TU-Delft Deep Learning course 2018-2019

05.optimization

6 Apr 2019

Delft University of Technology

Chapter 8

Lecturer: Jan van Gemert

Several slides inspired by Andrew Ng

Topic: Optimization

- Past gradient update statistics
- How to efficiently compute past statistics
- Momentum, RMSprop, Adam.
- Feature and Batch normalization

Book chapters: 8.3.1, 8.3.2, 8.5.2, 8.5.3, 8.7.1

Training a network

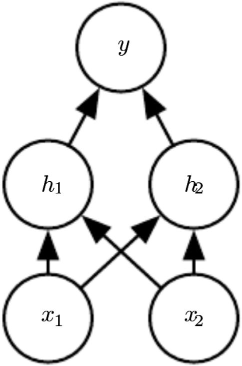
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

While not converged:

1. Present a training sample
2. Compare the results
3. Update the weights



Training a network

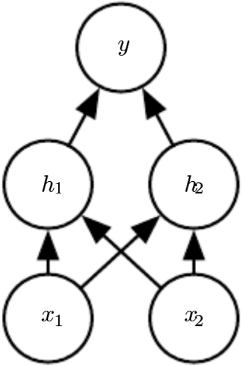
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

While not converged:

1. Present a training sample $\rightarrow$ Forward pass
2. Compare the results
3. Update the weights



Training a network

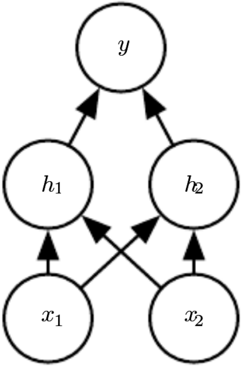
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

While not converged:

1. Present a training sample → Forward pass
2. Compare the results → Loss
3. Update the weights



Training a network

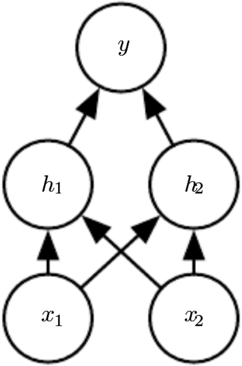
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

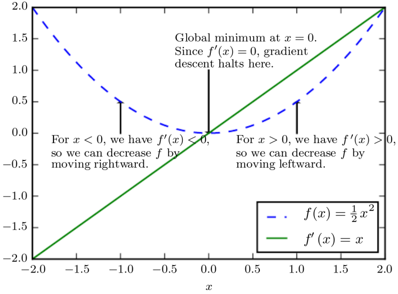
While not converged:

1. Present a training sample $\rightarrow$ Forward pass
2. Compare the results $\rightarrow$ Loss
3. Update the weights $\rightarrow$ Backward pass



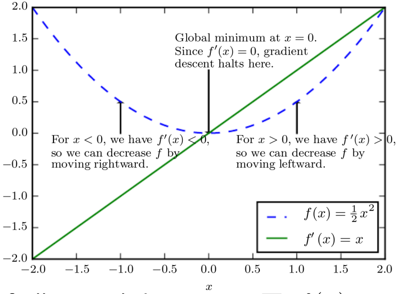
Stochastic Gradient Descent

Chapter 5.9 and 8.3.1



Stochastic Gradient Descent

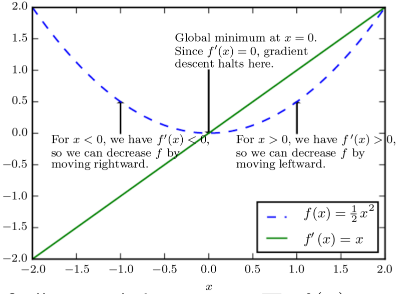
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$

Stochastic Gradient Descent

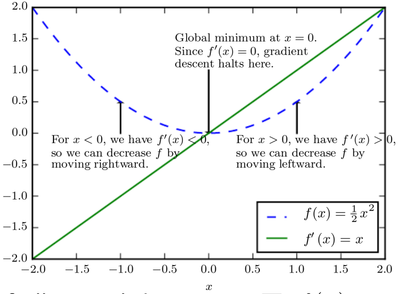
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta' = \theta - \epsilon \nabla_\theta f(x)$:

Stochastic Gradient Descent

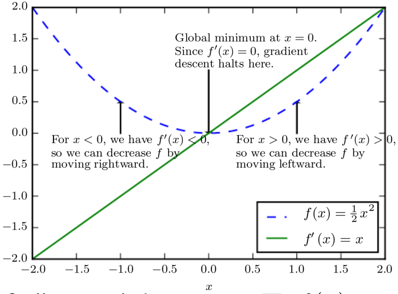
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta' = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate

Stochastic Gradient Descent

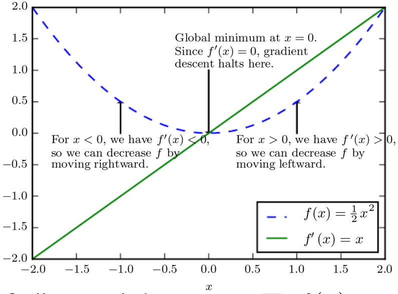
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta' = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$

Stochastic Gradient Descent

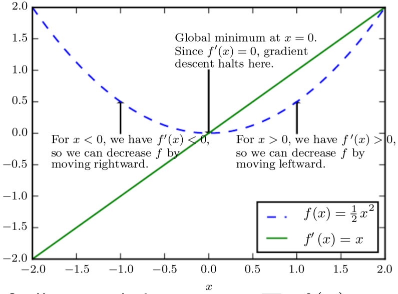
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta' = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$
- Gradient all samples: $\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$

Stochastic Gradient Descent

Chapter 5.9 and 8.3.1

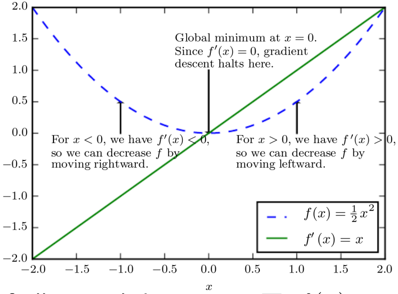


- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta' = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$
- Gradient all samples: $\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$

SGD approximates gradient from small sample set:

Stochastic Gradient Descent

Chapter 5.9 and 8.3.1



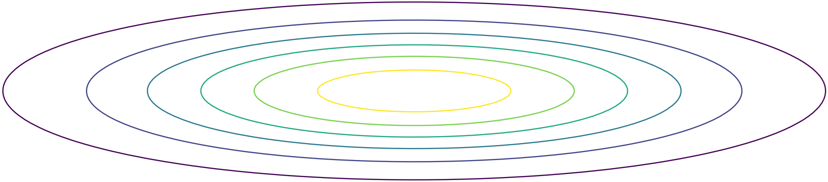
- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta' = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$
- Gradient all samples: $\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$

SGD approximates gradient from small sample set:

$$\theta' = \theta - \epsilon \frac{1}{m'} \sum_{i=1}^{m'} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$$

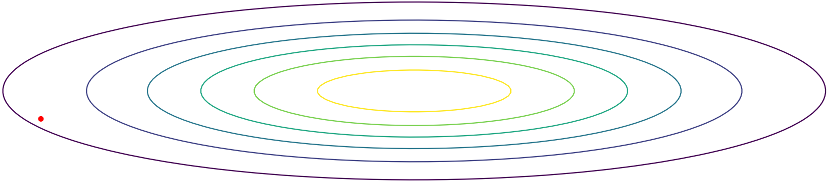
Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune



Tuning the learning rate; take 1.

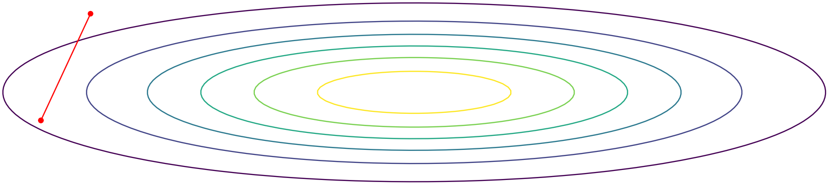
The learning rate is arguably the most important parameter to tune



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

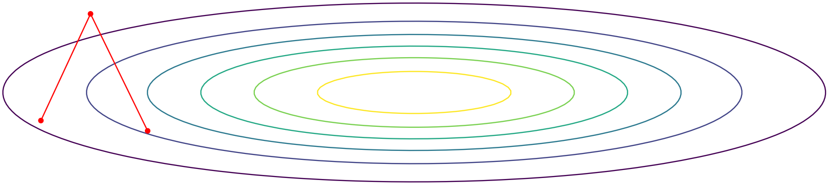
Q: Wait.. What is the ellipse? What are these red dots? Red line?



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

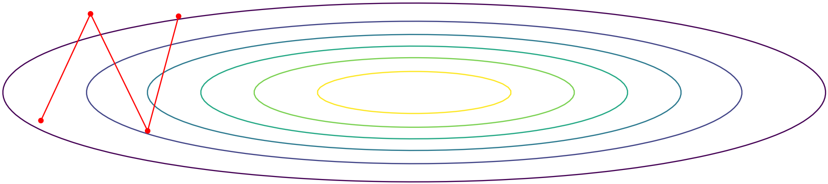
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

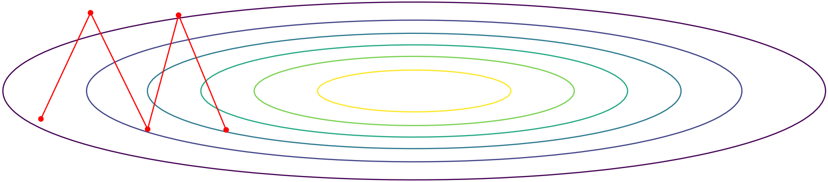
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

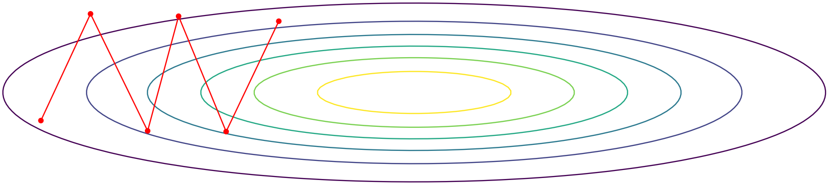
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

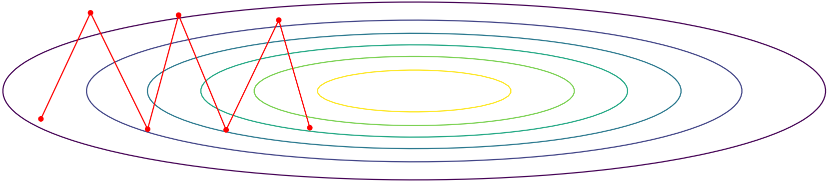
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

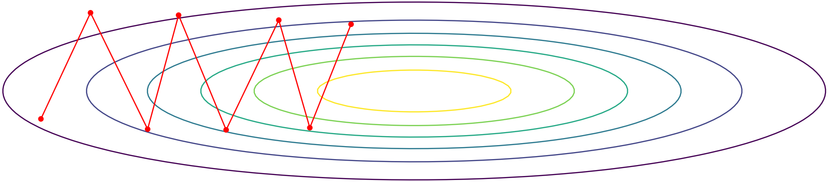
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

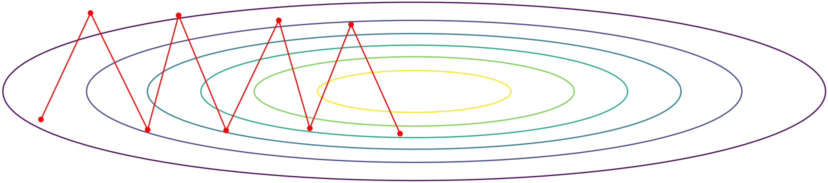
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.

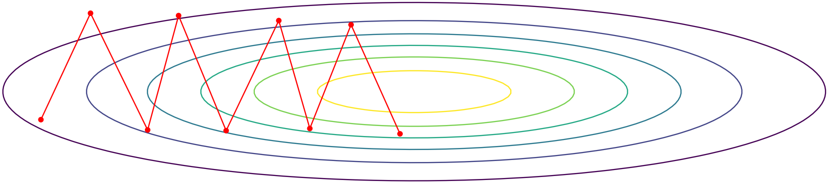


Q: What do you notice?

Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

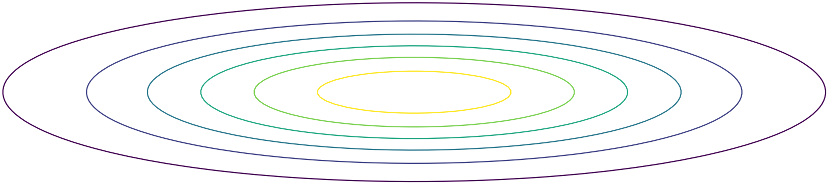
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Q: What do you notice?
A: If LR too high it will never find a good instantiation

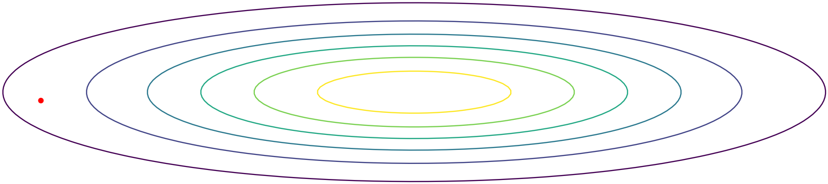
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



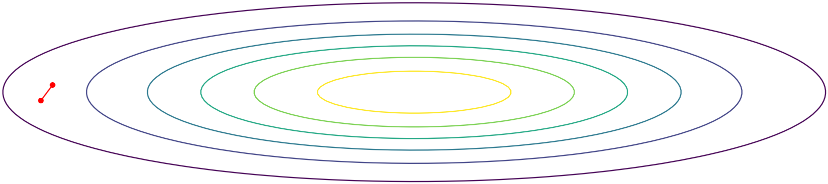
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



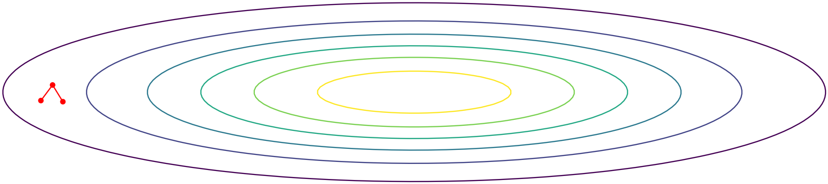
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



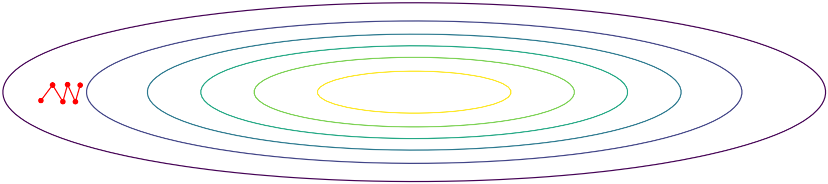
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



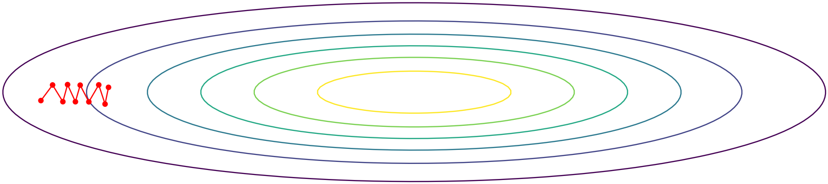
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



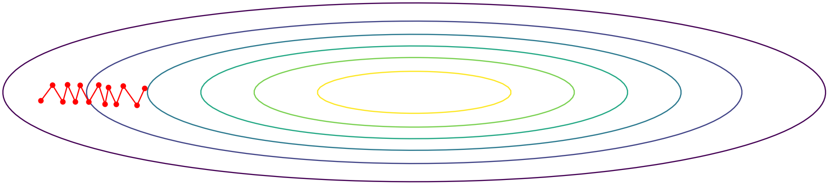
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



Tuning the learning rate; take 2.

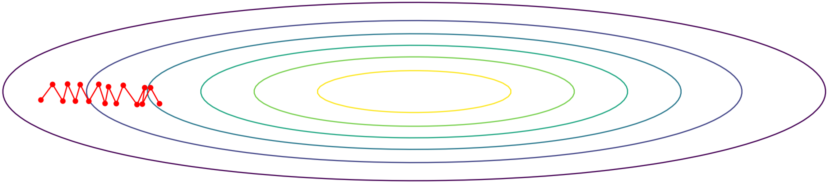
The learning rate is arguably the most important parameter to tune



Q: What do you notice?

Tuning the learning rate; take 2.

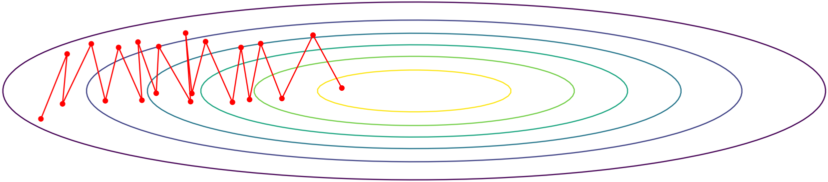
The learning rate is arguably the most important parameter to tune



A: If LR too low it will take very long to find a good instantiation

Tuning the learning rate; take 3.

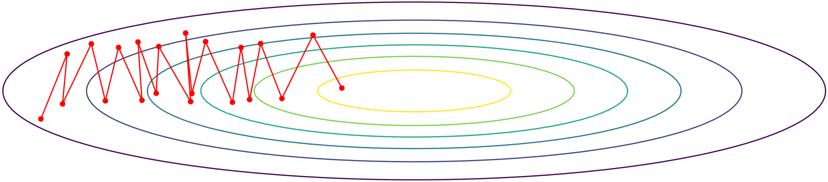
The learning rate is arguably the most important parameter to tune



Q: What do you notice?

Tuning the learning rate; take 3.

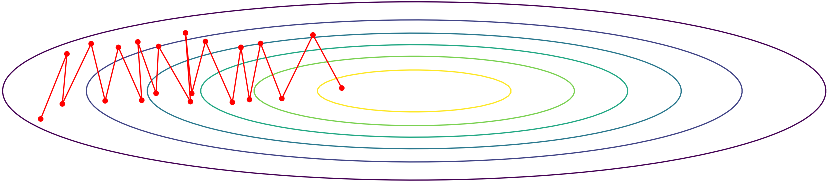
The learning rate is arguably the most important parameter to tune



Q: What do you notice? A: SGD is noisy; it's also noisy close to a good place.

Tuning the learning rate; take 3.

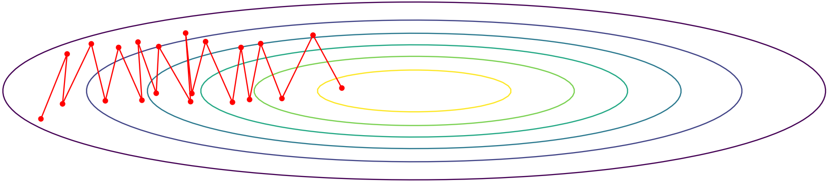
The learning rate is arguably the most important parameter to tune



Q: What do you notice? A: SGD is noisy; it's also noisy close to a good place.
Q: How to take smaller steps?

Tuning the learning rate; take 3.

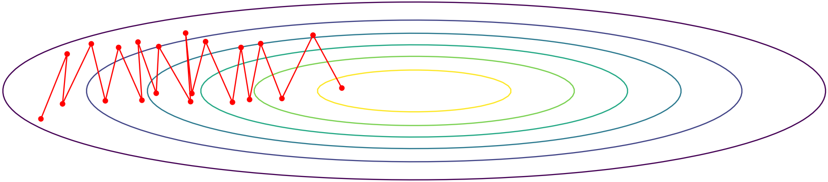
The learning rate is arguably the most important parameter to tune



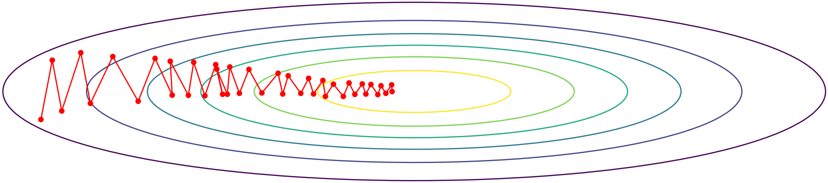
Q: What do you notice? A: SGD is noisy; it's also noisy close to a good place.
Q: How to take smaller steps? A: Reduce the LR over time (decay).

Tuning the learning rate; take 3.

The learning rate is arguably the most important parameter to tune



Q: What do you notice? A: SGD is noisy; it's also noisy close to a good place.
Q: How to take smaller steps? A: Reduce the LR over time (decay).



Learning rate: More art than science

Book, Chapter 8.3.1: "Most guidance should be regarded with some skepticism".

Learning rate: More art than science

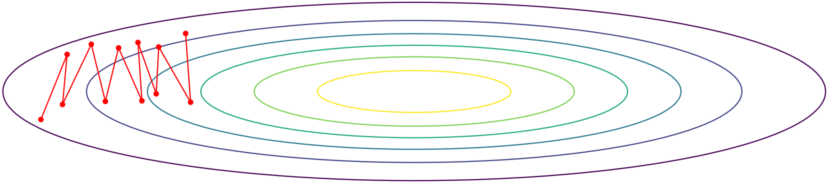
Book, Chapter 8.3.1: “Most guidance should be regarded with some skepticism”.

There are some algorithms that seem to generally improve over SGD

Learning rate: More art than science

Book, Chapter 8.3.1: "Most guidance should be regarded with some skepticism".

There are some algorithms that seem to generally improve over SGD

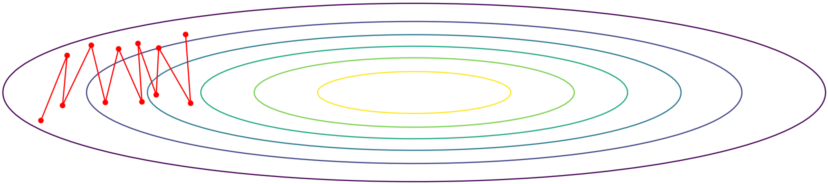


Q: Would you use the same learning rate for x-axis and y-axis parameters?

Learning rate: More art than science

Book, Chapter 8.3.1: "Most guidance should be regarded with some skepticism".

There are some algorithms that seem to generally improve over SGD



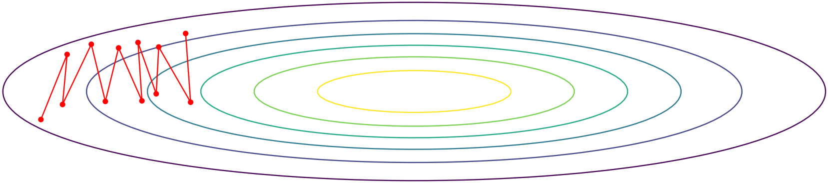
Q: Would you use the same learning rate for x-axis and y-axis parameters?

A: Use different learning rates per parameter: Smaller for y-axis parameter.

Learning rate: More art than science

Book, Chapter 8.3.1: "Most guidance should be regarded with some skepticism".

There are some algorithms that seem to generally improve over SGD



Q: Would you use the same learning rate for x-axis and y-axis parameters?

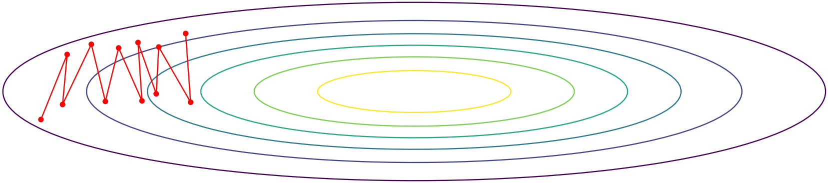
A: Use different learning rates per parameter: Smaller for y-axis parameter.

Q: What can you say of the average and of the variance for each dimension?

Learning rate: More art than science

Book, Chapter 8.3.1: "Most guidance should be regarded with some skepticism".

There are some algorithms that seem to generally improve over SGD



Q: Would you use the same learning rate for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

Q: What can you say of the average and of the variance for each dimension?
A: Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

Questions?

Questions?

Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

Questions?

Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

All these algorithms make use of past gradient update statistics.

Questions?

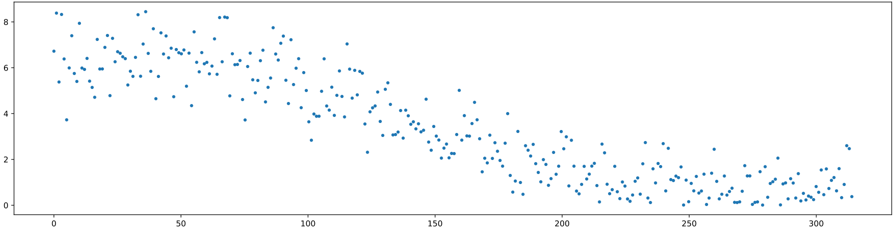
Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

All these algorithms make use of past gradient update statistics.

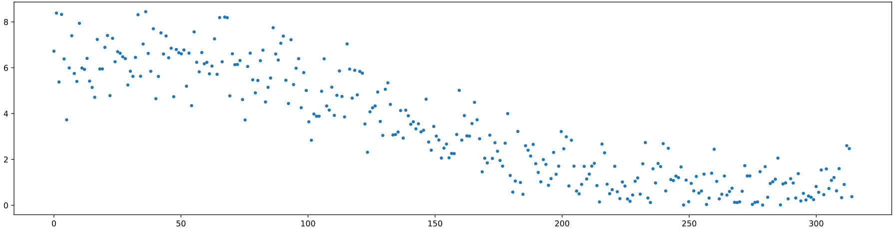
Next slides: How to efficiently keep track of such statistics.

Noisy time series



How to smooth out a noisy time series as values are coming in.

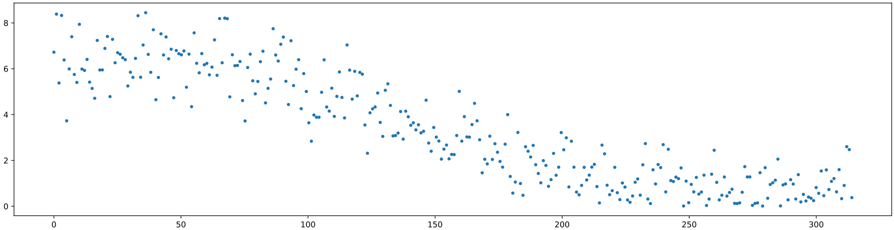
Noisy time series



How to smooth out a noisy time series as values are coming in.

- Q: Can we use convolution?

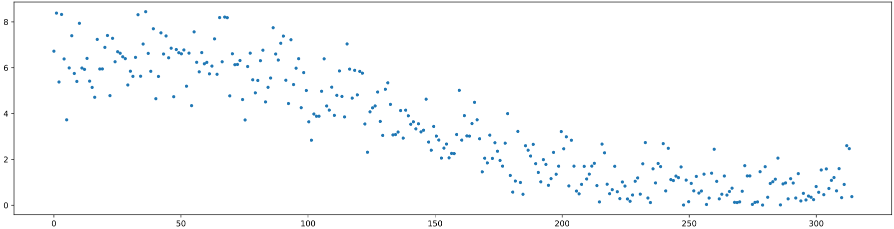
Noisy time series



How to smooth out a noisy time series as values are coming in.

- Q: Can we use convolution? A: No access to future values.
- Q: Keep recent values, and compute a weighted average?

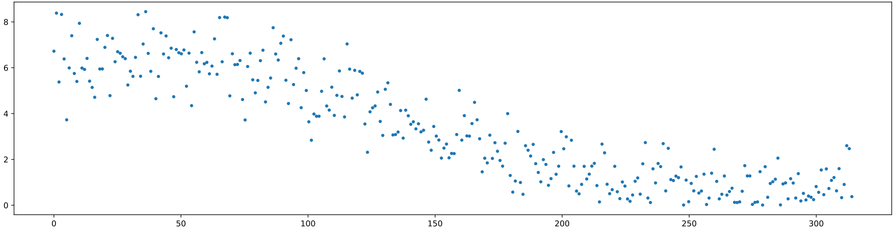
Noisy time series



How to smooth out a noisy time series as values are coming in.

- Q: Can we use convolution? A: No access to future values.
- Q: Keep recent values, and compute a weighted average?
- A: Out of memory (huge high-dimensional time series)

Noisy time series



How to smooth out a noisy time series as values are coming in.

- Q: Can we use convolution? A: No access to future values.
- Q: Keep recent values, and compute a weighted average?
- A: Out of memory (huge high-dimensional time series)

An infinite impulse response filter (IIR) to the rescue:

Recursively compute online average.

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0,$ if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ?

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0, \text{if } t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho\, S_{98} && + (1-\rho) y_{99}) + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho\, (\rho S_{97} + (1-\rho) y_{98}) && + (1-\rho) y_{99}) + (1-\rho) y_{100}
\end{aligned}
$$

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0,$ if $t = 1.$
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} &&+ (1-\rho)y_{100} \\
S_{100} &= \rho \quad (\rho\ S_{98} \quad +(1-\rho)y_{99}) &&+ (1-\rho)y_{100} \\
S_{100} &= \rho \quad (\rho\ (\rho S_{97} + (1-\rho)y_{98}) \quad +(1-\rho)y_{99}) &&+ (1-\rho)y_{100}
\end{aligned}
$$

Re-order as an exponentially weighted sum:

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1-\rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho\ S_{98} \quad +(1-\rho) y_{99}) && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho\ (\rho S_{97} + (1-\rho) y_{98}) \quad +(1-\rho) y_{99}) && + (1-\rho) y_{100}
\end{aligned}$$

Re-order as an exponentially weighted sum:
$S_{100} = (1-\rho)\ y_{100} + (1-\rho)\ \rho\ y_{99} + (1-\rho)\ \rho^2\ y_{98} + (1-\rho)\ \rho^3\ y_{97} + \ldots$

Exponentially weighted moving average (EWMA)

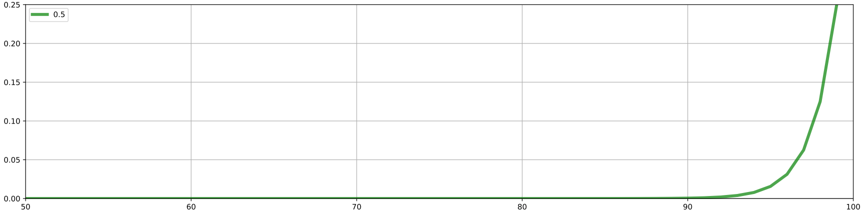
For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$S_{100} = \rho \quad S_{99} \quad + (1-\rho) y_{100}$$
$$S_{100} = \rho \quad (\rho\ S_{98} \quad + (1-\rho) y_{99}) + (1-\rho) y_{100}$$
$$S_{100} = \rho \quad (\rho\ (\rho S_{97} + (1-\rho) y_{98}) \quad + (1-\rho) y_{99}) + (1-\rho) y_{100}$$

Re-order as an exponentially weighted sum:

$$S_{100} = (1-\rho)\ y_{100} + (1-\rho)\ \rho\ y_{99} + (1-\rho)\ \rho^2\ y_{98} + (1-\rho)\ \rho^3\ y_{97} + \ldots$$



Example of the weights for $\rho = 0.5$

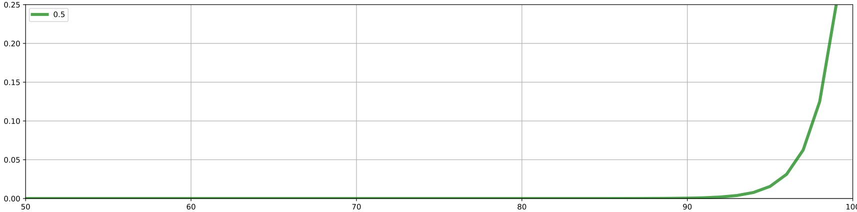
Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 \leq \rho \leq 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1-\rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

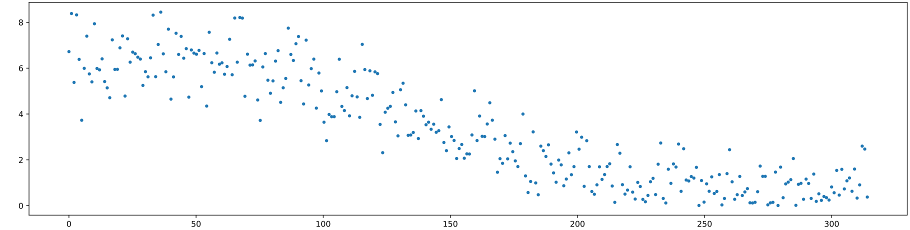
$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; S_{98} \quad + (1-\rho) y_{99}) && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; (\rho S_{97} + (1-\rho) y_{98}) \quad + (1-\rho) y_{99}) && + (1-\rho) y_{100}
\end{aligned}
$$

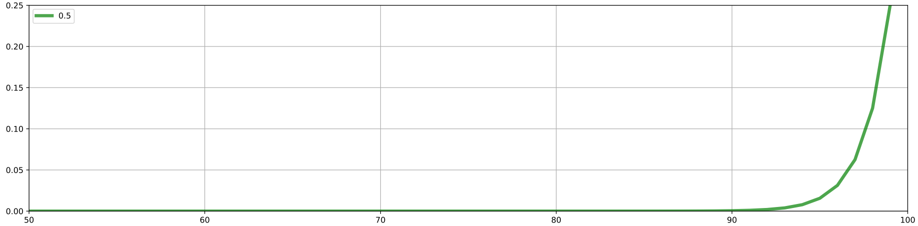
Re-order as an exponentially weighted sum:
$S_{100} = (1-\rho) \; y_{100} + (1-\rho) \; \rho \; y_{99} + (1-\rho) \; \rho^2 \; y_{98} + (1-\rho) \; \rho^3 \; y_{97} + \ldots$



Example of the weights for $\rho = 0.5$
Q: How about $\rho = 0.9$? $\rho = 0.95$?

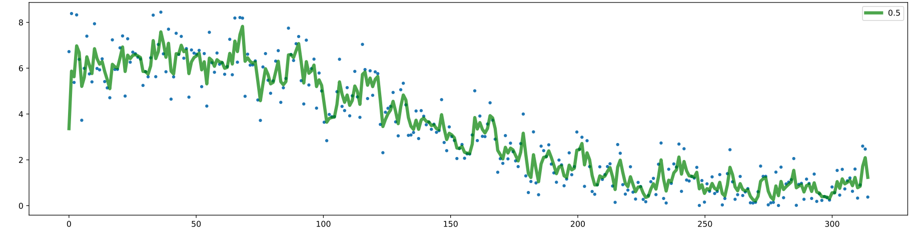
Exponentially weighted moving average (EWMA)

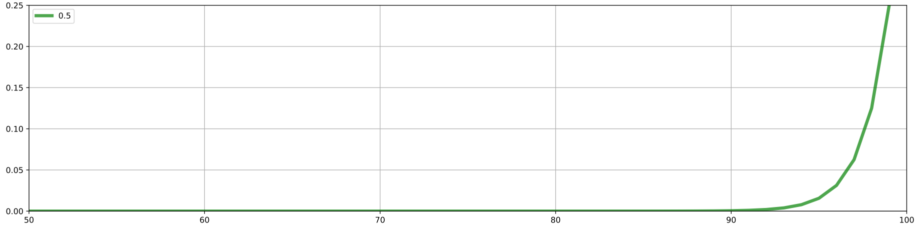




Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

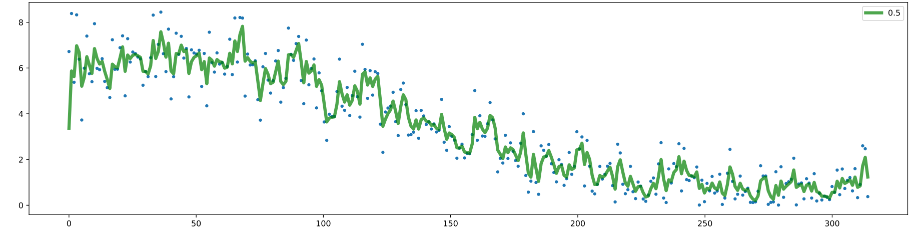
Exponentially weighted moving average (EWMA)

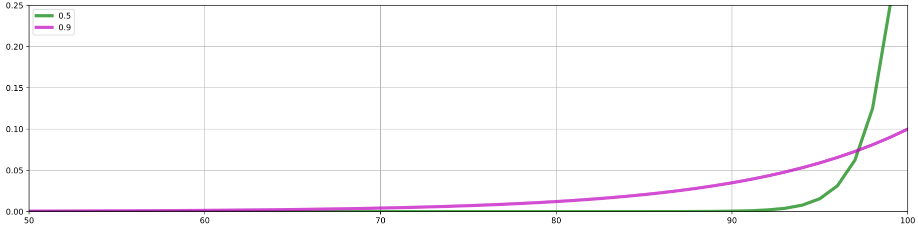




Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

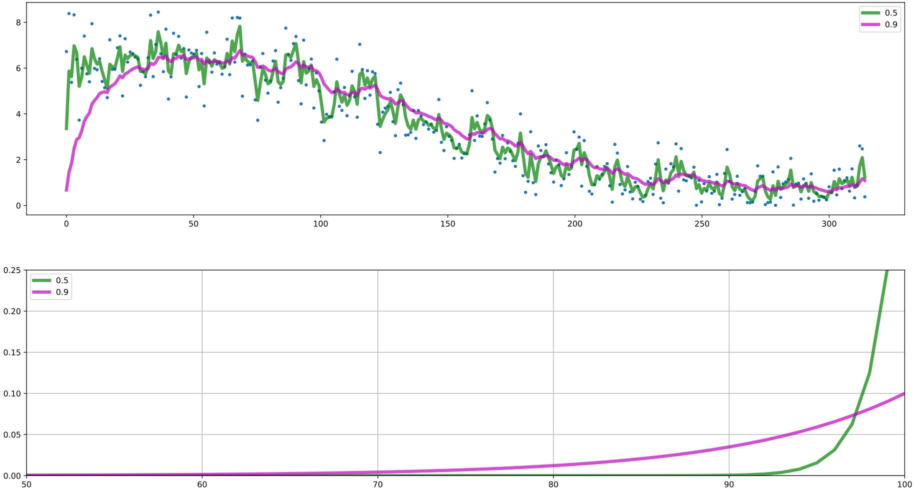
Exponentially weighted moving average (EWMA)





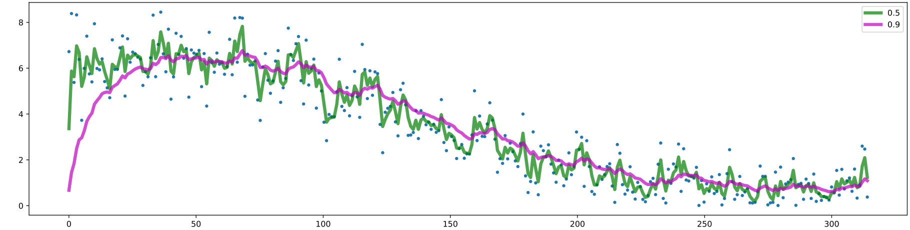
Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

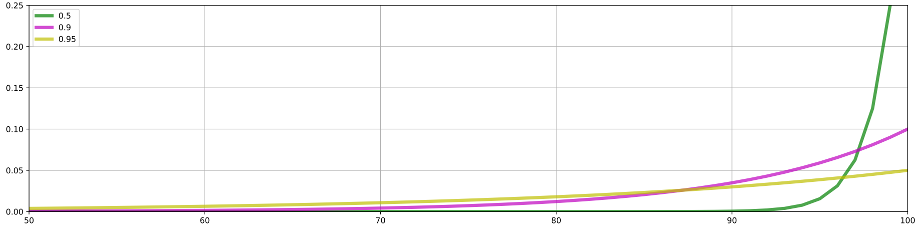
Exponentially weighted moving average (EWMA)



Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

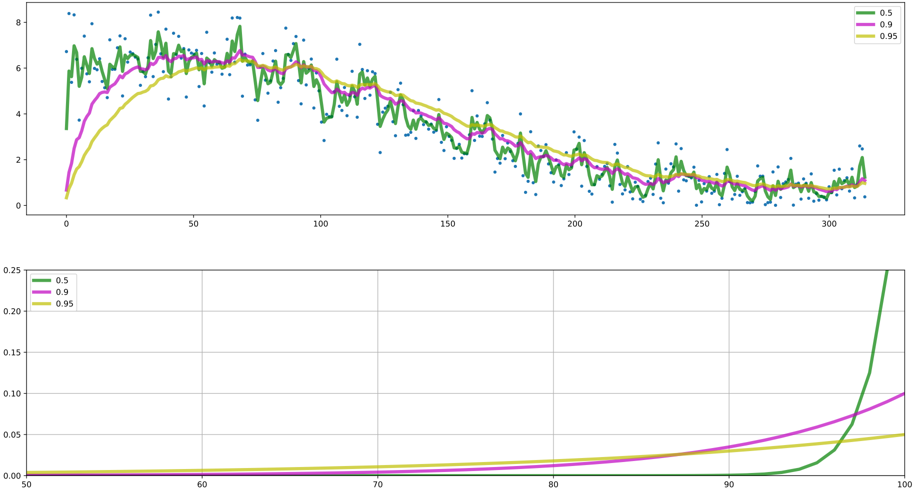
Exponentially weighted moving average (EWMA)





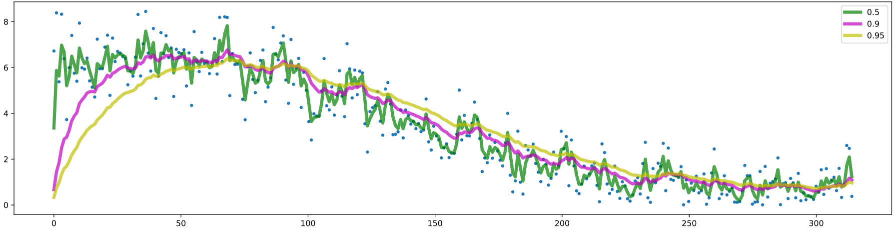
Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

Exponentially weighted moving average (EWMA)



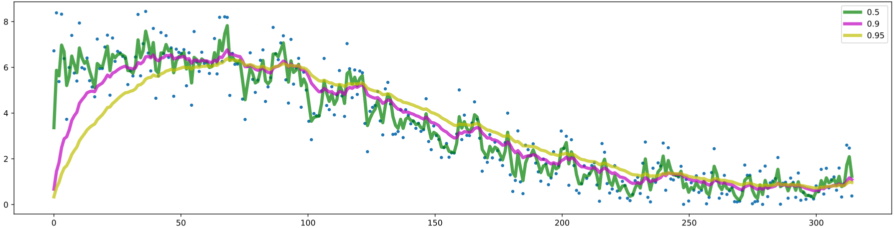
Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

Bias



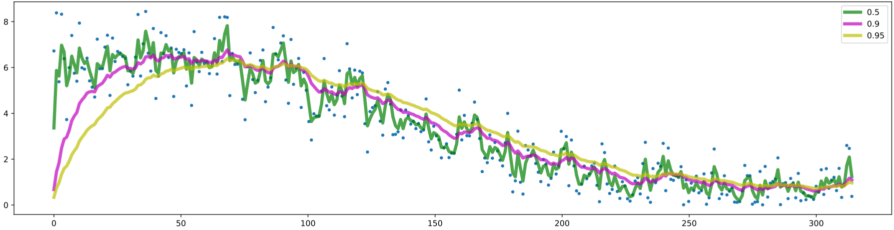
- Q: What do you notice about these curves?

Bias



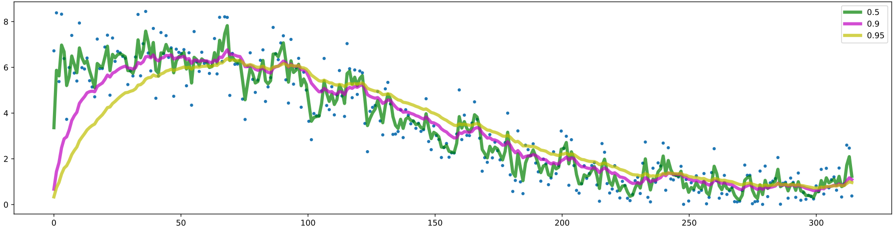
- Q: What do you notice about these curves?
- A: The larger ρ, the later it 'catches up'.

Bias



- Q: What do you notice about these curves?
- A: The larger ρ, the later it 'catches up'.
- Q: Why is that?

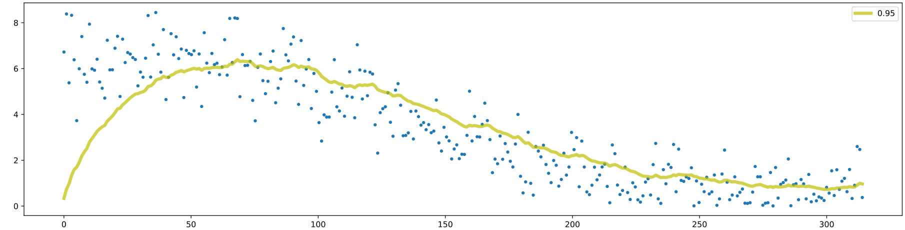
Bias



- Q: What do you notice about these curves?
- A: The larger ρ, the later it 'catches up'.
- Q: Why is that?
- A: Border effect: Unknown what happened before time.

Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

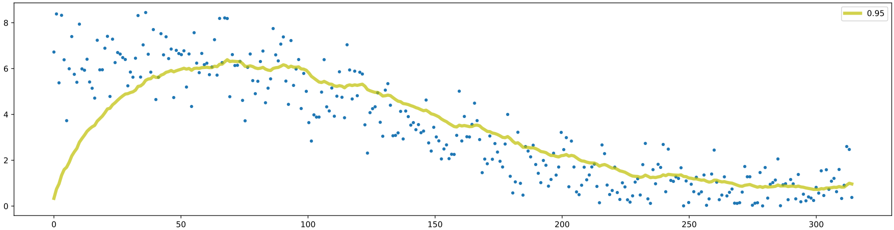
Lets start an example at S_1 for $\rho = 0.95$



Bias correction for $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

Lets start an example at S_1 for $\rho = 0.95$

- $S_1 = \rho \ S_0 + (1 - \rho) y_1 = 0 + 0.05 y_1$
- $S_2 = \rho \ S_1 + (1 - \rho) y_2 = 0.95 \times 0.05 y_1 + 0.05 y_2 = 0.0475 y_1 + 0.05 y_2$

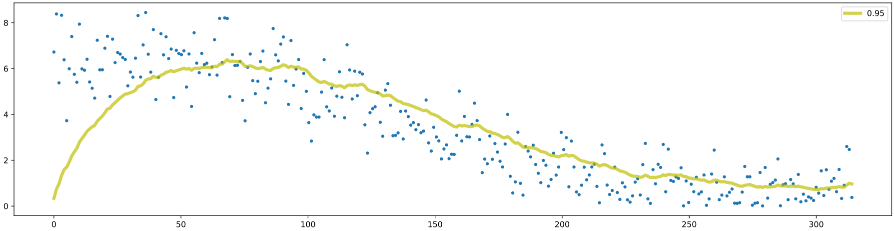


Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho) y_t$

Lets start an example at S_1 for $\rho = 0.95$

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.



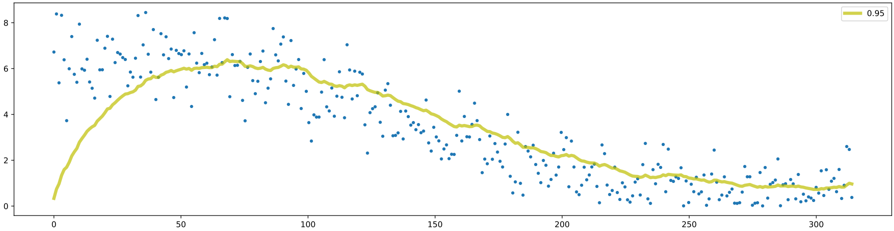
Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Lets start an example at S_1 for $\rho = 0.95$

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

Bias correction: $\hat{S}_t = \frac{S_t}{1-\rho^t}$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Lets start an example at S_1 for $\rho = 0.95$

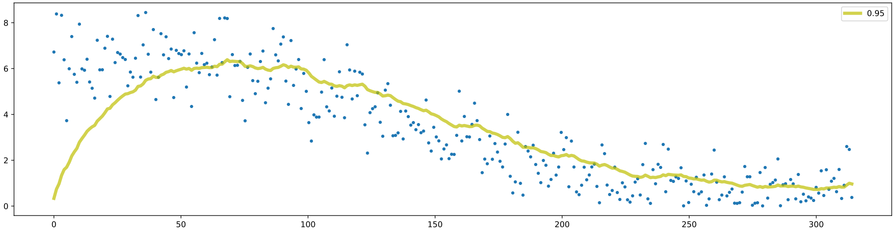
- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

Bias correction: $\hat{S}_t = \frac{S_t}{1-\rho^t}$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice?



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho) y_t$

Lets start an example at S_1 for $\rho = 0.95$

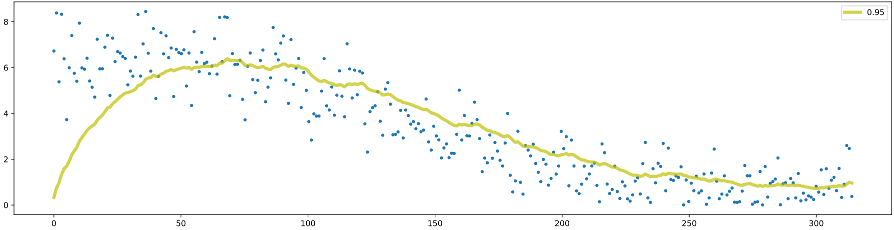
- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

Bias correction: $\hat{S}_t = \frac{S_t}{1-\rho^t}$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Lets start an example at S_1 for $\rho = 0.95$

- $S_1 = \rho \ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho \ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

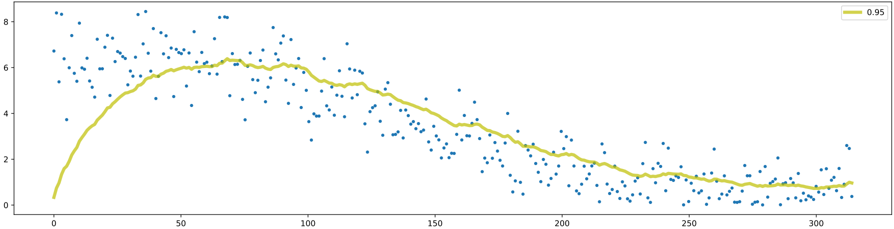
Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens for large t ?



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Lets start an example at S_1 for $\rho = 0.95$

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

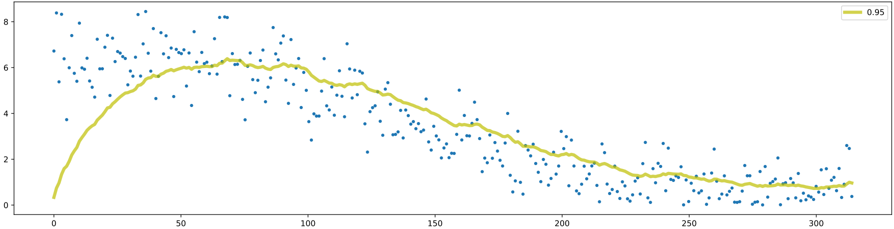
Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens for large t ? $1 - \rho^t \approx 1$, so $\hat{S}_t \approx S_t$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Lets start an example at S_1 for $\rho = 0.95$

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

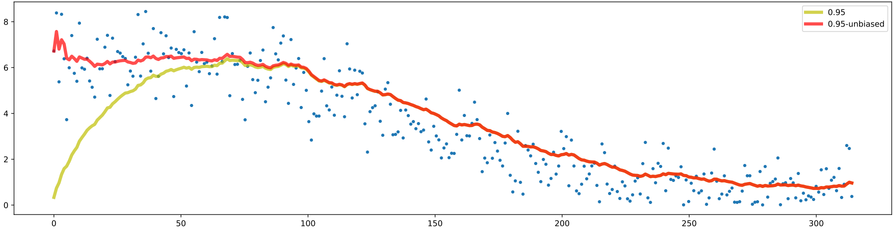
Indeed, very low values at the beginning.

Bias correction: $\hat{S}_t = \frac{S_t}{1-\rho^t}$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens for large t ? $1 - \rho^t \approx 1$, so $\hat{S}_t \approx S_t$



Questions?

Questions?

Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

All these algorithms make use of past gradient update statistics.

Questions?

Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

All these algorithms make use of past gradient update statistics.

Now can efficiently keep track of such statistics.

Questions?

Useful algorithms:

- Momentum: On average in the good direction
- RMSprop: Variance in the wrong direction is high
- Adam: Combines momentum and RMSprop

All these algorithms make use of past gradient update statistics.

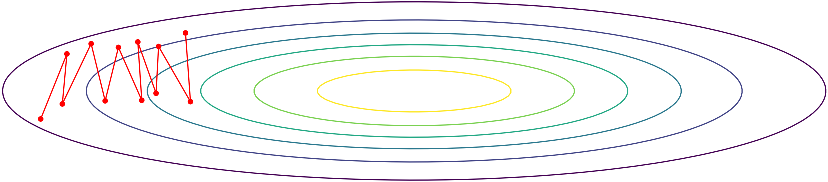
Now can efficiently keep track of such statistics.

Next slides: lets use them for gradient updates.

Stochastic Gradient Descent with momentum

Chapter 8.3.2

Momentum smooths the average of noisy gradients with EWMA



Hyper-parameters: EWMA: ρ; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

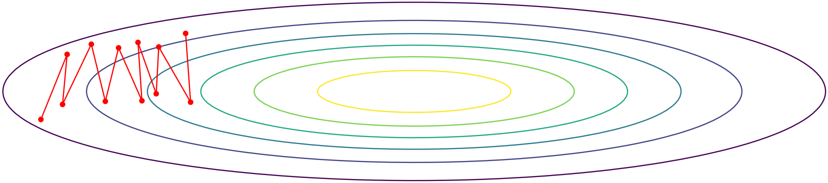
- $v_i = \rho v_{i-1} + (1-\rho)\nabla_\theta$
- $\theta' = \theta - \epsilon v_i$

Sometimes (book) written as $v_i = \rho v_{i-1} + \nabla_\theta$, where ϵ then needs to be re-tuned.
Often implemented without bias correction, as it does catch up quick.
Default setting $\rho = 0.9$

Stochastic Gradient Descent with RMSprop

Chapter 8.5.2

RMSprop smooths the zero-centered variance of noisy gradients with EWMA



Hyper-parameters: EWMA: ρ; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

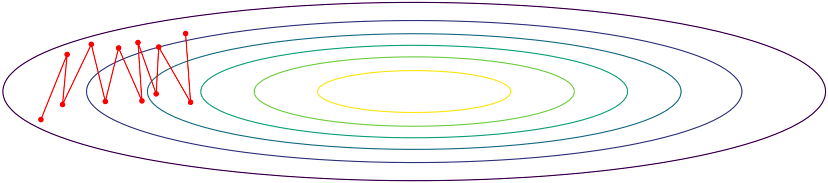
- $r_i = \rho r_{i-1} + (1-\rho)\nabla_\theta^2$
- $\theta' = \theta - \epsilon \frac{\nabla_\theta}{\sqrt{r_i}}$

To prevent dividing by 0, add a small $\delta \approx 10^{-6}$ to denominator: $r_i = \delta + r_i$
Often implemented without bias correction, as it does catch-up quick.
Default setting $\rho = 0.9$

Stochastic Gradient Descent with Adam

Chapter 8.5.3

Adam combines momentum with RMSprop



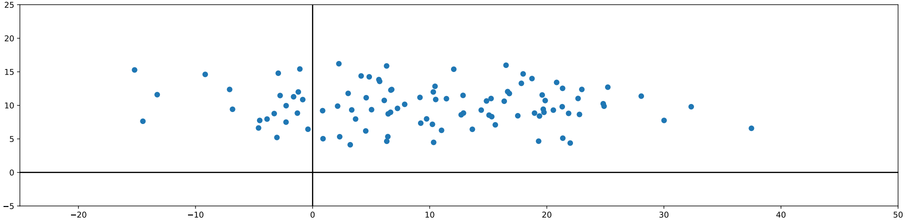
Hyper-parameters: EWMA: ρ_1, ρ_2; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

- $v_i = \rho_1 v_{i-1} + (1-\rho_1)\nabla_\theta,$ $\qquad r_i = \rho_2 r_{i-1} + (1-\rho_2)\nabla_\theta^2$
- $\hat{v}_i = \frac{v_i}{(1-\rho_1^i)},$ $\qquad \hat{r}_i = \frac{r_i}{(1-\rho_2^i)}$
- $\theta' = \theta - \epsilon \frac{\hat{v}_i}{\sqrt{\hat{r}_i}}$

To prevent dividing by 0, add a small $\delta \approx 10^{-6}$ to denominator: $r_i = \delta + r_i$
Default settings: $\rho_1 = 0.9$ and $\rho_2 = 0.999$

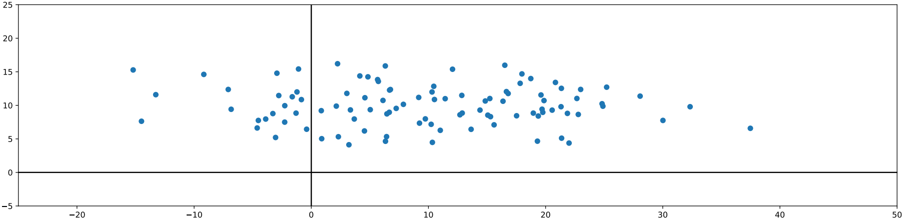
Questions?

Effect of feature normalization on optimization



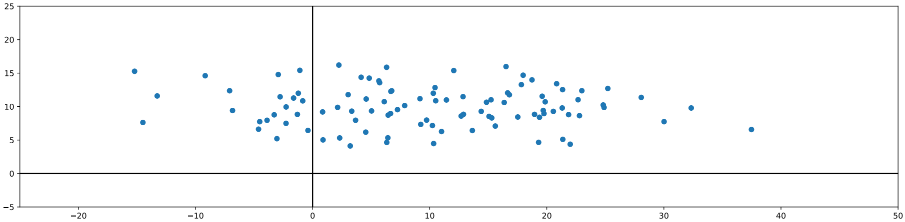
- Centering inputs at zero and equal dimension variance speeds up training

Effect of feature normalization on optimization



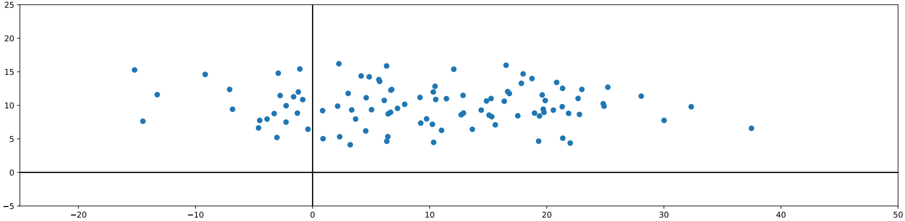
- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle?

Effect of feature normalization on optimization



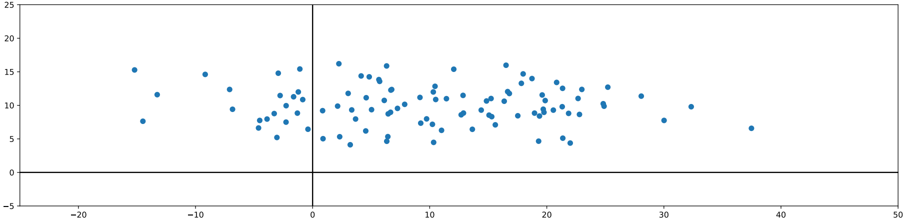
- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle? A:
- $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$,

Effect of feature normalization on optimization



- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle? A:
- $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$,
- $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$,

Effect of feature normalization on optimization

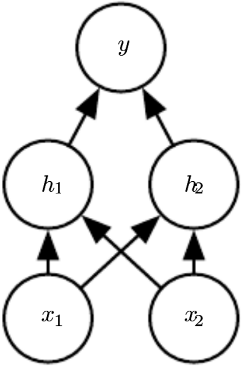


- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle? A:
- $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$,
- $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$,
- $X = \frac{X-\mu}{\sqrt{\sigma^2}}$

Batch norm: Normalizing learned features

Chapter 8.7.1

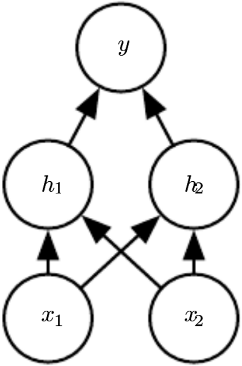
- A deep neural net learns many hidden feature representations.



Batch norm: Normalizing learned features

Chapter 8.7.1

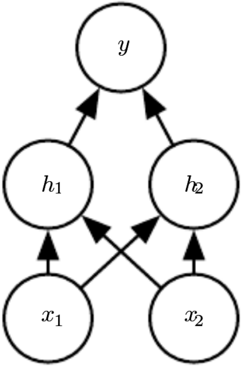
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?



Batch norm: Normalizing learned features

Chapter 8.7.1

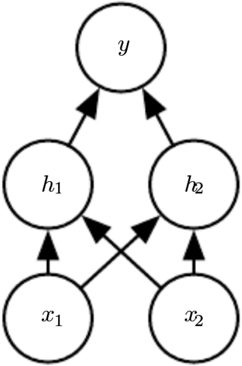
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.



Batch norm: Normalizing learned features

Chapter 8.7.1

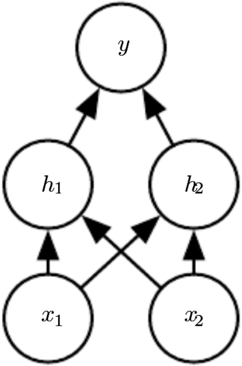
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.
- Batch norm: Normalize learned features.



Batch norm: Normalizing learned features

Chapter 8.7.1

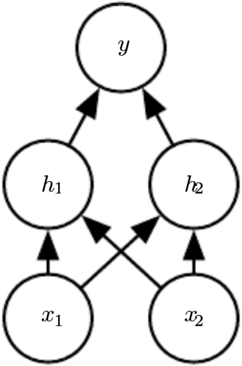
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.
- Batch norm: Normalize learned features.
- Q: Before or after activation $g(\cdot)$: $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$?



Batch norm: Normalizing learned features

Chapter 8.7.1

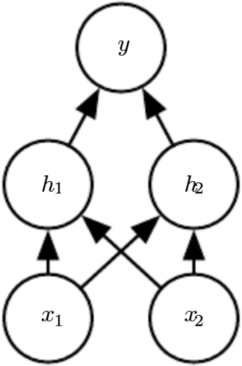
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.
- Batch norm: Normalize learned features.
- Q: Before or after activation $g(\cdot)$: $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$?
- A: There is debate; but before.



Batch norm: Normalizing learned features

Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

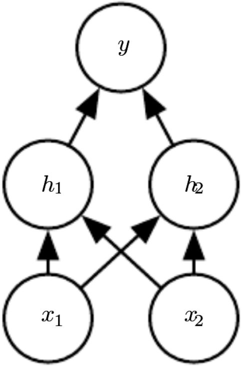


Batch norm: Normalizing learned features

Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n} \sum_{j=1}^{n} z^{(j)}$,

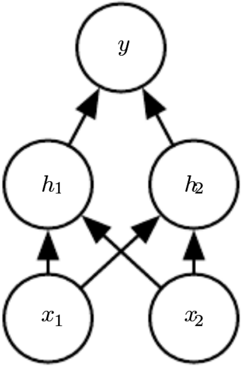


Batch norm: Normalizing learned features

Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,

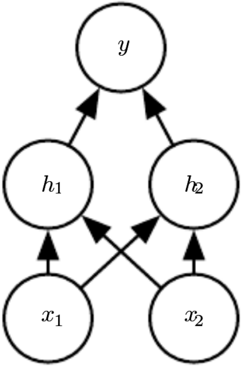


Batch norm: Normalizing learned features

Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),



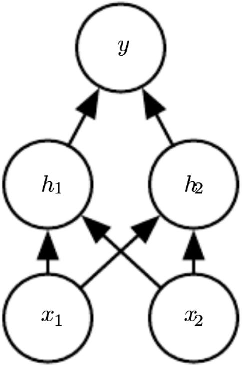
Batch norm: Normalizing learned features

Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.



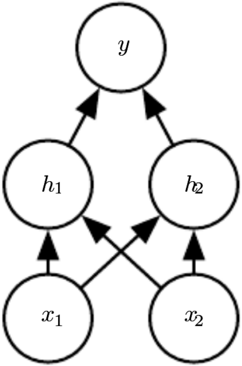
Batch norm: Normalizing learned features

Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:



Batch norm: Normalizing learned features

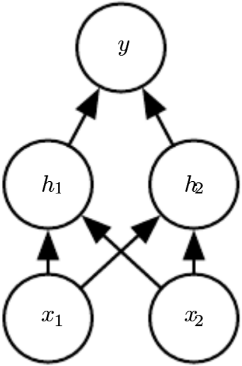
Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$



Batch norm: Normalizing learned features

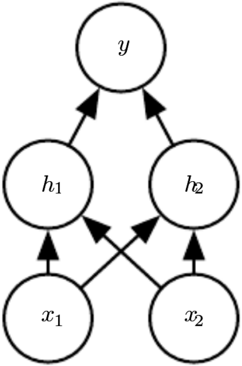
Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$
- Network can learn to undo it all:



Batch norm: Normalizing learned features

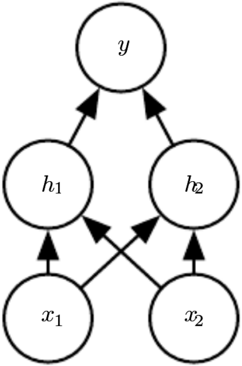
Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$
- Network can learn to undo it all:
  Q: How?



Batch norm: Normalizing learned features

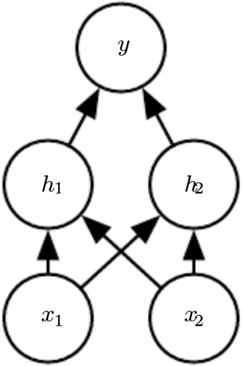
Chapter 8.7.1

For each layer i of n neurons $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{n}\sum_{j=1}^{n} z^{(j)}$,
- $\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n}(z^{(j)} - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$
- Network can learn to undo it all:
  Q: How? A: $\gamma_i = \sqrt{\delta + \sigma^2}$ and $\beta_i = \mu$



Mini-batches and Batch norm at test time

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.

Mini-batches and Batch norm at test time

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?

Mini-batches and Batch norm at test time

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of the last mini-batches.

Mini-batches and Batch norm at test time

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of the last mini-batches.
- How?

Mini-batches and Batch norm at test time

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of the last mini-batches.
- How? Our friend: the exponential weighted moving average

Mini-batches and Batch norm at test time

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of the last mini-batches.
- How? Our friend: the exponential weighted moving average
- Use the weighted average for μ and σ^2 and the learned parameters γ and β to compute $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$ for each test sample.

Questions?

Recap

- Stochastic Gradient Decent
- Learning rate
- Average and variance of past gradient update statistics help
- Exponentially weighted moving average
- Momentum: Average; RMSprop: variance; Adam: both.
- Feature and Batch normalization