Optimisasi

Hendrik Santoso Sugiarto

CALVIN
INSTITUTE OF TECHNOLOGY
God's People for God's Glory

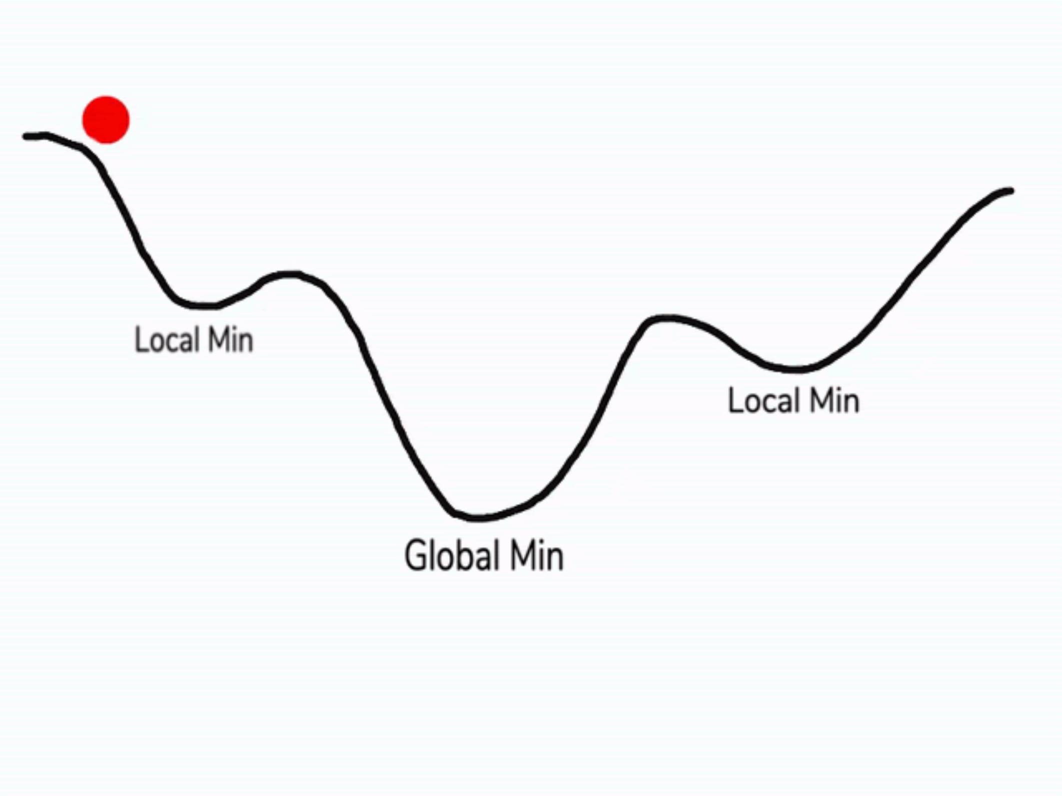
Capaian Pembelajaran

- Optimisasi
  - Menggunakan teknik-teknik optimisasi untuk memperoleh parameter optimal pada deep learning
- Optimisasi Ordo 2
  - Mengerti teknik-teknik optimisasi ordo ke 2
- GPU and Paralelisme
  - Menggunakan GPU dan komputasi parallel untuk proses operasi tensor
- Hyperparameter Tuning
  - Konsep hyperparameter dan cara optimisasi hyperparameter

Optimisasi

Optimisasi

- Proses belajar terjadi melalui proses optimisasi parameter (w) terhadap fungsi objektif (J) → model diharapkan menjadi semakin cerdas seiring dengan semakin optimalnya parameter tersebut
- Optimisasi Deep Learning sulit karena kompleksitas permodelan yang menghasilkan banyak minima lokal



Mini-Batch Learning

- Optimisasi terhadap seluruh training data dalam tiap iterasi disebut juga dengan metode deterministik atau batch
- Optimisasi terhadap 1 data dalam tiap iterasi disebut juga dengan metode stokastik atau online
- Optimisasi deep learning biasanya berada diantara 2 metode ini, yaitu menggunakan beberapa data tiap iterasi yang disebut juga dengan mini-batch

CALVIN INSTITUTE OF TECHNOLOGY

Kelebihan mini-batch

- Minibatch digunakan karena beberapa faktor berikut:
  - Limitasi memori jika menggunakan semua data
  - Estimasi gradien yang lebih akurat dengan beberapa data sekaligus
  - Batch bisa diproses secara parallel
  - Penggunaan perangkat GPU akan teroptimalkan jika menggunakan ukuran mini-batch dengan kelipatan pangkat 2 seperti 32 dan 256
  - Efek regularisasi karena noise selama proses belajar

Uji Pemahaman

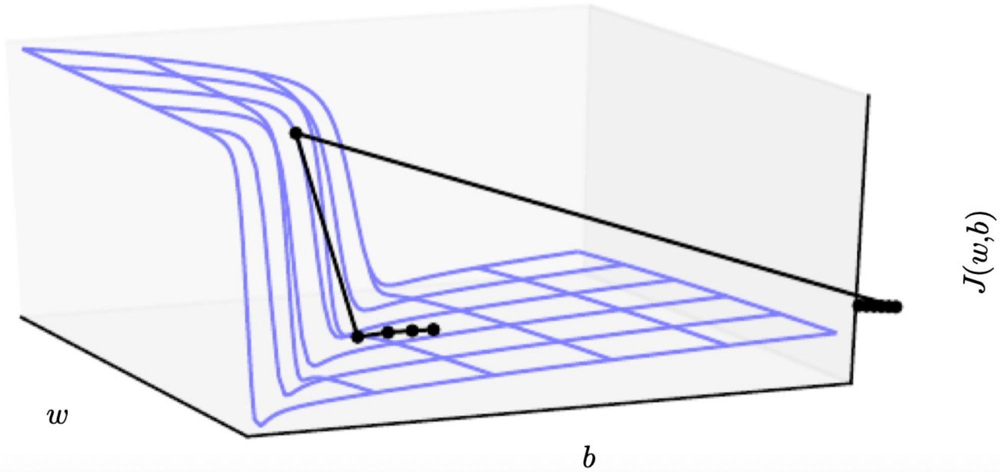
- Imagenet dataset memiliki 1,281,167 training data. Jika menggunakan batch size 512, berapa mini-batch yang dibutuhkan untuk bisa cover seluruh training data?

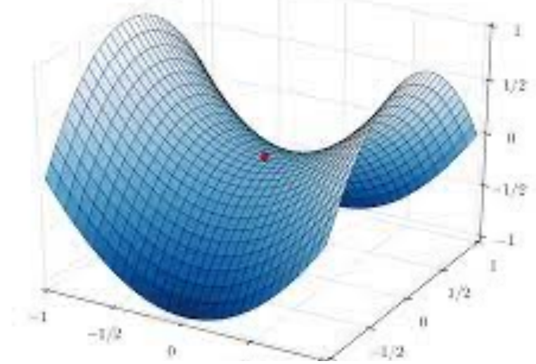
Aktivitas: Mini-Batch di PyTorch

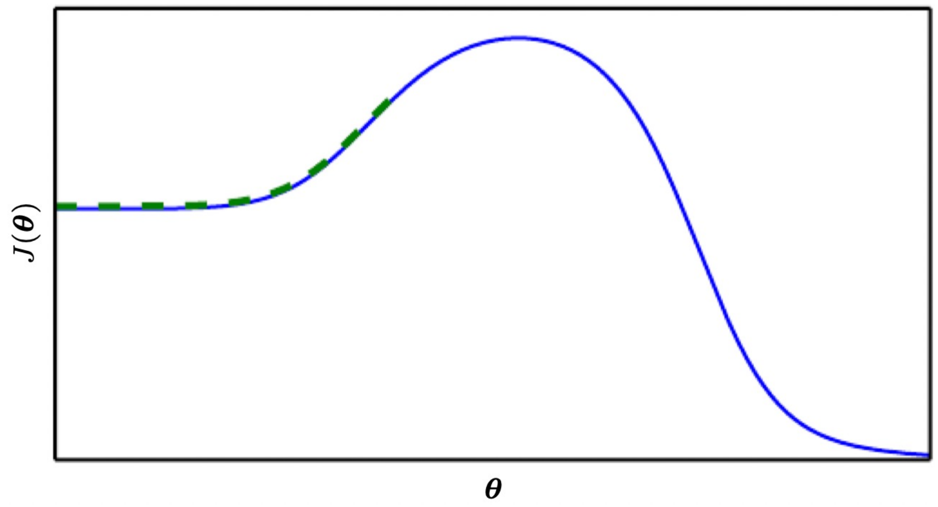
- Siapkan data berbentuk tensor mini-batch melalui dataloader
- Latih beberapa epoch gradient descent
    - 1 epoch berarti iterasi seluruh mini-batch

Problem dalam pembelajaran

- Pengkondisian buruk pada tensor gradien
- Minima local
- Saddle points
- Jurang dan ledakan gradien
- Overflow dan Underflow (karena kedalaman dependensi model)
- Gradien yang tidak eksak
- Tidak ada korespondensi struktur local dan global
- Limit teoretis







CALVIN INSTITUTE OF TECHNOLOGY

Algoritma Gradient Descent

- $\theta \leftarrow \theta - \epsilon \nabla_{\theta} J$
- Variasi:
  - Stochastic Gradient Descent
  - Momentum
  - Momentum Nesterov
  - AdaGrad
  - RMSProp
  - Adam
  - Etc

SGD (Stochastic Gradient Descent)

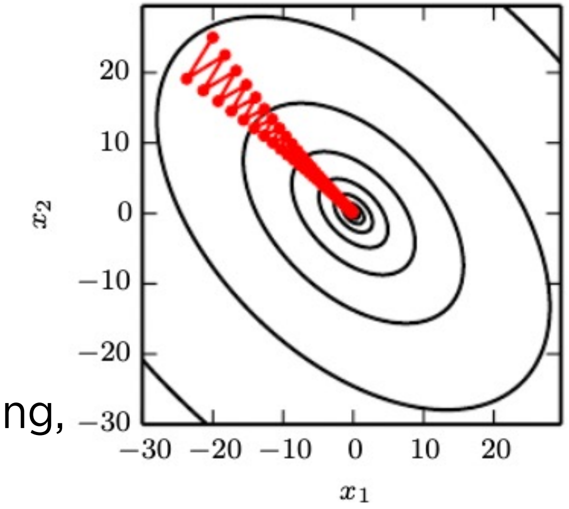
Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

---

Require: Learning rate ϵ_k.
Require: Initial parameter $\boldsymbol{\theta}$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$
  end while

Deep learning, p294

- Gradient Descent
  - $\theta \leftarrow \theta - \epsilon \nabla_\theta J$
- Excess error:
  - $O\left(\frac{1}{\sqrt{k}}\right)$
- Hyperparameter krusial:
  - ϵ (learning rate)



Deep learning, p89

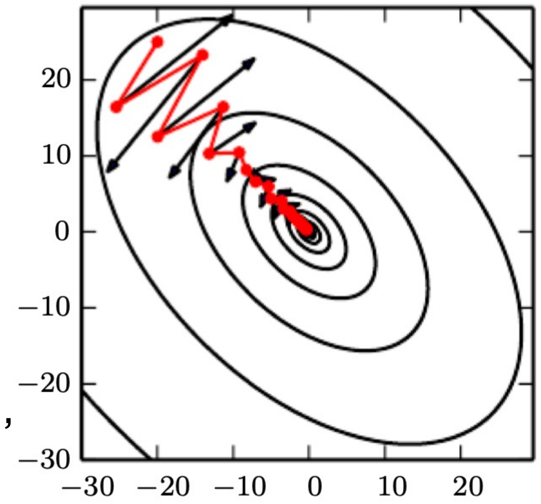
Momentum

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while

Deep learning, p298

- Analogi fisika
  - $g = \nabla_\theta J(\theta) = \frac{\partial v}{\partial t}$
  - $v = \frac{\partial \theta}{\partial t}$
- Excess error:
  - $O(\log(k))$
- Hyperparameter krusial:
  - ϵ (learning rate),
  - α (friction)



Deep learning, p297

Momentum Nesterov

Algorithm 8.3 Stochastic gradient descent (SGD) with Nesterov momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with corresponding labels $\boldsymbol{y}^{(i)}$.
    Apply interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
    Compute gradient (at interim point): $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while

Deep learning, p300

- Koreksi terhadap gradien:
  - Gradien dihitung setelah memperbaharui bobot
- Excess error:
  - $O\left(\frac{1}{k^2}\right)$
- Hyperparameter krusial:
  - ϵ (learning rate),
  - α (friction)

CALVIN INSTITUTE OF TECHNOLOGY

AdaGrad

Algorithm 8.4 The AdaGrad algorithm

Require: Global learning rate ϵ
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, perhaps 10^{-7}, for numerical stability
Initialize gradient accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. (Division and square root applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

Deep learning, p308

- Learning rate berubah sesuai akumulasi gradien
  - Parameter dengan gradien besar berkurang banyak
  - Parameter dengan gradien kecil berkurang sedikit
- Excess error:
  - $O(?)$
- Hyperparameter krusial:
  - ϵ (learning rate),
  - δ (scaling)

CALVIN INSTITUTE OF TECHNOLOGY

RMSprop

Algorithm 8.5 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, usually 10^{-6}, used to stabilize division by small numbers.
  Initialize accumulation variables $\boldsymbol{r} = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$.   ($\frac{1}{\sqrt{\delta+\boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

Deep learning, p309

- Modifikasi AdaGrad:
  - Akumulasi gradien → weighted average
- Excess error:
  - $O(?)$
- Hyperparameter krusial:
  - ϵ (learning rate),
  - δ (scaling)
  - ρ (decay rate)

CALVIN INSTITUTE OF TECHNOLOGY

Adam (adaptive moments)

Algorithm 8.7 The Adam algorithm

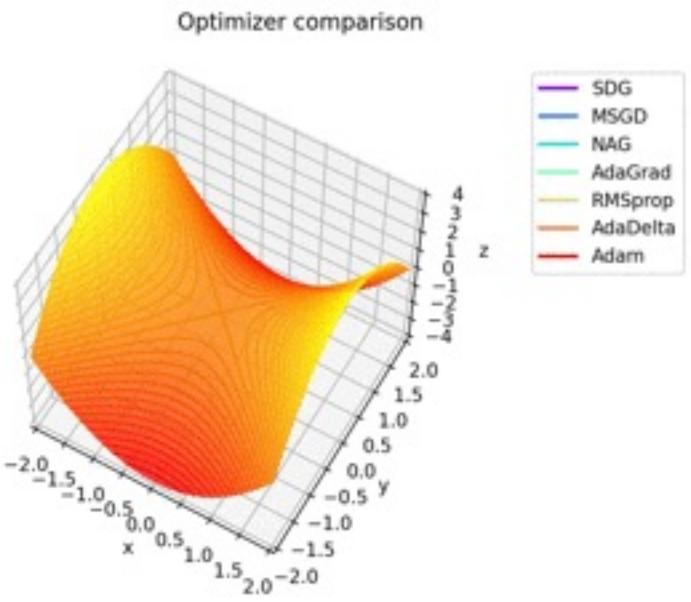
Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \mathbf{0}$, $\boldsymbol{r} = \mathbf{0}$
  Initialize time step $t = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
    Compute update: $\Delta \boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$
  end while

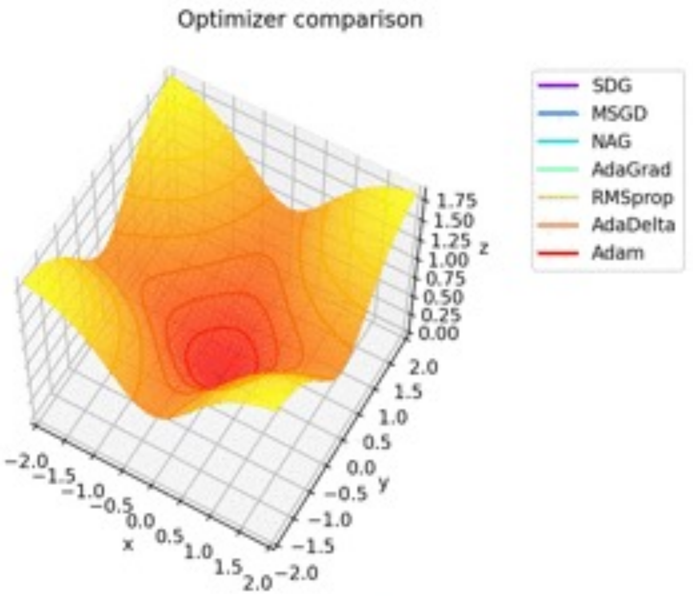
Deep learning, p311

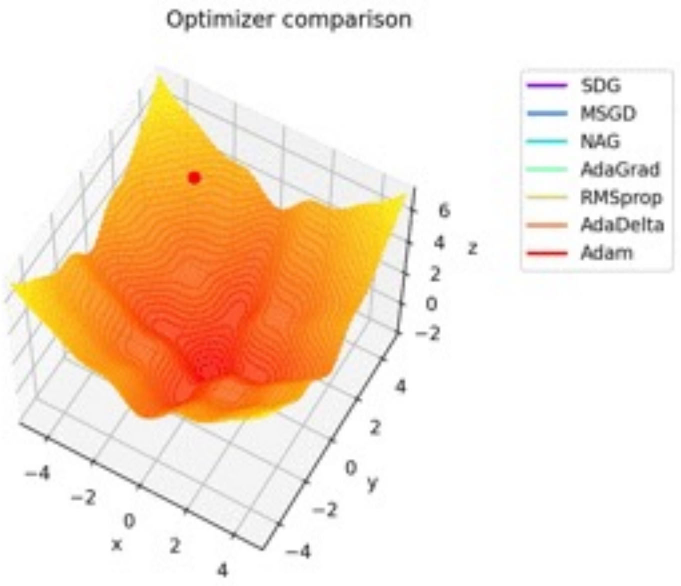
- Menambah momentum pada RMSprop
  - 1^{st} order moment: momentum
  - 2^{nd} order moment: akumulasi gradien
- Excess error:
  - $O(?)$
- Hyperparameter krusial:
  - ϵ (learning rate),
  - δ (scaling),
  - $\rho_1 \& \rho_2$ (exponential decay rate)

Perbandingan variasi optimisasi 1st order

- https://linuxtut.com/en/6695e0c79e888543e150/







CALVIN INSTITUTE OF TECHNOLOGY

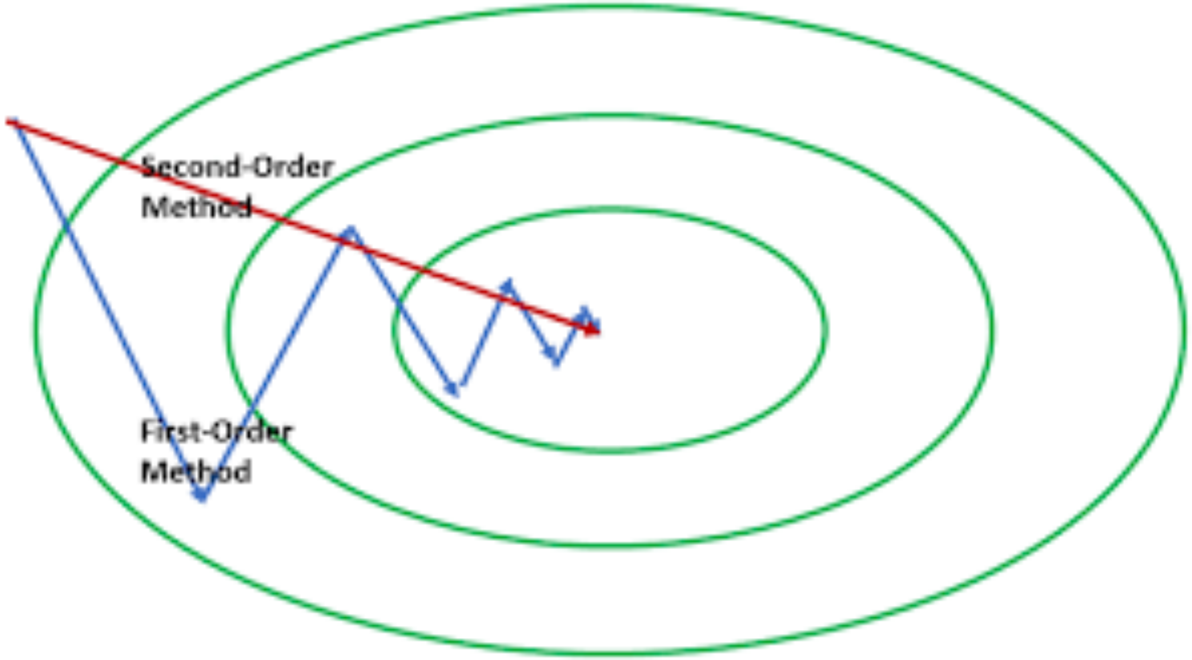
Aktivitas: Variasi Optimisasi di PyTorch

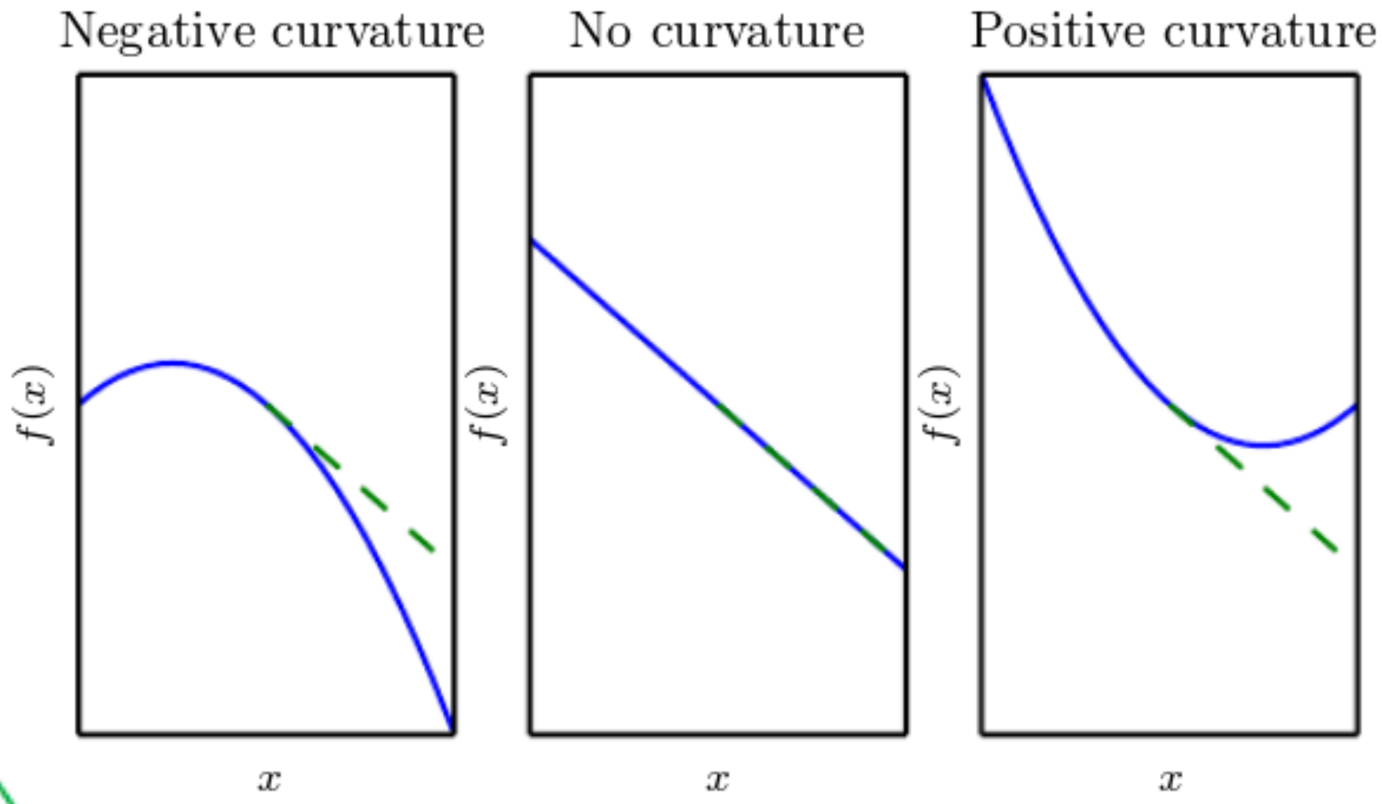
- Bandingkan penurunan nilai loss dari optimisasi SGD, RMSprop, AdaGrad, dan Adam di Pytorch

Optimisasi Ordo 2

Optimisasi 2nd Order

- Metode Newton
- Metode konjugat
- BFGS





Metode Newton

- Gunakan ekspansi Taylor disekitar titik $\boldsymbol{\theta}_0$

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Dimana $\boldsymbol{H}$ adalah matriks Hessian

- Setiap iterasi harus menghitung Hessian
- Kompleksitas komputasi $O(k^3)$

CALVIN INSTITUTE OF TECHNOLOGY

Metode Konjugat

- Mencari arah baru yang merupakan konjugat dari arah sebelumnya
- $\boldsymbol{d}_t$ dan $\boldsymbol{d}_{t-1}$ adalah konjugat jika $\boldsymbol{d}_t^T \mathbf{H} \boldsymbol{d}_{t-1} = 0$
- Trik untuk tidak perlu menghitung Hessian: $\boldsymbol{d}_t = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) + \beta_t \boldsymbol{d}_{t-1}$
  - Fletcher-Reeves:

$$\beta_t = \frac{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)}{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})}$$

  - Polak-Ribiere:

$$\beta_t = \frac{\left(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})\right)^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)}{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})^T \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{t-1})}$$

Deep Learning, p314

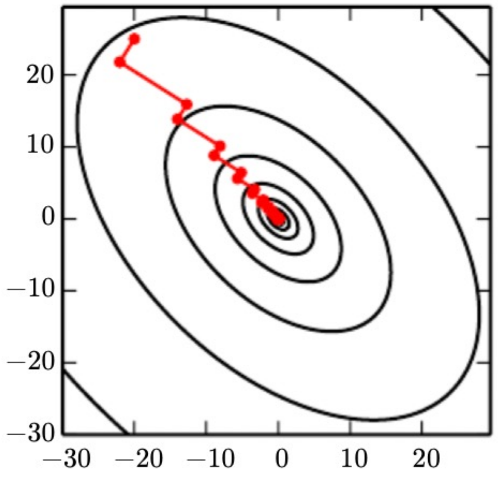


Figure 8.6: The method of steepest descent applied to a quadratic cost surface. The method of steepest descent involves jumping to the point of lowest cost along the line defined by the gradient at the initial point on each step. This resolves some of the problems seen with using a fixed learning rate in figure 4.6, but even with the optimal step size the algorithm still makes back-and-forth progress toward the optimum. By definition, at the minimum of the objective along a given direction, the gradient at the final point is orthogonal to that direction.

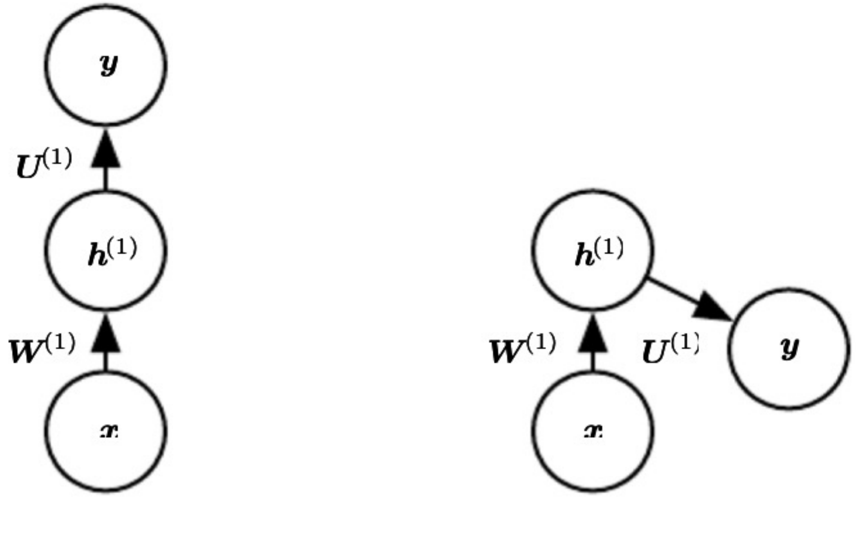
BFGS

- Broyden–Fletcher–Goldfarb–Shanno (BFGS) mencoba memanfaatkan keunggulan metode Newton tanpa membawa beban komputasinya
  - Memerlukan beban memori $O(n^2)$

- Limited Memory BFGS (L-BFGS) mengurangi beban memori dengan tidak menyimpan aproksimasi inverse Hessian
  - Memerlukan beban memori $O(n)$

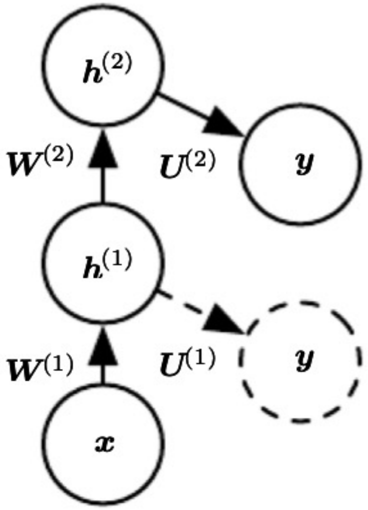
Meta-Algoritma

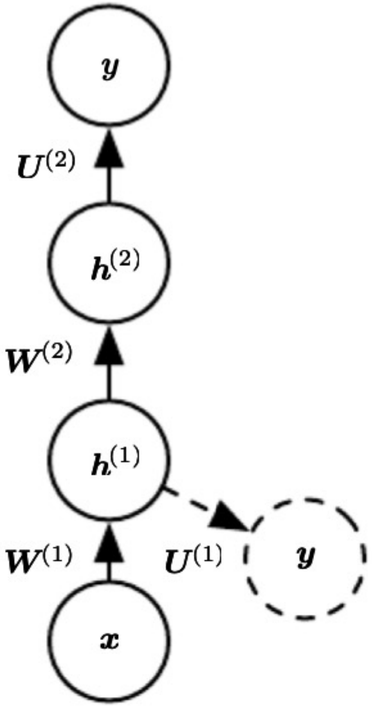
- Coordinate Descent
- Polyak Averaging
  - $\hat{\theta}^{(t)} = \alpha\hat{\theta}^{(t-1)} + (1-\alpha)\theta^{(t)}$
- Supervised Pretraining
- Continuation Methods
- Curriculum Learning



(a) (b)

Supervised Pretraining,
Deep Learning, p324





(c) (d)

CALVIN INSTITUTE OF TECHNOLOGY

Uji Pemahaman

- Manakah yang merupakan optimisasi dengan turunan ke 2?
  - AdaGrad
  - Adam
  - Newton
  - Polak-Ribiere

GPU & Paralelisme

God's People for God's Glory
CALVIN
INSTITUTE OF TECHNOLOGY

GPU dan parallelisme

- Graphic Processing Unit, awalnya untuk mempercepat proses gambar 3D
- Memiliki struktur parallel sehingga jauh lebih efisien dari CPU
    - 1 CPU memiliki 4 sampai 8 cores
    - 1 GPU memiliki ratusan cores yang lebih kecil
- Sekarang dipakai untuk mempercepat perhitungan tensor
    - Karena itu model dan data harus diletakan di perangkat GPU terlebih dahulu
    - Memori GPU terbatas, jadi harus dilakukan secara mini-batch
- Perhitungan paralel juga dapat dilakukan antar beberapa GPU

Aktivitas: GPU dan parallelisme di Pytorch

- Cek jumlah GPU yang tersedia, dan coba lakukan pelatihan paralel

Hyperparameter

God's People for God's Glory
CALVIN INSTITUTE OF TECHNOLOGY

Parameter vs Hyperparameter

	Parameter	Hyperparameter
Mempengaruhi prediksi	Ya	Ya
Bagian dari model	Internal	Eksternal
Ubah nilai	Selama proses belajar / optimisasi	Dituning selama proses validasi, namun tidak berubah selama proses belajar optimasi
Mekanisme mengubah nilai	Optimisasi (menggunakan algoritma belajar)	Tuning
Contoh	Bobot, bias	Learning rate, layers, nodes

Hyperparameter

Deep learning, p431

- Neural network mempunyai beberapa hyperparameter yang umum
- Perhatikan skala dan rentang hyperparameter
  - Linear: contoh $[0, 0.25, 0.5, 0.75, 1]$
  - Logaritmik: contoh $[0.1, 0.01, 0.001, 0.0001]$

Hyperparameter	Increases capacity when...	Reason	Caveats
Number of hidden units	increased	Increasing the number of hidden units increases the representational capacity of the model.	Increasing the number of hidden units increases both the time and memory cost of essentially every operation on the model.
Learning rate	tuned optimally	An improper learning rate, whether too high or too low, results in a model with low effective capacity due to optimization failure	
Convolution kernel width	increased	Increasing the kernel width increases the number of parameters in the model	A wider kernel results in a narrower output dimension, reducing model capacity unless you use implicit zero padding to reduce this effect. Wider kernels require more memory for parameter storage and increase runtime, but a narrower output reduces memory cost.
Implicit zero padding	increased	Adding implicit zeros before convolution keeps the representation size large	Increased time and memory cost of most operations.
Weight decay coefficient	decreased	Decreasing the weight decay coefficient frees the model parameters to become larger	
Dropout rate	decreased	Dropping units less often gives the units more opportunities to "conspire" with each other to fit the training set	

Table 11.1: The effect of various hyperparameters on model capacity.

CALVIN INSTITUTE OF TECHNOLOGY

Hyperparameter Tuning

- Gridsearch vs randomsearch

Deep learning,
p433

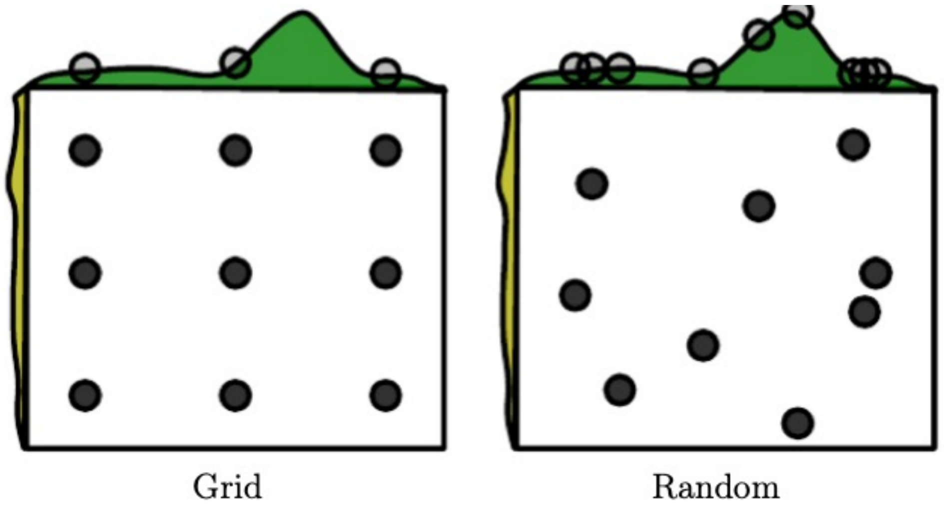
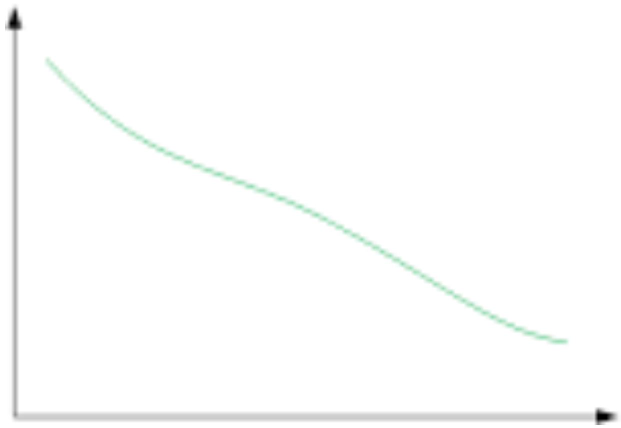


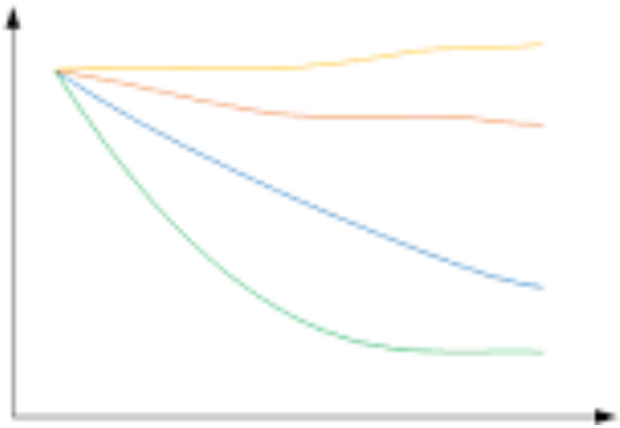
Figure 11.2: Comparison of grid search and random search. For illustration purposes we display two hyperparameters but we are typically interested in having many more. *(Left)*To perform grid search, we provide a set of values for each hyperparameter. The search algorithm runs training for every joint hyperparameter setting in the cross product of these sets. *(Right)*To perform random search, we provide a probability distribution over joint hyperparameter configurations. Usually most of these hyperparameters are independent from each other. Common choices for the distribution over a single hyperparameter include uniform and log-uniform (to sample from a log-uniform distribution, take the exp of a sample from a uniform distribution). The search algorithm then randomly samples joint hyperparameter configurations and runs training with each of them. Both grid search and random search evaluate the validation set error and return the best configuration. The figure illustrates the typical case where only some hyperparameters have a significant influence on the result. In this illustration, only the hyperparameter on the horizontal axis has a significant effect. Grid search wastes an amount of computation that is exponential in the number of non-influential hyperparameters, while random search tests a unique value of every influential hyperparameter on nearly every trial. Figure reproduced with permission from Bergstra and Bengio (2012).

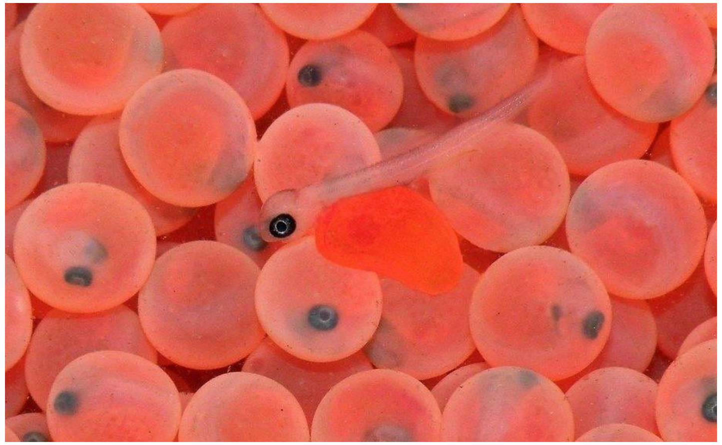
Panda vs Caviar

Babysitting one model



Training many models in parallel





Summary

- Masing-masing sebutkan konsep utama yang diingat

Tuhan Memberkati