilai yang kecil

, *Loss function*

, bobot update pada *epoch* ke-t

pada persamaan kedua digunakan untuk mengatur pembobotan. Pengaturan ini digunakan untuk mereduksi osilasi yang terjadi akibat learning rate yang relatif besar. Osilasi direduksi dengan menggunakan efek momentum oleh EMA pada persamaan pertama.

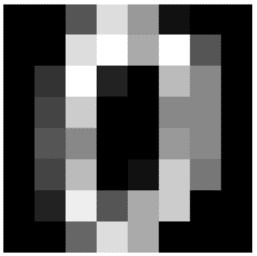
Keunggulan dan kelemahan dari RMSprop adalah adalah:

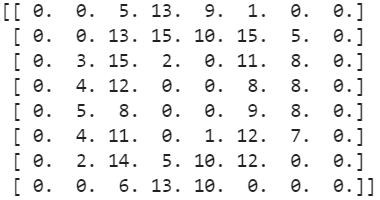
* Cenderung lebih cepat dengan kompleksitas yang kecil
* Rawan untuk terjadi eksploding atau vanishing dari pembobotan
* Membutuhkan hyperparameter baru, yaitu

1. **Eksperimen**

Eksperimen dilakukan untuk mengidentifikasi tulisan tangan berupa angka dari 0 sampai 9 yang berukuran 64 pixel. Pada eksperimen ini, akan dilakukan perbandingan dari parameter-parameter pada deep learning. Parameter-parameter ini berupa jumlah epoch, learning rate, normalisasi, fungsi aktivasi, dan fungsi kerugian. Akurasi adalah akurasi dari training, sesuai instruksi. Dimensi dari data yang diberikan adalah sebagai berikut

|  |  |
| --- | --- |
| Jenis | Ukuran |
| Ukuran Feature | 8x8 |
| Ukuran Target | 10x1 |
| Jumlah Sample | 1797 |

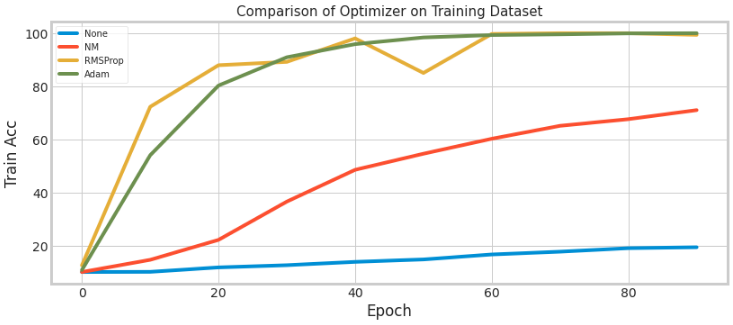




Sebelum melakukan permodelan, penulis terlebih dahulu membagi dataset menjadi dataset *training* yang digunakan untuk melatih model dan dataset *testing* yang digunakan untuk mengevaluasi performa model. Proporsi antara dataset *training* dan *testing* adalah 80 : 20. *Metrics* yang akan digunakan untuk mengevaluasi model adalah akurasi karena proporsi tiap angka yang ingin kita prediksi seimbang. Seperti yang telah dijelaskan sebelumnya di pendahuluan, eksperimen ini dilakukan juga untuk melihat perbedaan yang dihasilkan oleh penggunaan *optimizer* NesterovMomentum, Adam, dan RMSProp. Pada dasarnya *optimizer* digunakan untuk mempercepat model konvergen ke titik minimum pada proses *backward propagation* sehingga pertama-tama penulis akan menggunakan model dengan parameter sebagai berikut

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Nilai | Parameter *Optimizer* | Nilai |
| Jumlah Neuron | 128 | m\_coeff (NesterovMomentum dan RMSProp) | 0.9 |
| Learning Rate | 0.01 | beta1 (Adam) | 0.9 |
| Scaler | None | Beta2 (Adam) | 0.999 |

Dengan menggunakan *epoch* sebesar 100, didapatlah hasil sebagai berikut



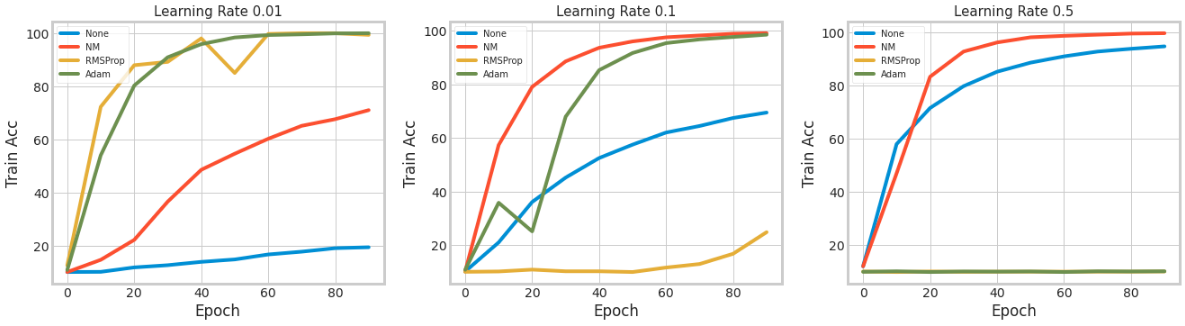
Dari grafik di atas dapat dilihat bahwa memang penggunaan *optimizer* dapat secara signifikan mempercepat kekonvergenan dari model ANN. Jika penulis tidak menggunakan *optimizer* dapat dilihat bahwa nilai akurasi dari model hanya mencapai sekitaran 20% sedangkan jika menggunakan *optimizer* nilai akurasi dari model bahka dapat mencapai 100% dengan menggunakan jumlah *epoch* yang sama.

Selain itu dapat dilihat bahwa NesterovMomentum dapat mencapai nilai akurasi sebesar 71%, masih kalah dengan RMSProp dan Adam yang dapat mencapai nilai akurasi 100%. Tetapi perhatikan bahwa pergerakan nilai akurasi dari RMSProp tidak semulus Adam, mengindikasikan bahwa Adam merupakan *optimizer* yang lebih stabil dari RMSProp.

|  |  |
| --- | --- |
| *Optimizer* | Waktu yang diperlukan |
| None | 2.13 sec |
| NesterovMomentum | 2.88 sec |
| RMSProp | 2.44 sec |
| Adam | 2.38 sec |

Tabel di atas menunjukkan waktu yang diperlukan untuk melatih model menggunakan tiap *optimizer*. Dapat dilihat bahwa *optimizer* memerlukan waktu yang lebih lama untuk melatih model, namun tidak berbeda secara signifikan. Jika menimbang hasil akurasi yang didapat dengan menggunakan *optimizer* dapat disimpulkan bahwa lebih baik menggunakan *optimizer* untuk melatih model.

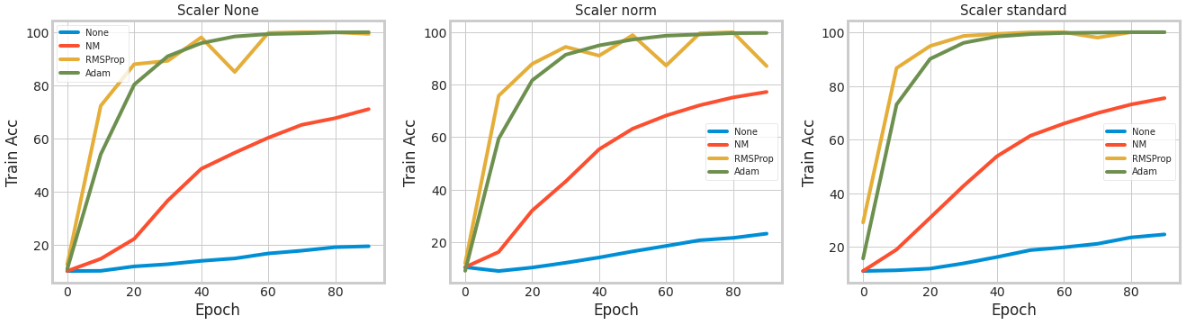
Selanjutnya akan dilihat pengaruh dari mengubah nilai learning rate pada model. Didapat hasil sebagai berikut



Dari grafik di atas dapat dilihat bahwa dengan menggunakan jumlah *epoch* yang sama, nilai akurasi dari NesterovMomentum dan tanpa *optimizer* bertambah seiring dengan bertambahnya nilai dair learning rate yang digunakan. Walau memang NesterovMomentum lebih cepat menuju kekonvergenan jika dibandingkan dengan tanpa *optimizer*. Ini menandakan bahwa NesterovMomentum perlu menggunakan nilai learning rate yang besar agar cepat menuju kekonvergenan.

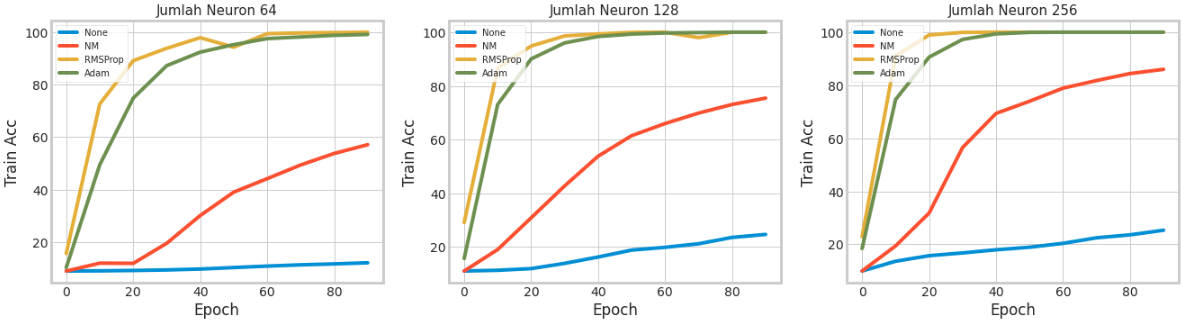
Jika kita tinjau hasil dari RMSProp dan Adam dapat dilihat bahwa performa dari *optimizer* tersebut mengalami penurunan seiring bertambahnya nilai learning rate yang digunakan, dimana bahkan pada learning rate 0.5 model sama sekali tidak mengalami peningkatan akurasi. Ini menandakan bahwa untuk kedua *optimizer* tersebut diperlukan nilai learning rate yang kecil.

Selanjutnya akan dilihat pengaruh dari melakukan *scaling* terhadap data yang digunakan untuk modelling. Didapat hasil sebagai berikut



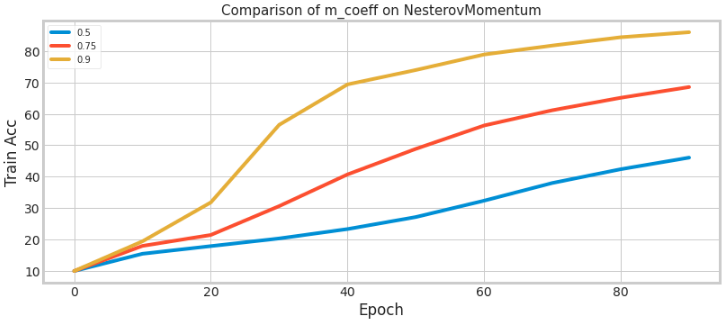
Dari grafik di atas dapat dilihat bahwa **Normalisasi** membuat model yang menggunakan NesterovMomentum dan tanpa *optimizer* bertambah akurasinya jika dibandingkan dengan tanpa melakukan *scaling*. Walau begitu model yang menggunakan Adam terlihat tidak ada perbedaan yang signifikan dan model yang menggunakan RMSProp menjadi tidak stabil. Hal berbeda terlihat pada model yang datanya di*scale* menggunakan **Standarisasi**, dapat dilihat bahwa semua jenis *optimizer* menjadi lebih cepat menuju kekonvergenan terutama model yang menggunakan RMSProp dimana perubahan akurasinya menjadi lebih stabil. Dari sini dapat disimpulkan bahwa penggunaan Standarisasi secara umum dapat memberikan pengaruh yang baik pada permodelan ANN sehingga untuk eksperimen selanjutnya penulis akan menggunakan **Standarisasi**

Selanjutnya akan dilihat pengaruh dari mengubah jumlah neuron pada layer ANN. Didapat hasil sebagai berikut



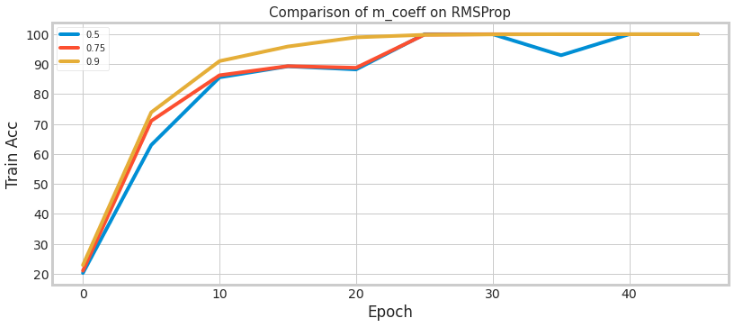
Dari grafik di atas dapat dilihat bahwa semakin besar jumlah neuron yang digunakan maka semakin cepat juga model menuju kekonvergenan. Dapat dilihat bahwa pada model dengan jumlah neuron 256, model yang menggunakan RMSProp lebih cepat menuju kekonvergenan dibandingkan dengan model Adam. Selain itu model NesterovMomentum dapat mencapai nilai akurasi sebesar 86%. Dari sini dapat disimpulkan bahwa penambahan neuron dapat membuat model menjadi lebih bagus lagi sehingga untuk eksperimen selanjutnya akan digunakan jumlah neuron 256

*Optimizer* NesterovMomentummenggunakan parameter bernama **m\_coeff** untuk mengatur sebarapa besar pengaruh nilai *gradient descent* sebelumnya. Selanjutnya akan dilihat pengaruh dari mengubah nilai parameter **m\_coeff**. Didapat hasil sebagai berikut



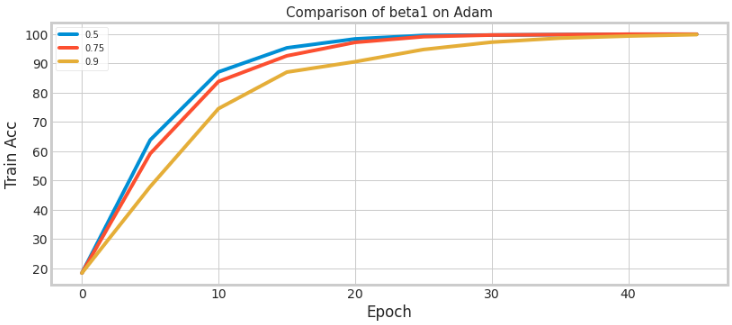
Dari grafik di atas dapat dilihat bahwa semakin besar nilai dari parameter **m\_coeff** semakin besar juga nilai akurasi yang didapat. Dari sini dapat disimpulkan bahwa lebih baik menggunkana nilai parameter **m\_coeff** yang besar tapi perlu dilihat juga kestabilan perubahan nilai akurasinya

*Optimizer* RMSProp jugamenggunakan parameter bernama **m\_coeff** untuk mengatur sebarapa besar pengaruh nilai *gradient descent* sebelumnya. Selanjutnya akan dilihat pengaruh dari mengubah nilai parameter **m\_coeff**. Didapat hasil sebagai berikut dengan menggunakan jumlah epoch sebesar 50



Dari grafik di atas dapat dilihat bahwa semakin besar nilai dari parameter **m\_coeff** semakin besar juga nilai akurasi yang didapat, walau perbedaannya tidak sesignifikan pada NesterovMomentum. Dari sini dapat disimpulkan bahwa lebih baik menggunkana nilai parameter **m\_coeff** yang besar tapi perlu dilihat juga kestabilan perubahan nilai akurasinya

*Optimizer* Adam menggunakan parameter **beta1** dan **beta2** untuk mengatur seberapa besar pengaruh momen pertama dan kedua pada *optimizer* tersebut. Selanjutnya akan dilihat pengaruh dari mengubah nilai parameter **beta1** dan **beta2**. Didapat hasil sebagai berikut



Berbeda dengan hasil sebelumnya, dari grafik di atas dapat dilihat bahwa nilai **beta1** yang kecil membuat model lebih cepat menuju kekonvergenan. Tapi hal ini tidak berlaku pada **beta2** dimana nilai yang besar membuat model lebih cepat menuju kekonvergenan

1. **Kesimpulan**

Setelah melakukan berbagai macam eksperimen didapat kesimpulan bahwa penggunaan *optimizer* dapat mempercepat model menuju kekonvergenan. Namun untuk tiap *optimizer* diperlukan juga parameter model yang tepat agar *optimizer* tersebut dapat bekerja dengan optimal. Secara umum Standarisasi dan menambah jumlah neuron dapat membuat model menjadi lebih cepat menuju kekonvergenan. Pemilihan learning rate harus disesuaikan dengan *optimizer* yang digunakan. NesterovMomentum diperlukan nilai learning rate yang besar dimana pada eksperimen didapat bahwa learning rate 0.5 paling optimal untuk *optimizer* ini. RMSProp dan Adam memerlukan nilai learning rate yang kecil dimana pada eksperimen dilihat bahwa nilai learning rate yang besar membuat model menjadi tidak belajar. Learning rate 0.1 cocok untuk kedua *optimizer* tersebut

Selain paramter model, perlu juga ditinjau parameter dari *optimizer* itu sendiri. Dari eksperimen didapat bahwa nilai **m\_coeff** yang kecil dapat memberikan performa yang lebih baik untuk *optimizer* NesterovMomentum dan RMSProp. Selain itu didapat bahwa nilai **beta1** yang kecil dan nilai **beta2** yang besar cocok untuk *optimizer* Adam. Namum perlu diperhatikan juga kestabilan dari perubahan akurasi jika menggunakan nilai parameter yang terlalu kecil atau terlalu besar.