CS4246 / CS5446

Tutorial Week 10

Muhammad **Rizki** Maulana
rizki@u.nus.edu

First

Review

- Linear Regression
- Neural Networks
- Gradient Descent
- Common Issues

Linear Regression

Single variable

f(x) = wx+b

(linear) function approximator

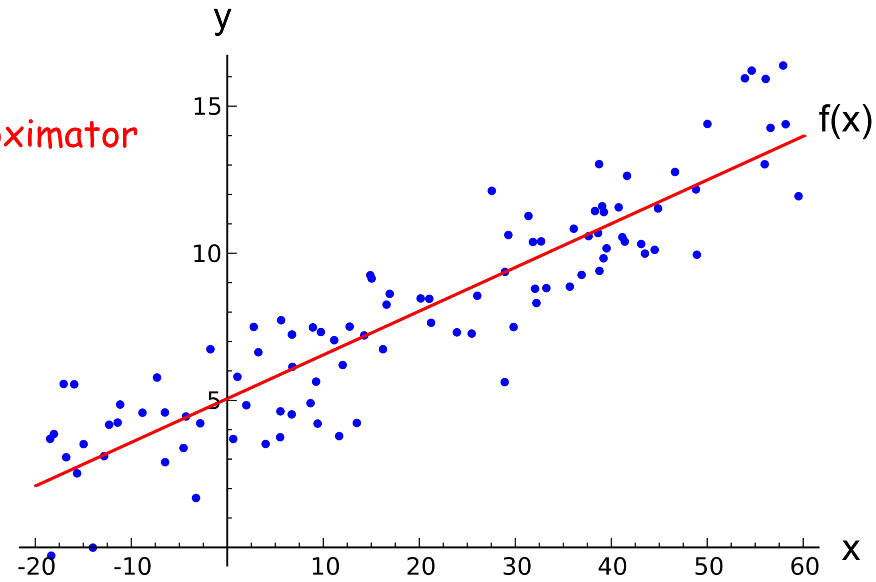


Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Linear Regression

Single variable

$f(x) = wx+b$

Multiple variables:

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b$

where $\mathbf{w} = [w_1,...,w_F]^T$, $\mathbf{x} = [x_1,...,x_F]^T$

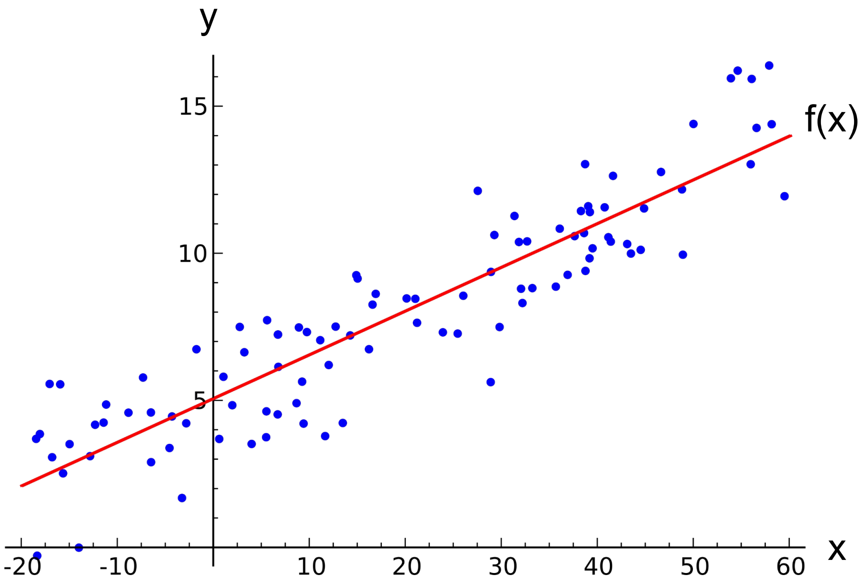


Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Linear Regression

Single variable

$f(x) = wx+b$

Multiple variables:

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b = w_1x_1+\ldots+w_Fx_F+b$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$

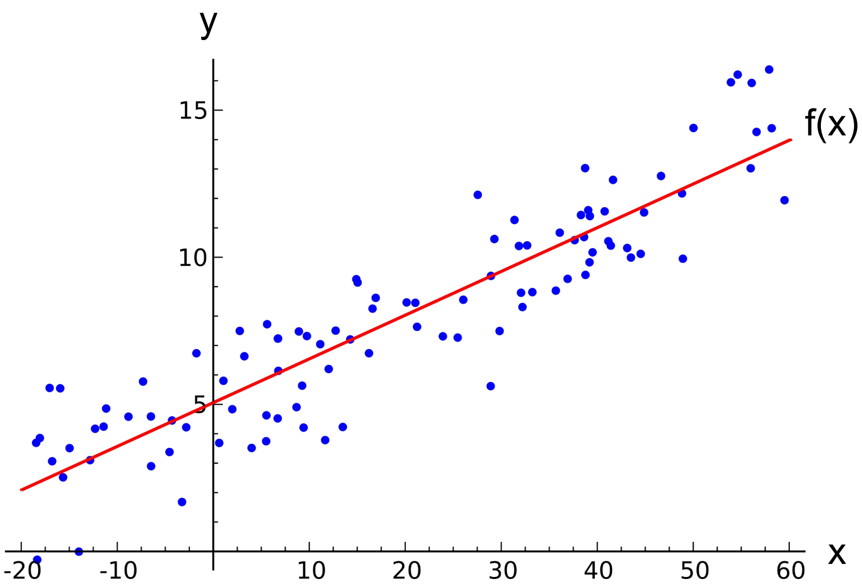


Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Linear Regression

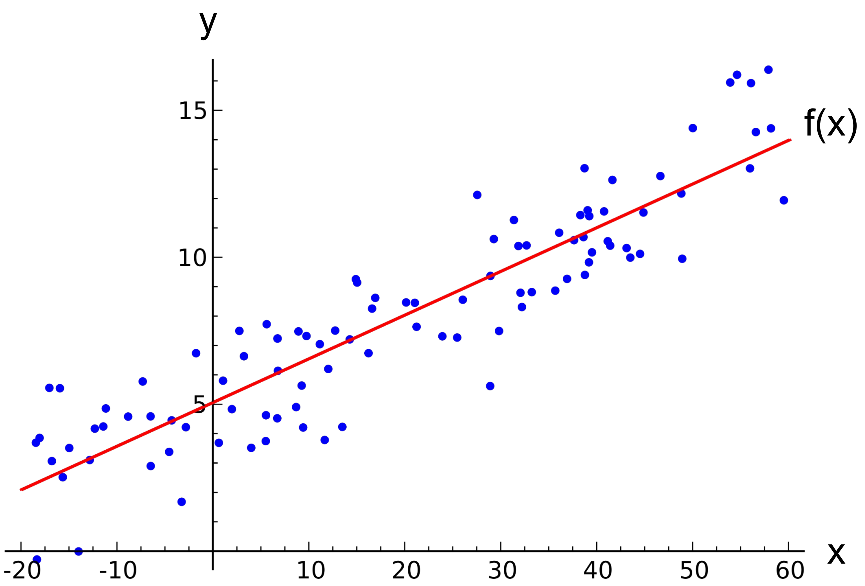
Single variable

$f(x) = wx+b$

Multiple variables:

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b = w_1x_1+...+w_Fx_F+b$

where $\mathbf{w} = [w_1,...,w_F]^T$, $\mathbf{x} = [x_1,...,x_F]^T$



Goal: minimize $\sum_{(x,y)} [\ f(x) - y\ ]^2$

Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Linear Regression

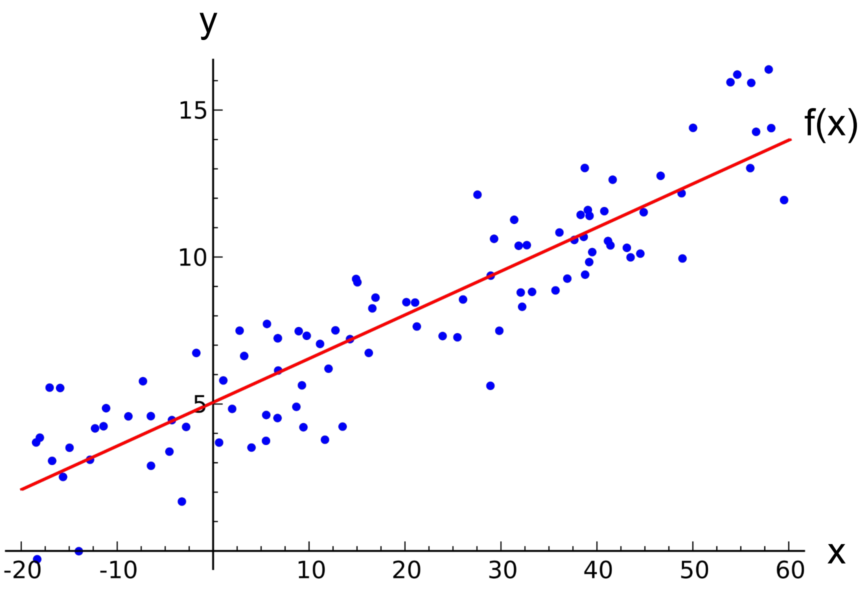
Single variable

$f(x) = wx+b$

Multiple variables:

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b = w_1x_1+\ldots+w_Fx_F+b$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$



Goal: minimize $\underbrace{\sum_{(x,y)} [\, f(x) - y \,]^2}_{\text{Squared distance}}$

Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Linear Regression

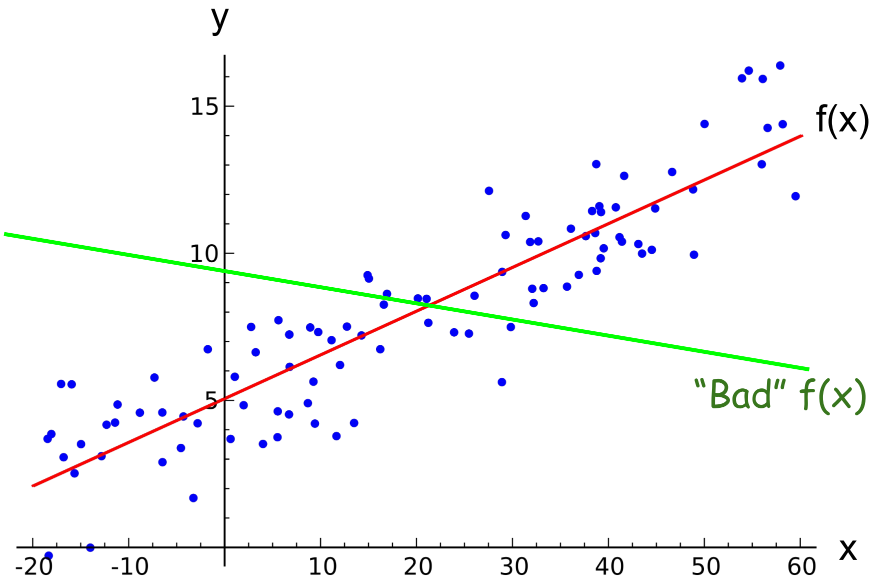
Single variable

$f(x) = wx+b$

Multiple variables:

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b = w_1x_1+\ldots+w_Fx_F+b$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$



Goal: minimize $\underbrace{\sum_{(x,y)} [\ f(x) - y\ ]^2}_{\text{Squared distance}}$

Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Linear Regression

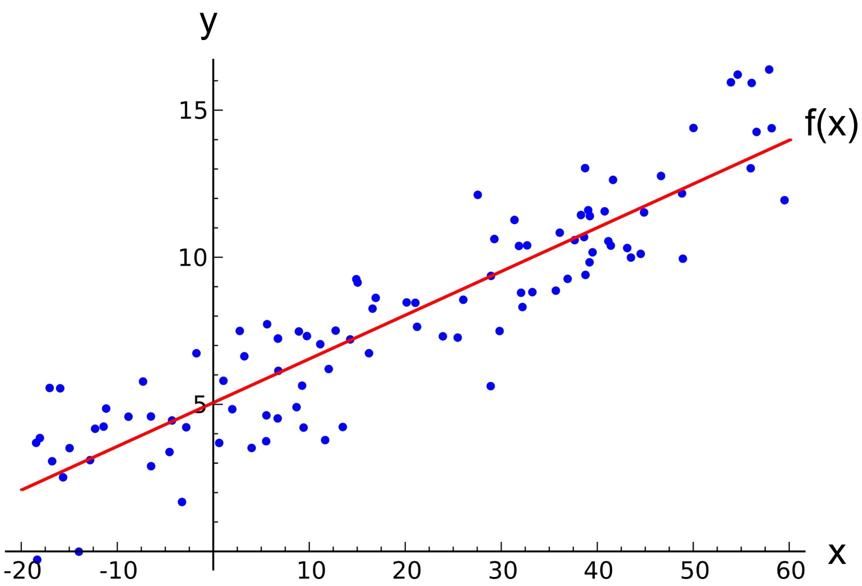
Single variable

$f(x) = wx+b$

Multiple variables:

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b = w_1x_1+\ldots+w_Fx_F+b$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$



Goal: minimize $\sum_{(x,y)} [\, f(x) - y \,]^2 = L_{(x,y)}(\, f(x)\, ,\, y\,)$

Generally: loss function

Image source: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Neural Networks (NN)

(another) function approximator

Neural Networks (NN)

Recall our linear function

$f(\mathbf{x}) = \quad \mathbf{w}^T\mathbf{x}+b$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$

Neural Networks (NN)

Simple NN (Neuron)

Now it goes **neural**!

$$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$

Neural Networks (NN)

Simple NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,...,w_F]^T$, $\mathbf{x} = [x_1,...,x_F]^T$,

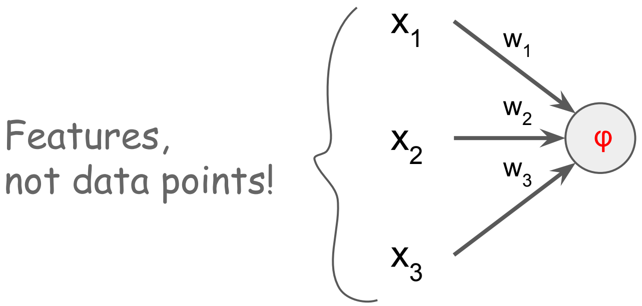
φ non-linear activation function

Neural Networks (NN)

Simple NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,
φ non-linear activation function



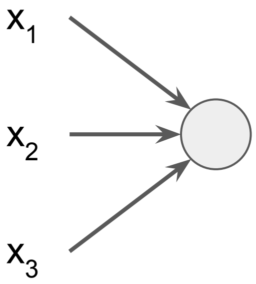
Neural Networks (NN)

Simple NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function



Neural Networks (NN)

Simple NN

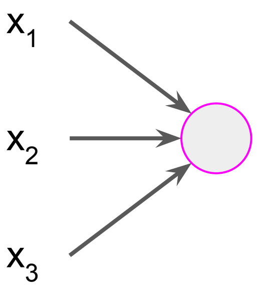
$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function

"Deep" NN

$f(\mathbf{x}) =$ $f_{i1}(\mathbf{x})$



Neural Networks (NN)

Simple NN

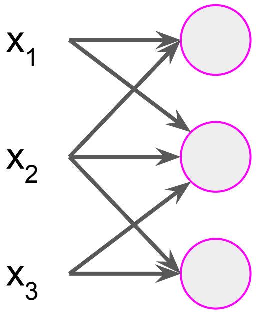
$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function

"Deep" NN

$f(\mathbf{x}) =$ $f_{i1}(\mathbf{x})\ldots$ $\ldots$



Neural Networks (NN)

Simple NN

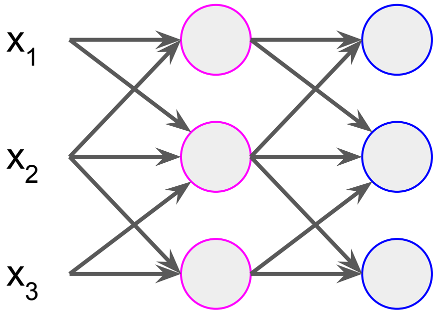
$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function

"Deep" NN

$f(\mathbf{x}) = f_{H1}([f_{i1}(\mathbf{x})\ldots]^T) , \ldots, f_{HM}([\ldots]^T)$



Neural Networks (NN)

Simple NN

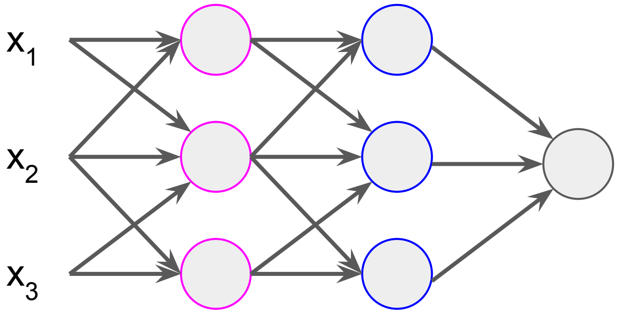
$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function

"Deep" NN

$f(\mathbf{x}) = f_O([\, f_{H1}([f_{i1}(\mathbf{x})\ldots]^T)\, ,\, \ldots,\, f_{HM}([\ldots]^T)\,]^T)$



Neural Networks (NN)

“building blocks”

Simple NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

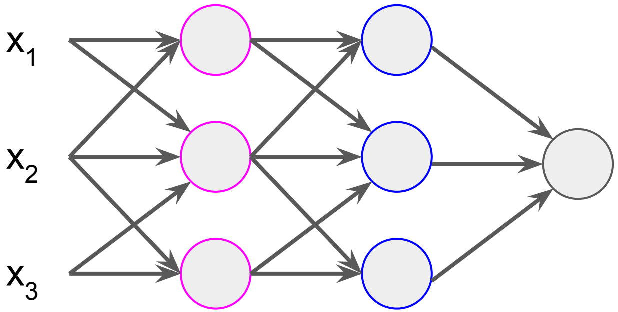
where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function

Convolutional NN

“Deep” NN

$f(\mathbf{x}) = f_O([\ f_{H1}([f_{i1}(\mathbf{x})\ldots]^T)\ ,\ \ldots,\ f_{HM}([\ldots]^T)\ ]^T)$



Neural Networks (NN)

Simple NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,...,w_F]^T$, $\mathbf{x} = [x_1,...,x_F]^T$,

φ non-linear activation function

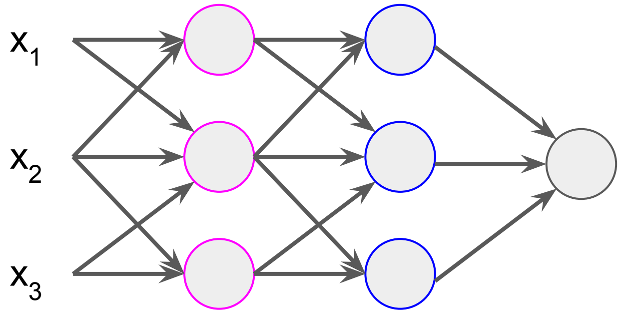
Convolutional NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T \ast \mathbf{x})$

where $\mathbf{w} = [w_1,...,w_K]^T$, $K \leq F$

“Deep” NN

$f(\mathbf{x}) = f_O([\ f_{H1}([f_{i1}(\mathbf{x})...]^T)\ , \ldots, f_{HM}([...]^T)\ ]^T)$



Neural Networks (NN)

Simple NN

$$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F]^T$,

φ non-linear activation function

Convolutional NN

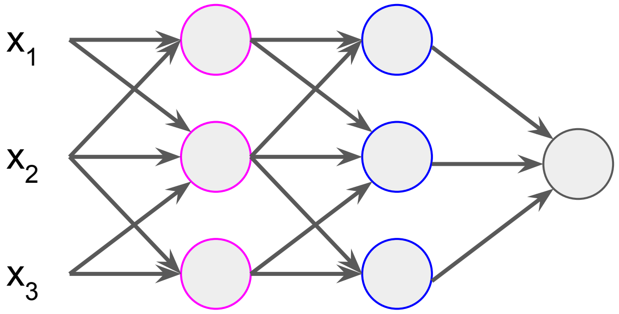
$$f(\mathbf{x}) = \varphi(\mathbf{w}^T * \mathbf{x})$$

where $\mathbf{w} = [w_1,\ldots,w_K]^T$, $K \leq F$

x f(**x**)

"Deep" NN

$$f(\mathbf{x}) = f_O([\ f_{H1}([f_{i1}(\mathbf{x})\ldots]^T)\ ,\ \ldots,\ f_{HM}([\ldots]^T)\ ]^T)$$



Source: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Neural Networks (NN)

Simple NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x}+b)$

where $\mathbf{w} = [w_1,\ldots,w_F]^T$, $\mathbf{x} = [x_1,\ldots,x_F$

φ non-linear activation func

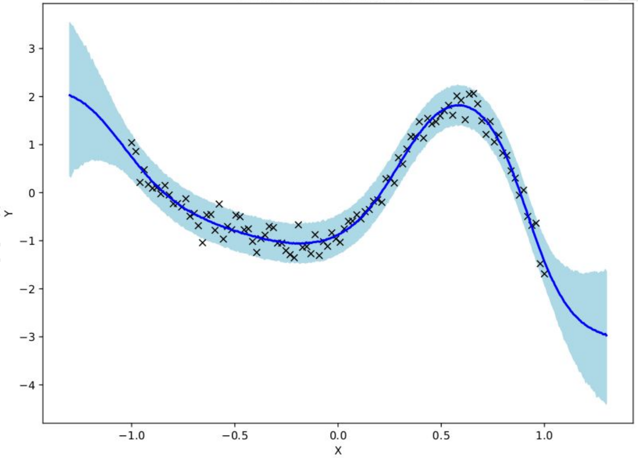
Convolutional NN

$f(\mathbf{x}) = \varphi(\mathbf{w}^T * \mathbf{x})$

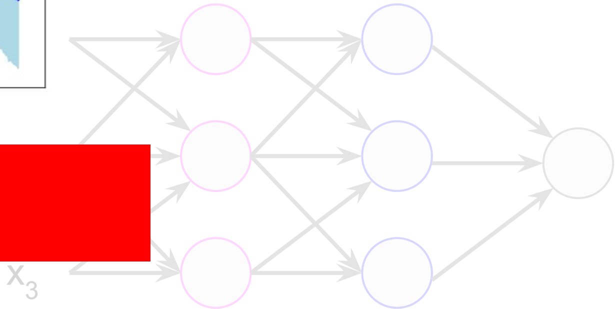
where $\mathbf{w} = [w_1,\ldots,w_K]^T$, $K \leq F$

"Deep" NN

$= f_O([\, f_{H1}([f_{i1}(\mathbf{x})\ldots]^T) , \ldots , f_{HM}([\ldots]^T)$



Goal: Minimize Loss



$\mathbf{x}$ $f(\mathbf{x})$

http://num.pyro.ai/en/0.5.0/examples/bnn.html

Source: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

The only independent variable!

Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$

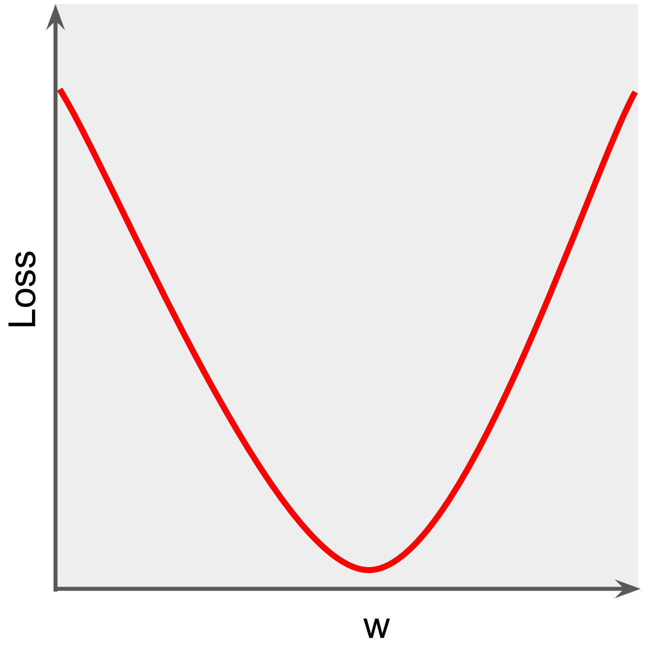
Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$



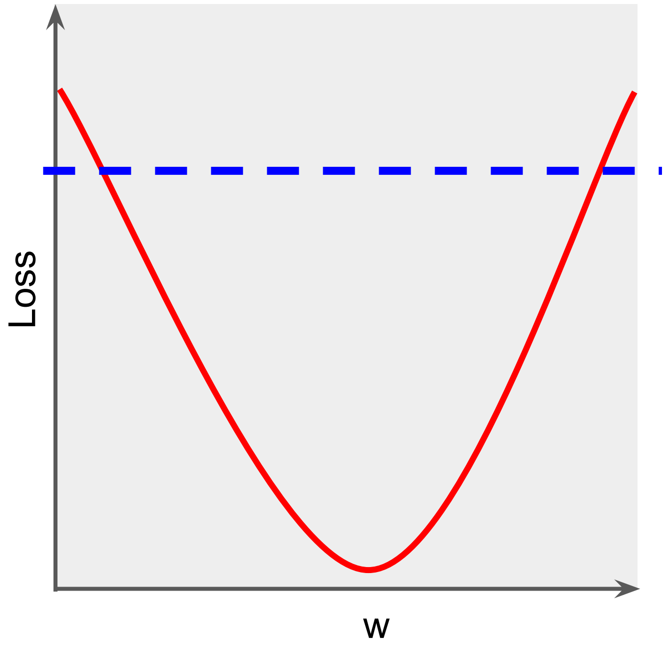
Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$



Minimizing Loss

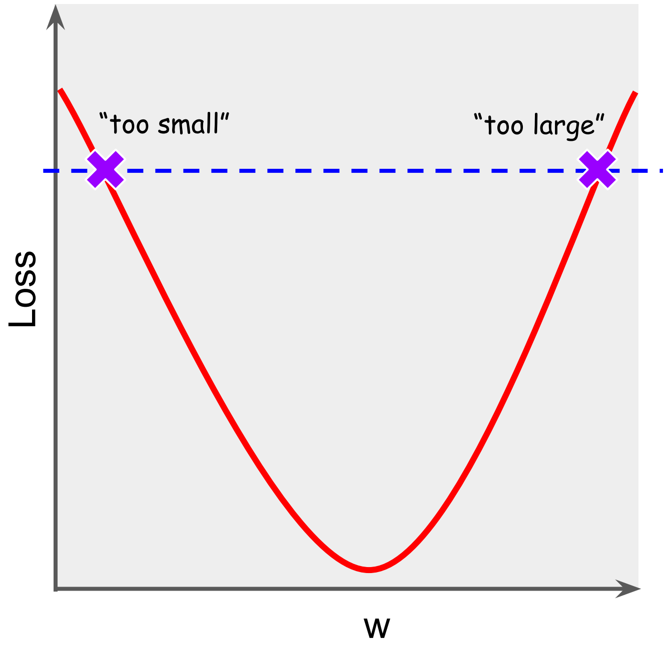
Consider:

- $\text{Loss} = \sum_{(x,y)} [\, f(x) - y \,]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\, f(x) - y \,]^2 = [\, wx - y \,]^2$

wx < y → "too small"
wx > y → "too large"



Minimizing Loss

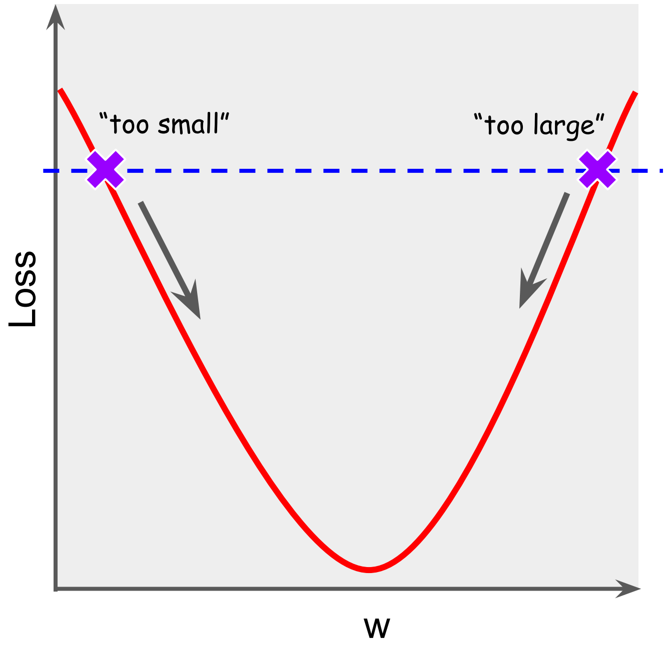
Consider:

- $\text{Loss} = \sum_{(x,y)} [\, f(x) - y \,]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\, f(x) - y \,]^2 = [\, wx - y \,]^2$

$wx < y \rightarrow$ "too small"
$wx > y \rightarrow$ "too large"



Minimizing Loss

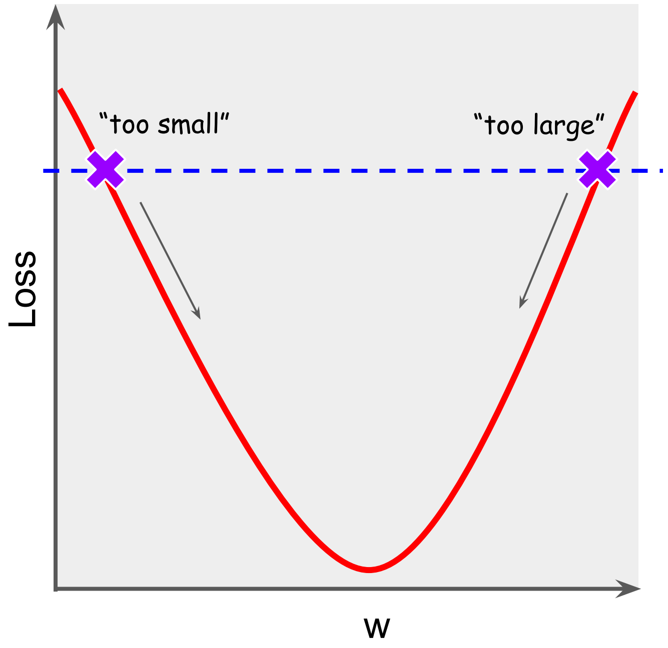
Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$

$wx < y \rightarrow$ "too small"
$wx > y \rightarrow$ "too large"



How do we know which one?

(should be general enough for any function)

Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

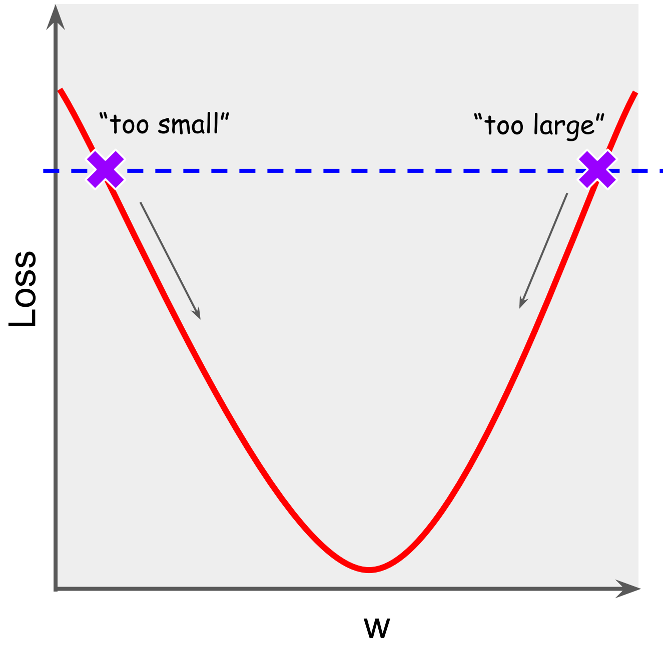
$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$

wx < y → "too small"
wx > y → "too large"

How do we know which one?

$dL_{(x,y)}/dw = 2[\ wx - y\ ]x$

"too small" → gradient -ve
"too large" → gradient +ve



Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$

$wx < y \rightarrow$ "too small"
$wx > y \rightarrow$ "too large"

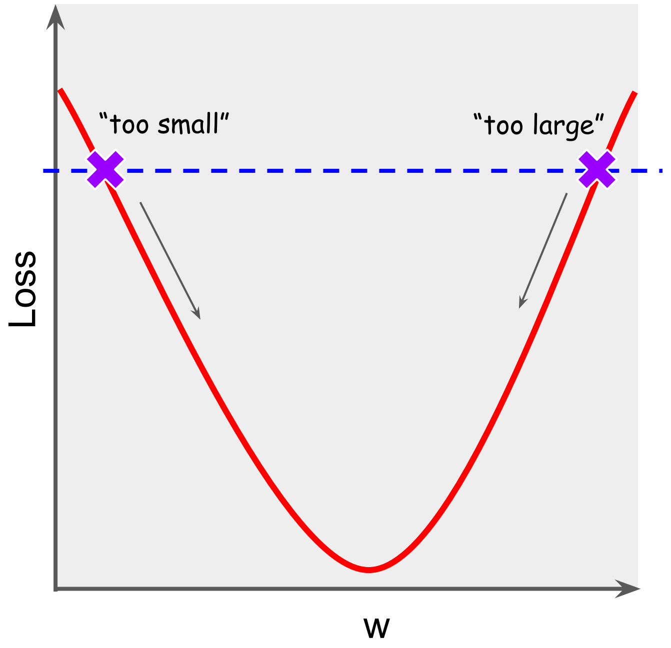
How do we know which one?

$dL_{(x,y)}/dw = 2[\ wx - y\ ]x$

"too small" → gradient -ve
"too large" → gradient +ve

Update:

$w_t = w_{t-1}$ - gradient



Minimizing Loss

Consider:

- Loss = $\sum_{(x,y)} [\ f(x) - y\ ]^2$
- $f(x) = wx$

How to find w such that we get minimum loss?

$L_{(x,y)} = [\ f(x) - y\ ]^2 = [\ wx - y\ ]^2$

$wx < y \rightarrow$ "too small"
$wx > y \rightarrow$ "too large"

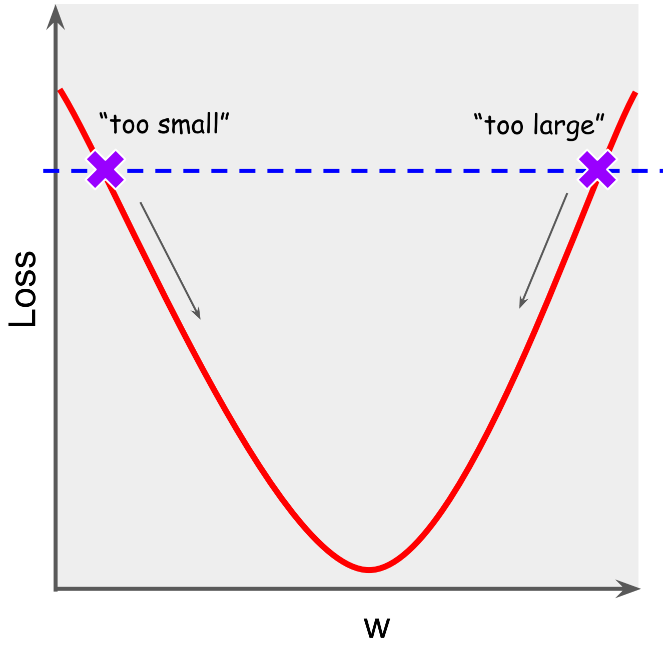
How do we know which one?

$dL_{(x,y)}/dw = 2[\ wx - y\ ]x$

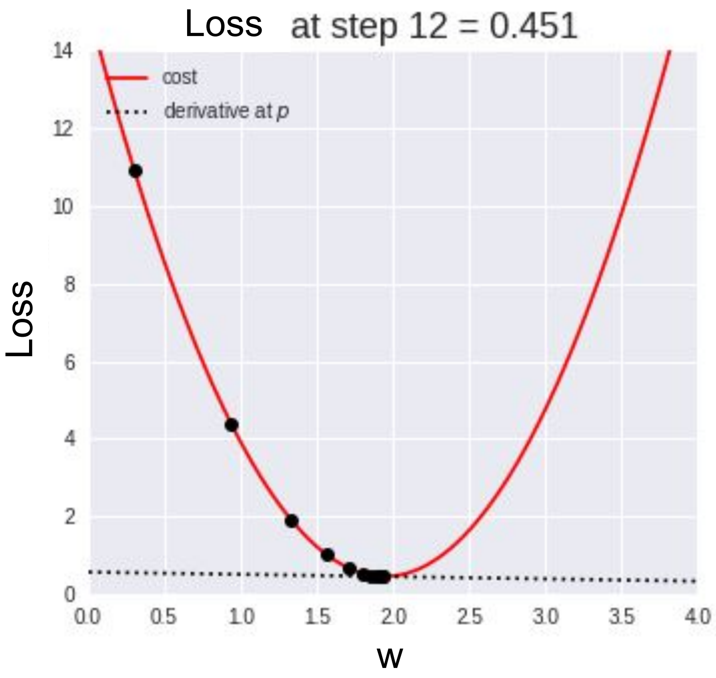
"too small" $\rightarrow$ gradient -ve
"too large" $\rightarrow$ gradient +ve

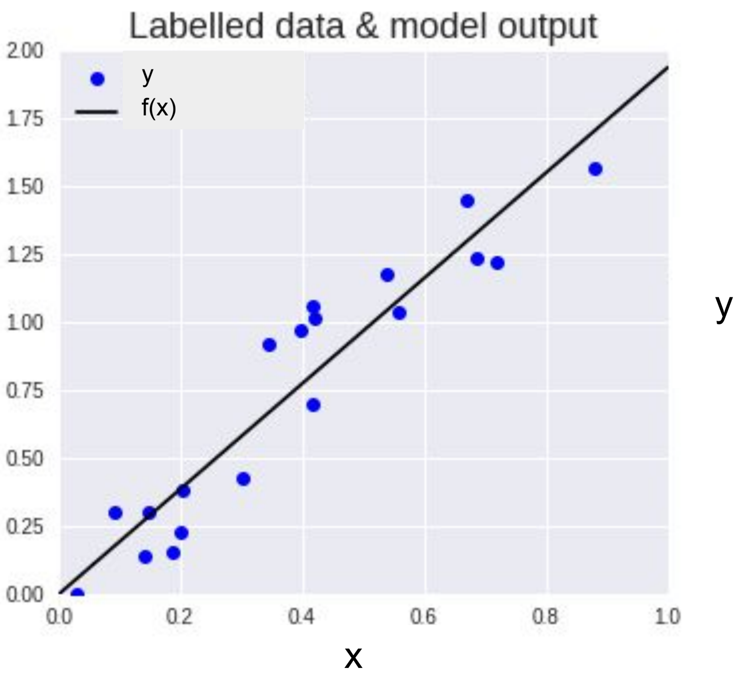
Update:

$w_t = w_{t-1}$ - gradient (do in a loop) ← Gradient Descent



Gradient Descent: Illustration





Source: https://camo.githubusercontent.com/a401a48f5503c52004369148a784e779aa7e3411/68747470733a2f2f63646e2d696d616765732d312e6d656469756d2e636f6d2f6d61782f313630302f312a70775049472d475748796150564d564747354f6841512e676966

Gradient Descent: Backpropagation

What about complicated functions (e.g., neural networks)?

Gradient Descent: Backpropagation

What about complicated functions (e.g., neural networks)?

Backpropagation = Chain Rule of Calculus

$$\frac{da}{dx}a(b(c(d(e(f(g(x)))))))$$

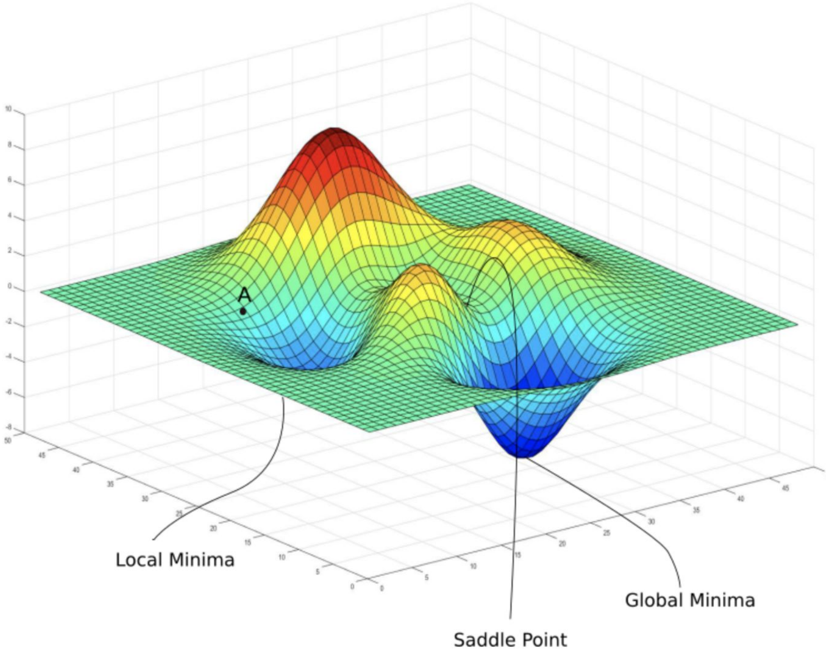
$$\frac{da}{\partial x} = \frac{da}{db} \times \frac{db}{dc} \times \frac{dc}{dd} \times \frac{dd}{de} \times \frac{de}{df} \times \frac{df}{\mathrm{dg}} \times \frac{\mathrm{dg}}{dx}$$

Source: https://ai.plainenglish.io/calculus-for-backpropagation-doesnt-have-to-be-scary-16595d76e744

Gradient Descent: Issues with Deep Neural Networks

Gradient Descent: Issues with Deep Neural Networks

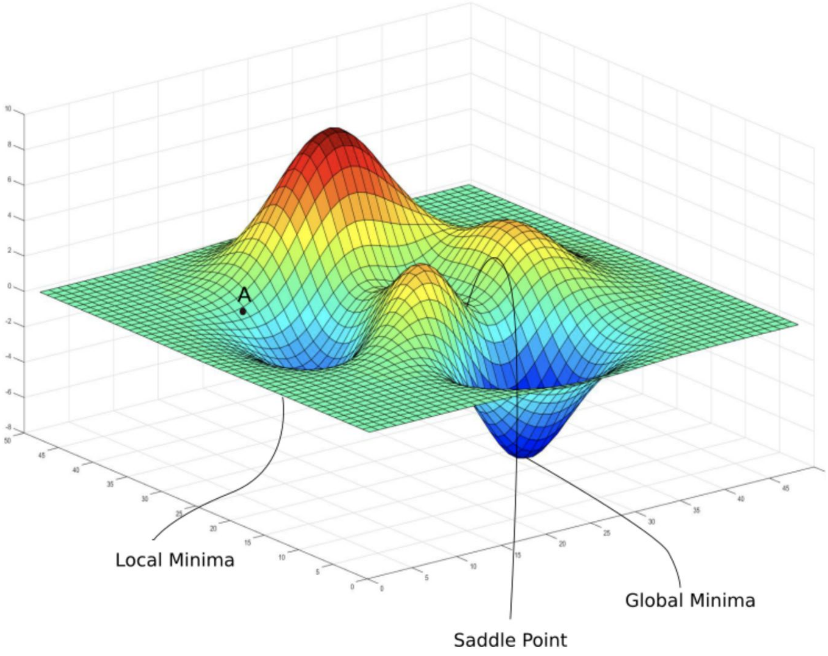
Local Minima



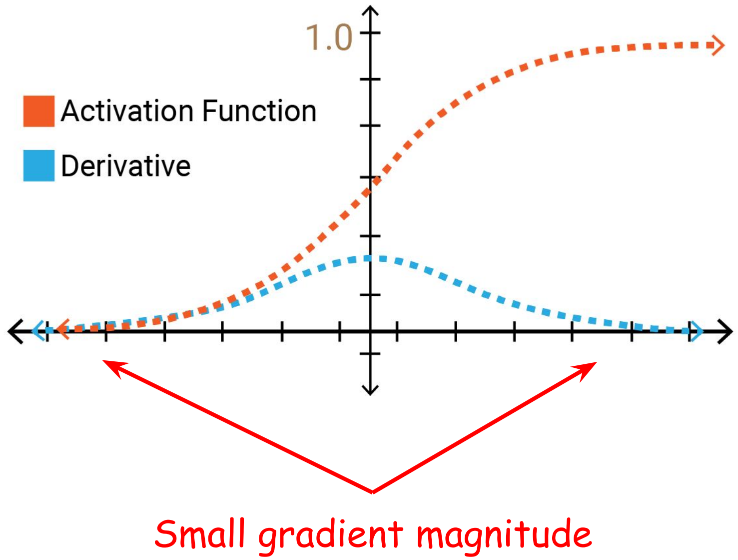
Sources:
https://medium.com/analytics-vidhya/journey-of-gradient-descent-from-local-to-global-c851eba3d367

Gradient Descent: Issues with Deep Neural Networks

Local Minima



Vanishing Gradient



Sources:
https://medium.com/analytics-vidhya/journey-of-gradient-descent-from-local-to-global-c851eba3d367
https://towardsdatascience.com/the-problem-of-vanishing-gradients-68cea05e2625

Issue with Function Approximation in RL

The Deadly Triad in RL:

- **Function Approximation**
- Bootstrapping
- Off-policy Learning

Second

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x,y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)}$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x, y)$$

$$L = \frac{1}{2}\left(R(s \quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\right)^2$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x,y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x,y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s\quad) + \gamma \hat{U}_\theta(s')}_{\text{Target}} - \underbrace{\hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x,y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x,y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s\quad) + \gamma \hat{U}_\theta(s')}_{\text{Target}} - \underbrace{\hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2 \qquad \theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

Question

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x, y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s) + \gamma \hat{U}_\theta(s')}_{\text{Target}} - \underbrace{\hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2 \qquad \theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

$$\frac{\partial L}{\partial \theta_i} = \Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\left[-\frac{\partial \hat{U}_\theta}{\partial \theta_i}\right]$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x, y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s) + \gamma \hat{U}_\theta(s')}_{\text{Target}} - \underbrace{\hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2 \qquad \theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

$$\frac{\partial L}{\partial \theta_i} = \Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\left[-\frac{\partial \hat{U}_\theta}{\partial \theta_i}\right]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = 1 \qquad \theta_0 \leftarrow \theta_0 - \alpha\Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-1\Big]$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x,y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s\quad) + \gamma \hat{U}_\theta(s')}_{\text{Target}} \underbrace{- \hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2 \qquad \theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

$$\frac{\partial L}{\partial \theta_i} = \Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-\frac{\partial \hat{U}_\theta}{\partial \theta_i}\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = 1 \qquad \theta_0 \leftarrow \theta_0 - \alpha\Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-1\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = x \qquad \theta_1 \leftarrow \theta_1 - \alpha\Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-x\Big]$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x, y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s\quad) + \gamma \hat{U}_\theta(s')}_{\text{Target}} - \underbrace{\hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2 \qquad \theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

$$\frac{\partial L}{\partial \theta_i} = \Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-\frac{\partial \hat{U}_\theta}{\partial \theta_i}\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = 1 \qquad \theta_0 \leftarrow \theta_0 - \alpha\Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-1\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = x \qquad \theta_1 \leftarrow \theta_1 - \alpha\Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-x\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = y \qquad \theta_2 \leftarrow \theta_2 - \alpha\Big(R(s\quad) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-y\Big]$$

[RN 21.4] Write out the parameter update equations for TD learning with

$$\hat{U}_\theta(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g) + (y - y_g)} \qquad s = (x, y)$$

$$L = \frac{1}{2}\Big(\underbrace{R(s) + \gamma \hat{U}_\theta(s')}_{\text{Target}} - \underbrace{\hat{U}_\theta(s)}_{\text{Prediction}}\Big)^2 \qquad \theta_i \leftarrow \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

$$\frac{\partial L}{\partial \theta_i} = \Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-\frac{\partial \hat{U}_\theta}{\partial \theta_i}\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = 1 \qquad \theta_0 \leftarrow \theta_0 - \alpha\Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-1\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = x \qquad \theta_1 \leftarrow \theta_1 - \alpha\Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-x\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = y \qquad \theta_2 \leftarrow \theta_2 - \alpha\Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-y\Big]$$

$$\frac{\partial \hat{U}_\theta}{\partial \theta_0} = \sqrt{(x - x_g) + (y - y_g)} \qquad \theta_3 \leftarrow \theta_3 - \alpha\Big(R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)\Big)\Big[-\sqrt{(x - x_g) + (y - y_g)}\Big]$$

Third

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Q	$x = 0$	$x = 1$
a_1	0	0
a_2	0	0

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Q	$x = 0$	$x = 1$
a_1	0	0
a_2	0	0

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Q	$x = 0$	$x = 1$
a_1	0	0
a_2	0	0

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

i. First observed transition: initial value of $x = 0$, observed reward $r = 10$, action a_1, next state $x = 1$.

Question

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Q	$x=0$	$x=1$
a_1	0	0
a_2	0	0

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

i. First observed transition: initial value of $x = 0$, observed reward $r = 10$, action a_1, next state $x = 1$.

$$Q(0,a_1) \leftarrow 0 + 0.5(10 + 0.9\max(0,0) - 0) = 5$$

Q	$x=0$	$x=1$
a_1	$\boxed{5}$	0
a_2	0	0

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Q	$x=0$	$x=1$
a_1	0	0
a_2	0	0

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

i. First observed transition: initial value of $x = 0$, observed reward $r = 10$, action a_1, next state $x = 1$.

$$Q(0,a_1) \leftarrow 0 + 0.5(10 + 0.9\max(0,0) - 0) = 5$$

Q	$x=0$	$x=1$
a_1	5	0
a_2	0	0

ii. Second observed transition: from $x = 1$, observed reward $r = -5$, action a_2, next state $x = 0$.

Question

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(a) Perform two steps of Q-learning with the observed transitions shown below in (i) and (ii) using a table representation of the Q-function. Use a learning rate = 0.5 starting from a table with all entries initialized to 0. Show the Q-function after each step.

Q	$x = 0$	$x = 1$
a_1	0	0
a_2	0	0

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)).$$

i. First observed transition: initial value of $x = 0$, observed reward $r = 10$, action a_1, next state $x = 1$.

$$Q(0,a_1) \leftarrow 0 + 0.5(10 + 0.9 \max(0,0) - 0) = 5$$

Q	$x = 0$	$x = 1$
a_1	5	0
a_2	0	0

ii. Second observed transition: from $x = 1$, observed reward $r = -5$, action a_2, next state $x = 0$.

$$Q(1,a_2) \leftarrow 0 + 0.5(-5 + 0.9 \max(5,0) - 0) = -0.25$$

Q	$x = 0$	$x = 1$
a_1	5	0
a_2	0	-0.25

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(b) Now perform Q-learning with function approximation using

- $Q(x, a_1) = \beta_1 x$
- $Q(x, a_2) = \beta_2 x$
- $\alpha = 0.5$
- $\beta_1 = 0$
- $\beta_2 = 0$

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(b) Now perform Q-learning with function approximation using

- $Q(x, a_1) = \beta_1 x$
- $Q(x, a_2) = \beta_2 x$
- $\alpha = 0.5$
- $\beta_1 = 0$
- $\beta_2 = 0$

$$\beta_i \leftarrow \beta_i + \alpha(R(x) + \gamma \max_{a'} Q(x', a') - Q(x, a)) \frac{\partial Q(x, a)}{\partial \beta_i}$$

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(b) Now perform Q-learning with function approximation using

$$\beta_i \leftarrow \beta_i + \alpha(R(x) + \gamma \max_{a'} Q(x', a') - Q(x, a))\frac{\partial Q(x, a)}{\partial \beta_i}$$

- $Q(x, a_1) = \beta_1 x$
- $Q(x, a_2) = \beta_2 x$
- $\alpha = 0.5$
- $\beta_1 = 0$
- $\beta_2 = 0$

i. First observed transition: initial value of $x = 1$, observed reward $r = 10$, action a_1, next state $x = 1$.

Question

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(b) Now perform Q-learning with function approximation using

- $Q(x, a_1) = \beta_1 x$
- $Q(x, a_2) = \beta_2 x$
- $\alpha = 0.5$
- $\beta_1 = 0$
- $\beta_2 = 0$

$$\beta_i \leftarrow \beta_i + \alpha(R(x) + \gamma \max_{a'} Q(x', a') - Q(x, a)) \frac{\partial Q(x, a)}{\partial \beta_i}$$

i. First observed transition: initial value of $x = 1$, observed reward $r = 10$, action a_1, next state $x = 1$.

$Q(x, a_1) = \beta_1 x$

$$\beta_1 \leftarrow 0 + 0.5(10 + 0.9 \max(0, 0) - 0)1 = 5$$

$$\beta_2 \leftarrow 0 + 0.5(10 + 0.9 \max(0, 0) - 0)0 = 0$$

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(b) Now perform Q-learning with function approximation using

- $Q(x, a_1) = \beta_1 x$
- $Q(x, a_2) = \beta_2 x$
- $\alpha = 0.5$
- $\beta_1 = 0$
- $\beta_2 = 0$

$$\beta_i \leftarrow \beta_i + \alpha(R(x) + \gamma \max_{a'} Q(x', a') - Q(x, a)) \frac{\partial Q(x, a)}{\partial \beta_i}$$

i. First observed transition: initial value of $x = 1$, observed reward $r = 10$, action a_1, next state $x = 1$.

$Q(x, a_1) = \beta_1 x$

$$\beta_1 \leftarrow 0 + 0.5(10 + 0.9 \max(0, 0) - 0)1 = 5$$

$$\beta_2 \leftarrow 0 + 0.5(10 + 0.9 \max(0, 0) - 0)0 = 0$$

ii. Second observed transition: from $x = 1$, observed reward $r = -5$, action a_2, next state $x = 0$.

Question

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(b) Now perform Q-learning with function approximation using

- $Q(x, a_1) = \beta_1 x$
- $Q(x, a_2) = \beta_2 x$
- $\alpha = 0.5$
- $\beta_1 = 0$
- $\beta_2 = 0$

$$\beta_i \leftarrow \beta_i + \alpha(R(x) + \gamma \max_{a'} Q(x', a') - Q(x, a))\frac{\partial Q(x, a)}{\partial \beta_i}$$

i. First observed transition: initial value of $x = 1$, observed reward $r = 10$, action a_1, next state $x = 1$.

$Q(x, a_1) = \beta_1 x$

$$\beta_1 \leftarrow 0 + 0.5(10 + 0.9 \max(0, 0) - 0)1 = 5$$

$$\beta_2 \leftarrow 0 + 0.5(10 + 0.9 \max(0, 0) - 0)0 = 0$$

ii. Second observed transition: from $x = 1$, observed reward $r = -5$, action a_2, next state $x = 0$.

$Q(x, a_2) = \beta_2 x$

$$\beta_1 \leftarrow 5 + 0.5(-5 + 0.9 \max(0, 0) - 0)0 = 5$$

$$\beta_2 \leftarrow 0 + 0.5(-5 + 0.9 \max(0, 0) - 0)1 = -2.5$$

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(c) After enough data is observed, which method would give better performance, the tabular method in (a) or the function approximation method in (b)? Why? Suggest how the poorer performing method can be improved.

Question

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(c) After enough data is observed, which method would give better performance, the tabular method in (a) or the function approximation method in (b)? Why? Suggest how the poorer performing method can be improved.

Tabular is better

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(c) After enough data is observed, which method would give better performance, the tabular method in (a) or the function approximation method in (b)? Why? Suggest how the poorer performing method can be improved.

Tabular is better

Failure case of the function approximation

$$Q(x, a1) = \beta_1 x \text{ and } Q(x, a2) = \beta_2 x$$

Can't have non-zero value for x=0

Consider a system with a single state variable x that can take value 0 or 1 and actions a_1 and a_2. An agent can observe the value of the state variable as well as the reward in the observed state. Assume a discount factor $\gamma = 0.9$.

(c) After enough data is observed, which method would give better performance, the tabular method in (a) or the function approximation method in (b)? Why? Suggest how the poorer performing method can be improved.

Tabular is better

Failure case of the function approximation

$$Q(x, a1) = \beta_1 x \text{ and } Q(x, a2) = \beta_2 x$$

Can't have non-zero value for x=0

Improvements

$$Q(x, a_1) = \beta_1 x + \delta_1 \text{ and } Q(x, a2) = \beta_2 x + \delta_2$$

Question?

<EOF>