Deep Learning
MSDS 631

Tips and Tricks for Deep Learning

Michael Ruddy

Some Thoughts

- Final Project Reminders
- Tips
  - Loss Function Documentation (output? average?)
  - What is true for this dataset may not be true for another
  - For experiments, make sure you are re-initializing parameters
- Only small penalty for submitted lab on time
  - Come see me today for issues!
- Please put your name on your lab
- Isn't it enough to know how to use software for DL? Why peak "under the hood"?

Questions?

- From last lecture?
- From the lab assignment?

Overview

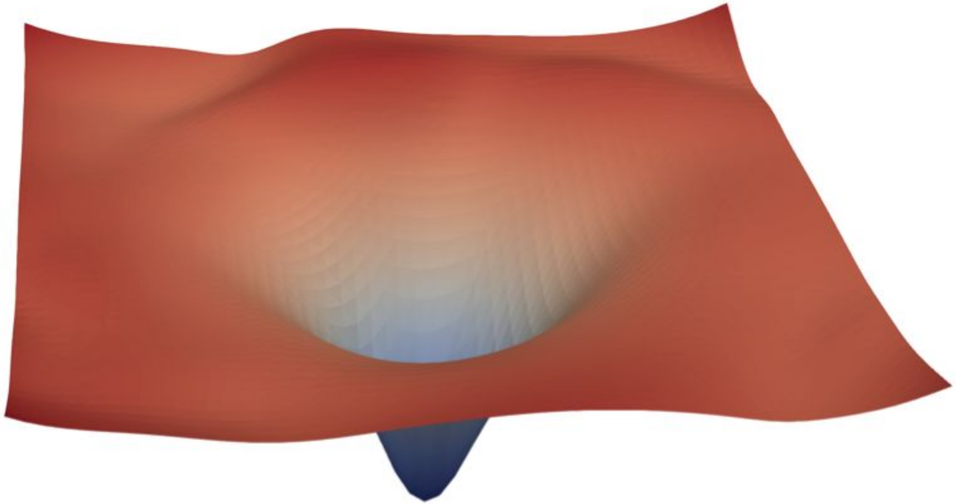
- What is training a model?
- Optimization
- Regularization
- Embeddings

What does it mean to train a model?

- Gradient Descent: Navigating through the space of possible models
  - Trying to find a “good” minimum for our loss function

What does it mean to train a model?

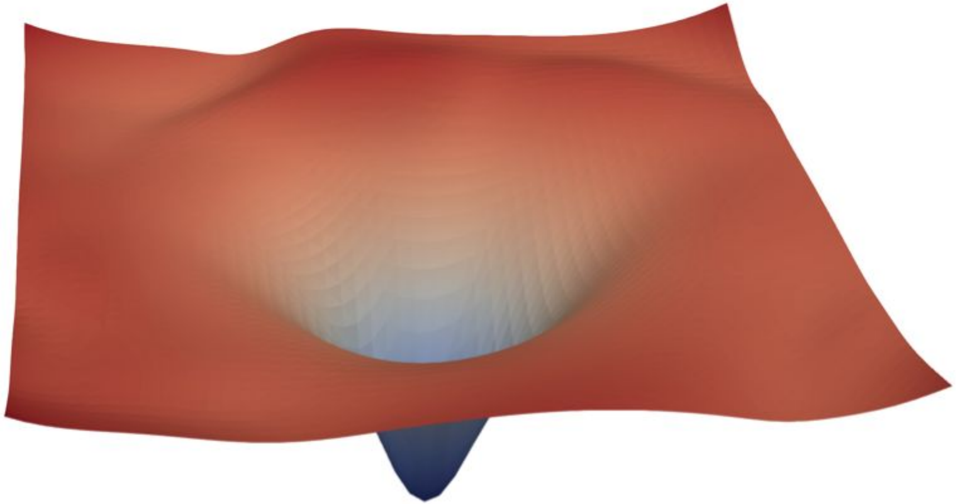
- Gradient Descent: Navigating through the space of possible models
  - Trying to find a “good” minimum for our loss function
- Loss Landscape: The environment which informs our journey



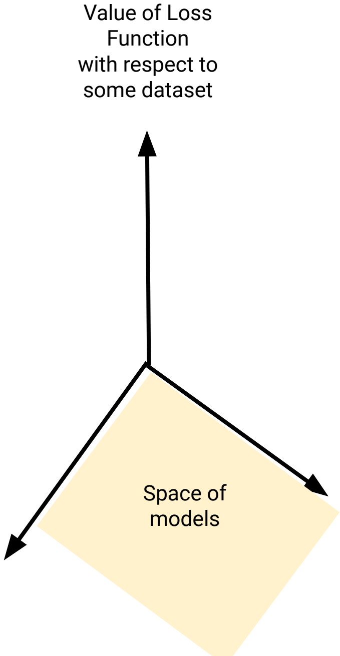
From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- Gradient Descent: Navigating through the space of possible models
    - Trying to find a “good” minimum for our loss function
- Loss Landscape: The environment which informs our journey

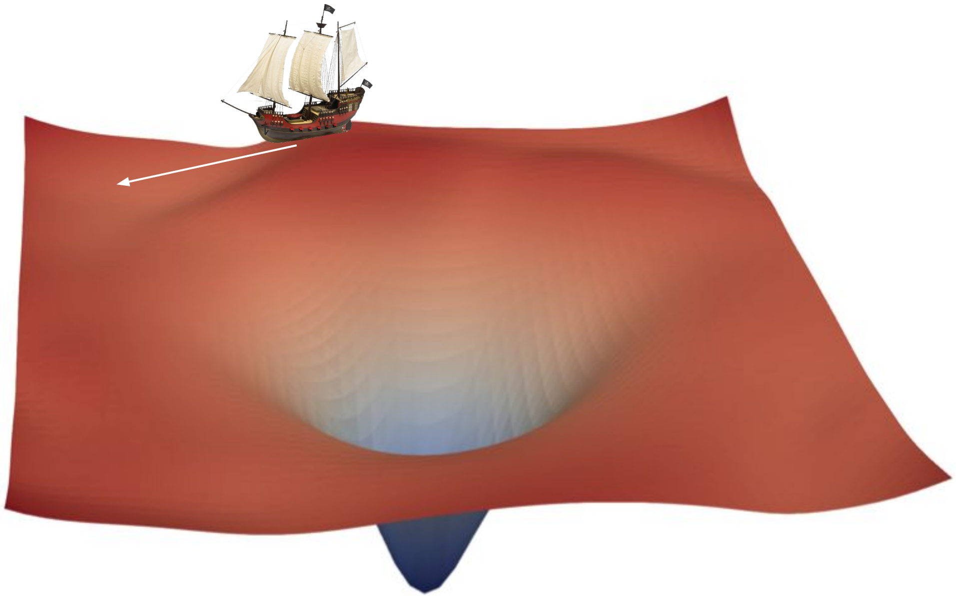


From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein



What does it mean to train a model?

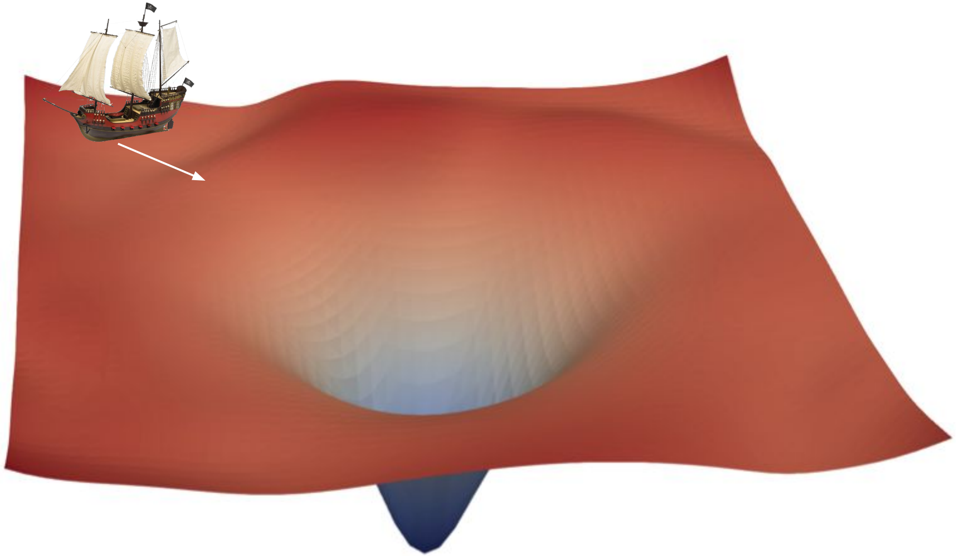
- Gradient Descent: Navigating through the space of possible models
  - Trying to find a "good" minimum for our loss function
- Loss Landscape: The environment which informs our journey



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

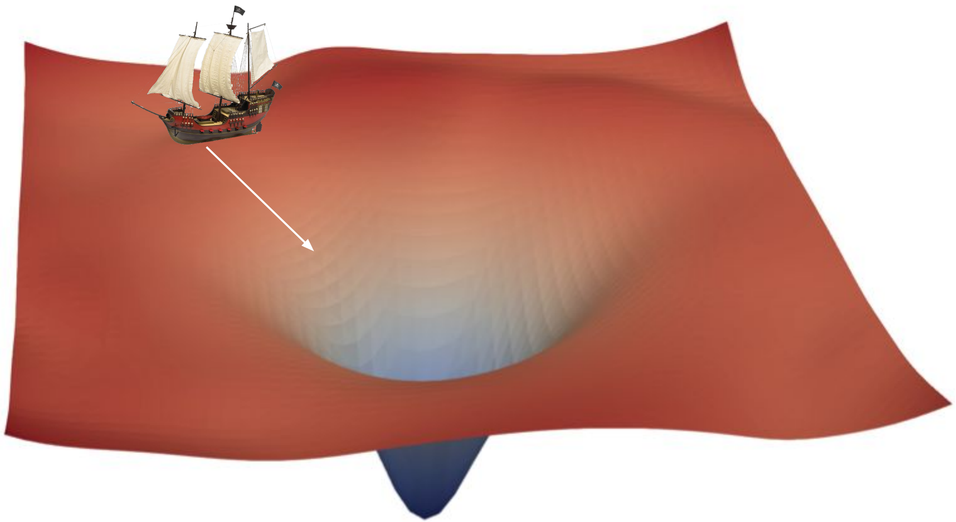
- Gradient Descent: Navigating through the space of possible models
  - Trying to find a "good" minimum for our loss function
- Loss Landscape: The environment which informs our journey



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

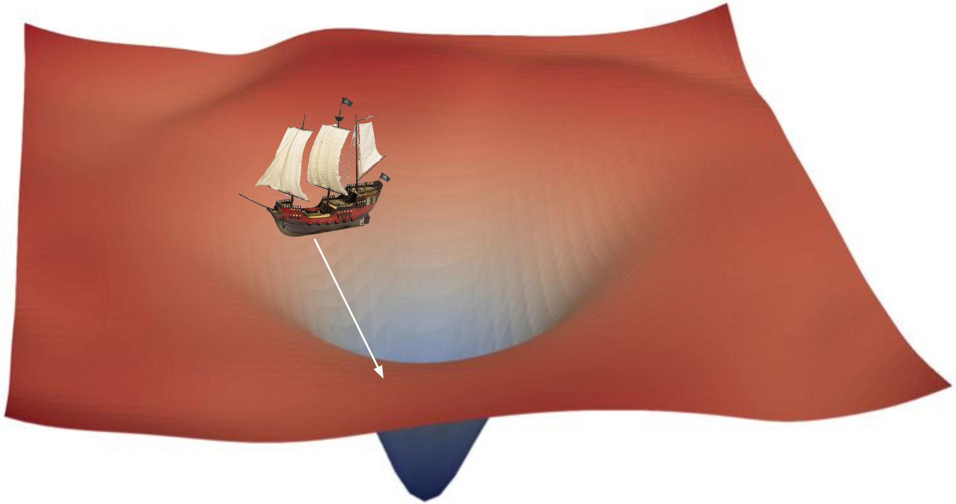
- Gradient Descent: Navigating through the space of possible models
    - Trying to find a “good” minimum for our loss function
- Loss Landscape: The environment which informs our journey



From *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

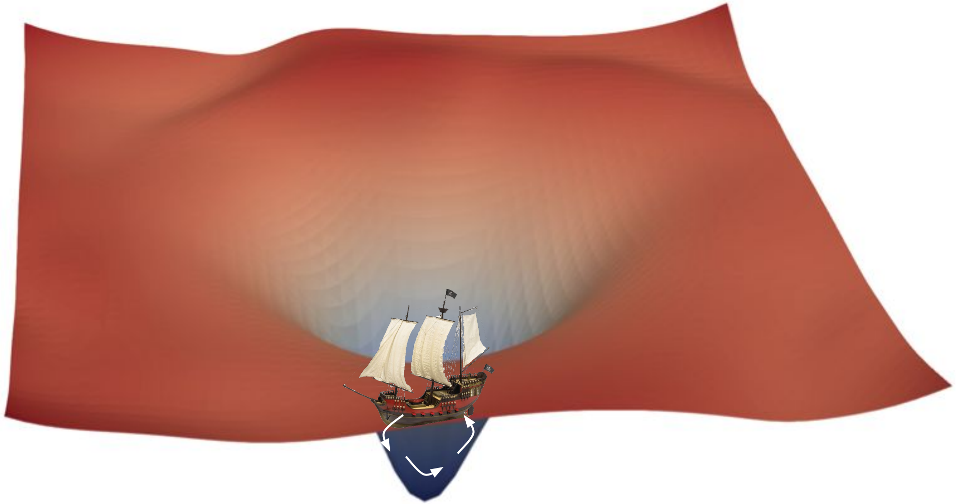
- Gradient Descent: Navigating through the space of possible models
    - Trying to find a “good” minimum for our loss function
- Loss Landscape: The environment which informs our journey



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- Gradient Descent: Navigating through the space of possible models
    - Trying to find a “good” minimum for our loss function
- Loss Landscape: The environment which informs our journey
    - Success!



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
  - Too flat

From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
  - Too flat

From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
  - Too flat

From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
    - Too flat

From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

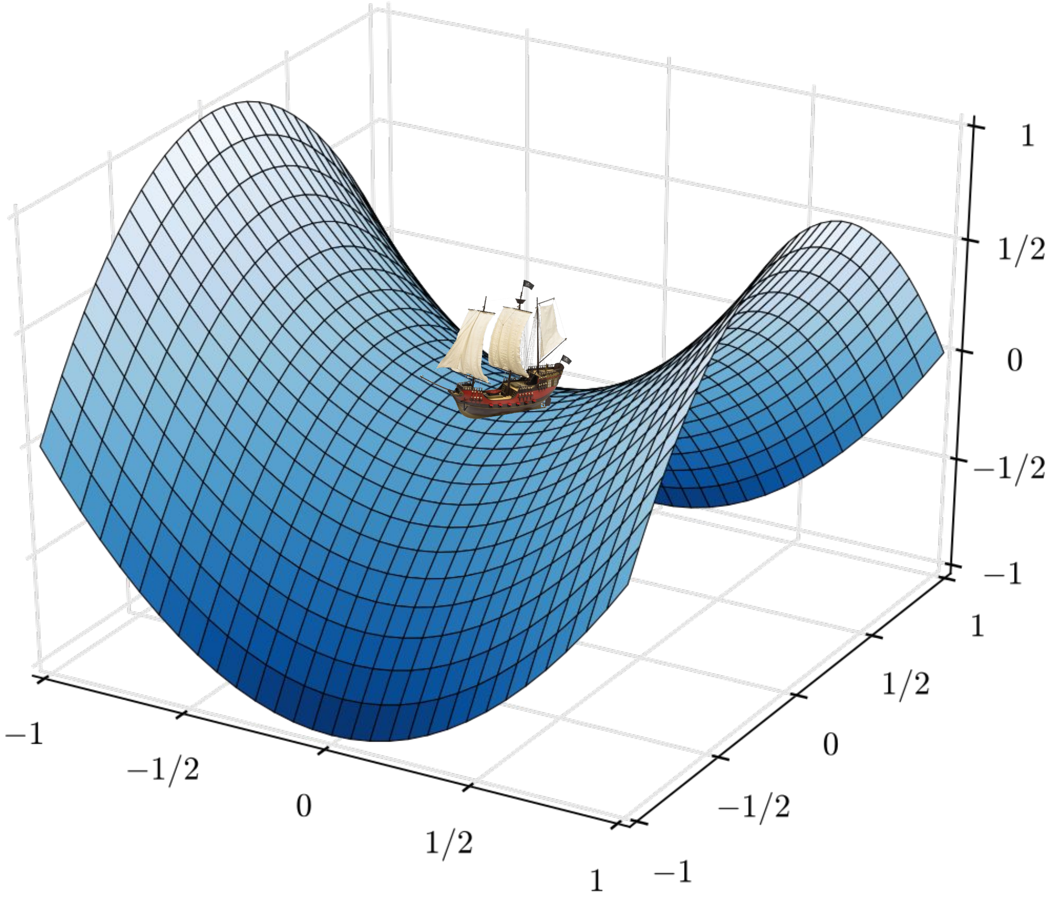
What does it mean to train a model?

- What can go wrong?
  - Too flat

From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

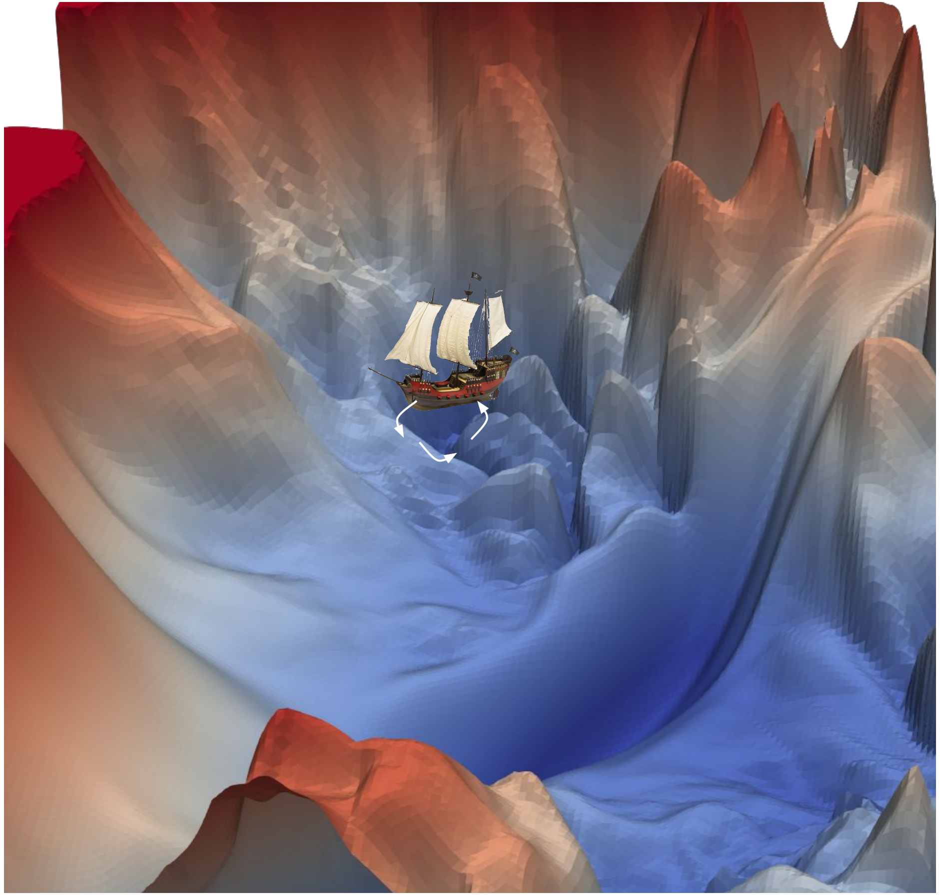
What does it mean to train a model?

- What can go wrong?
    - Too flat (or saddle point)



What does it mean to train a model?

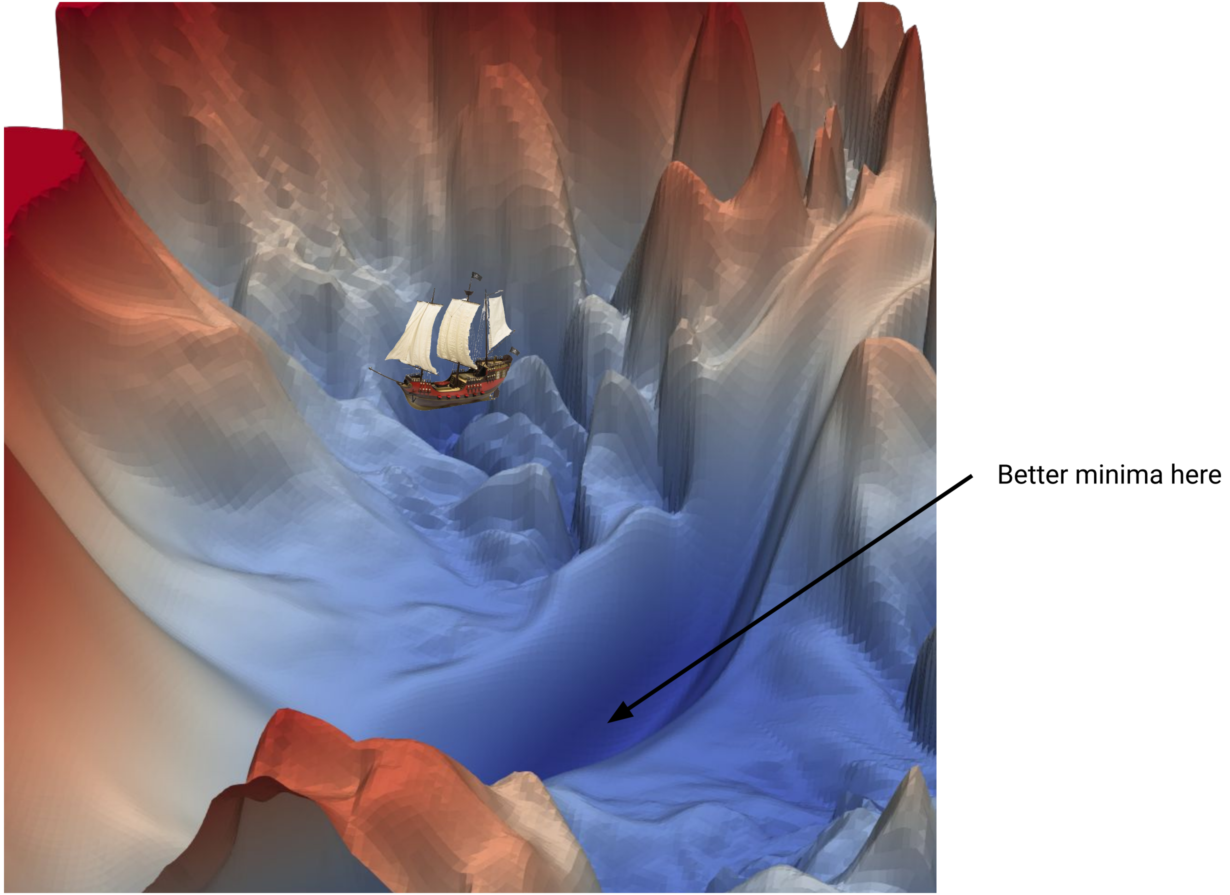
- What can go wrong?
    - Too flat
    - “Bad” minima



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
    - Too flat
    - “Bad” minima
- Shallow

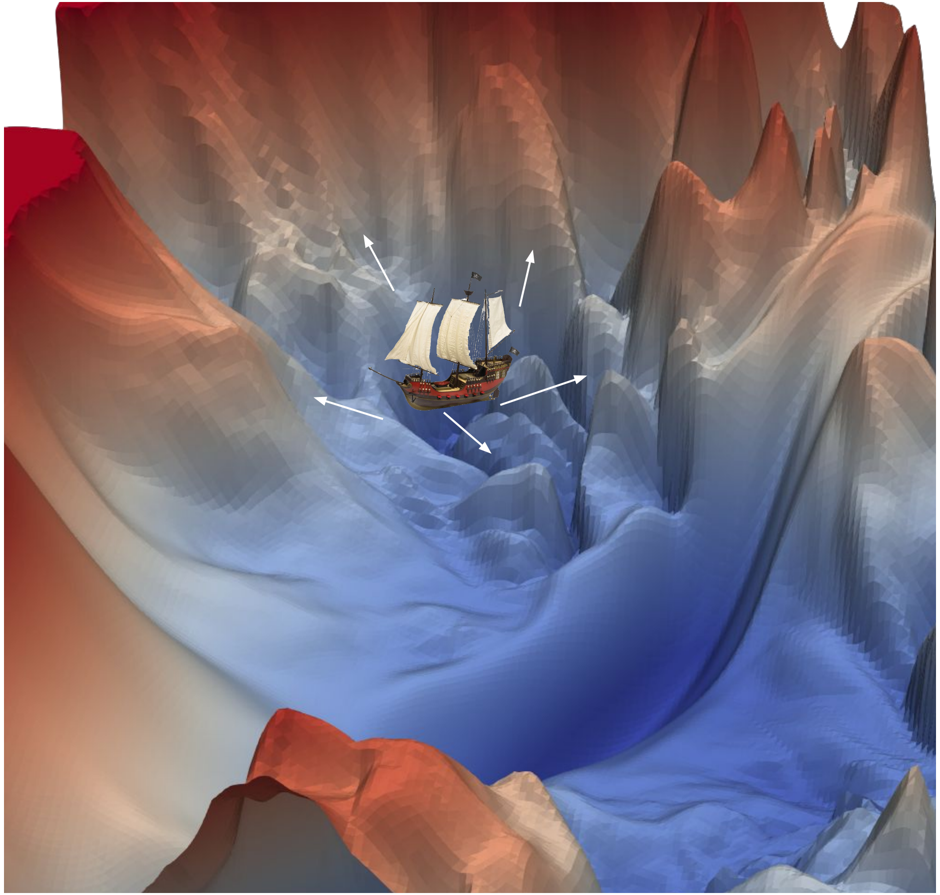


From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
  - Too flat
  - “Bad” minima

- Shallow
- Unstable

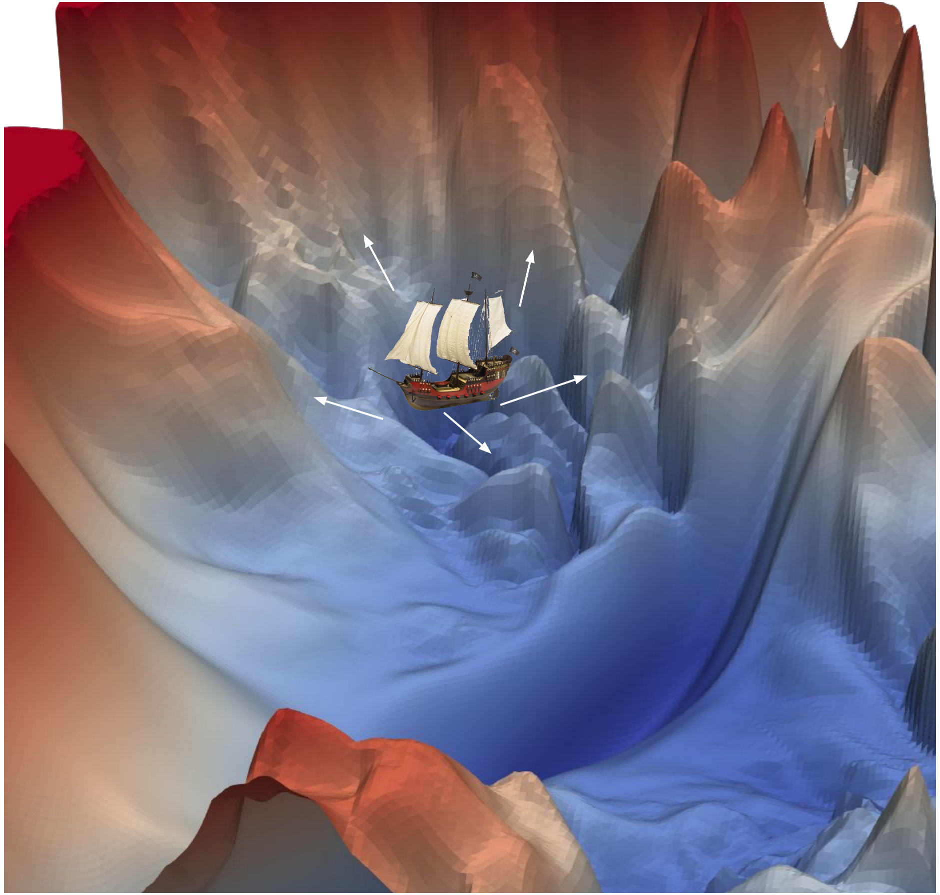


From *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Small movement may incur massive loss increase

What does it mean to train a model?

- What can go wrong?
  - Too flat
  - "Bad" minima
- Shallow
- Unstable
  - May not generalize well

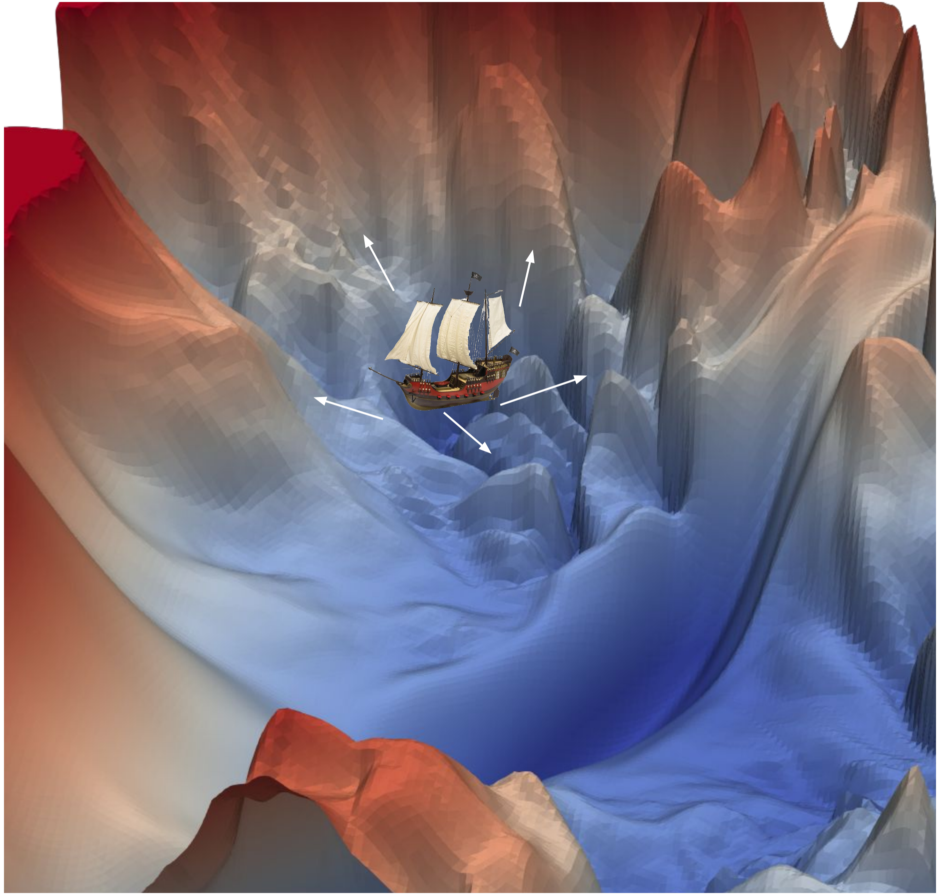


Small movement may incur massive loss increase

From *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

What does it mean to train a model?

- What can go wrong?
    - Too flat
    - “Bad” minima
- Shallow
- Unstable
    - May not generalize well
- May not be as big as a problem as previously thought!

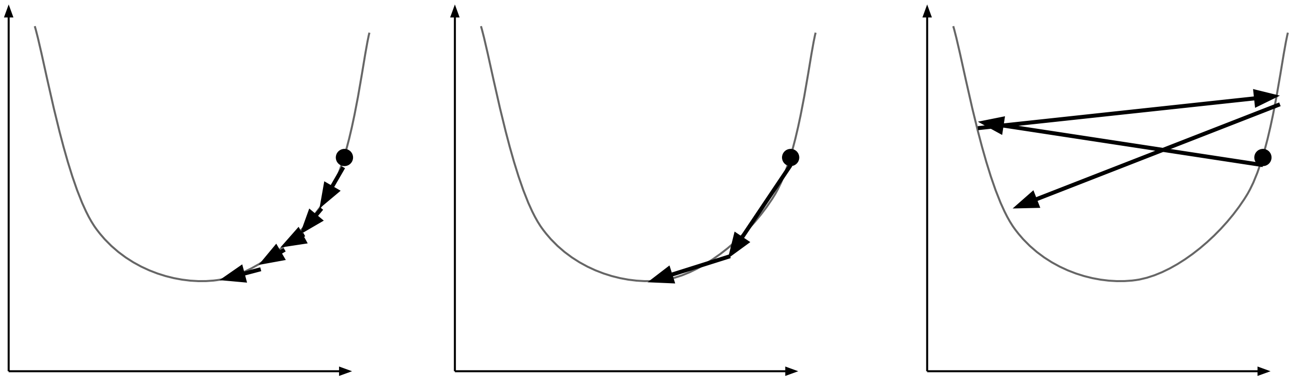


From *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Small movement may incur massive loss increase

Adjusting Learning Rates

- Choosing a learning rate is an art-form



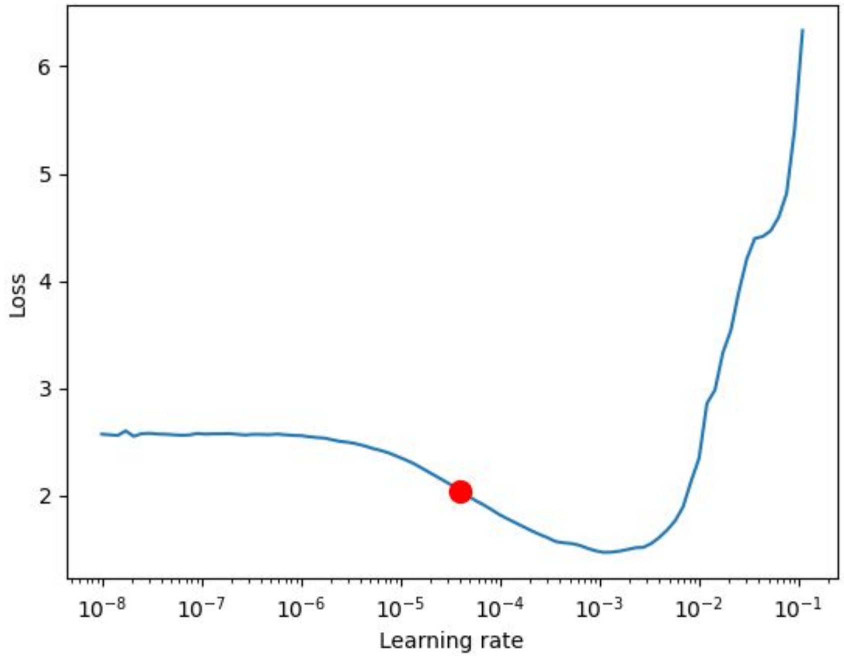
Low LR:
Too slow

Just right:
Good
convergence

High LR:
diverges

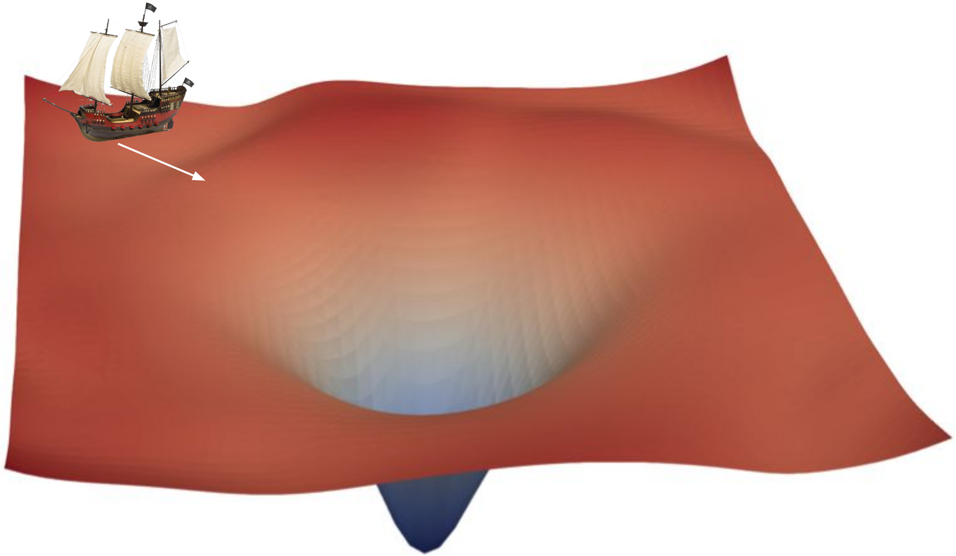
Adjusting Learning Rates

- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss



Adjusting Learning Rates

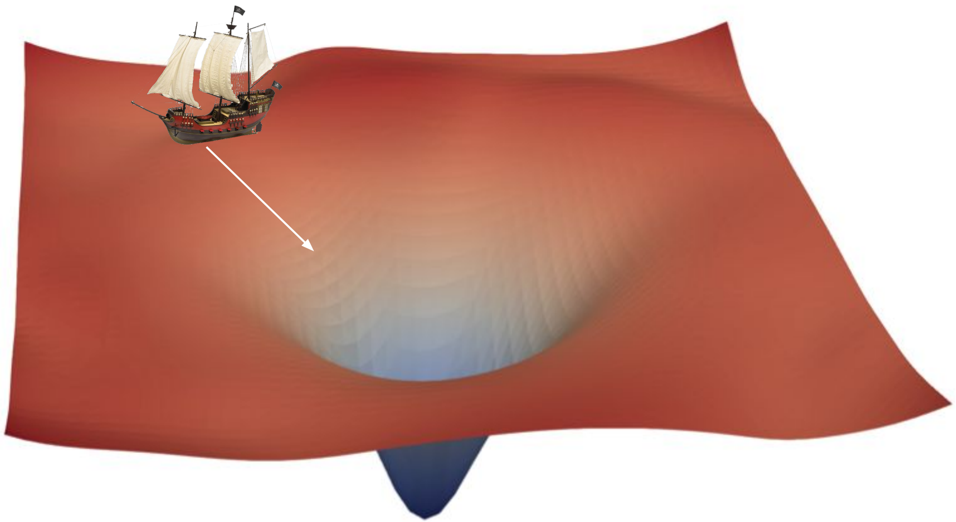
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

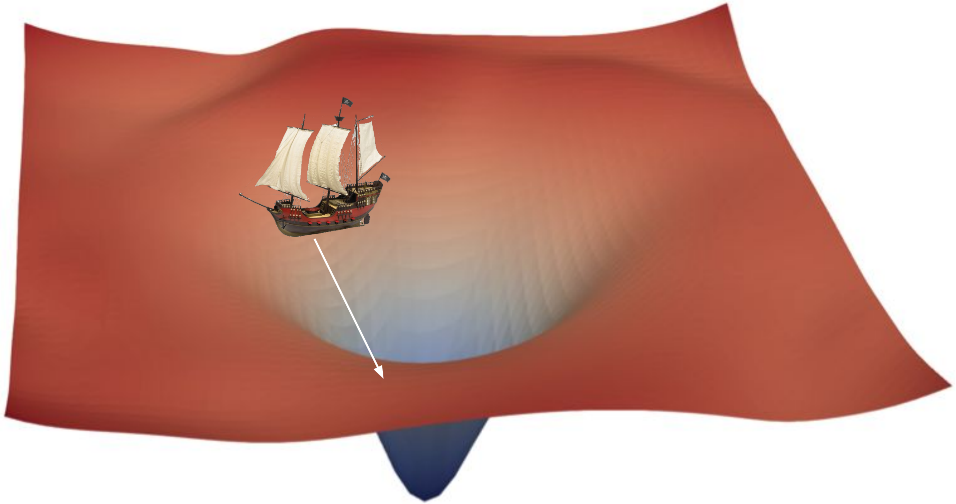
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

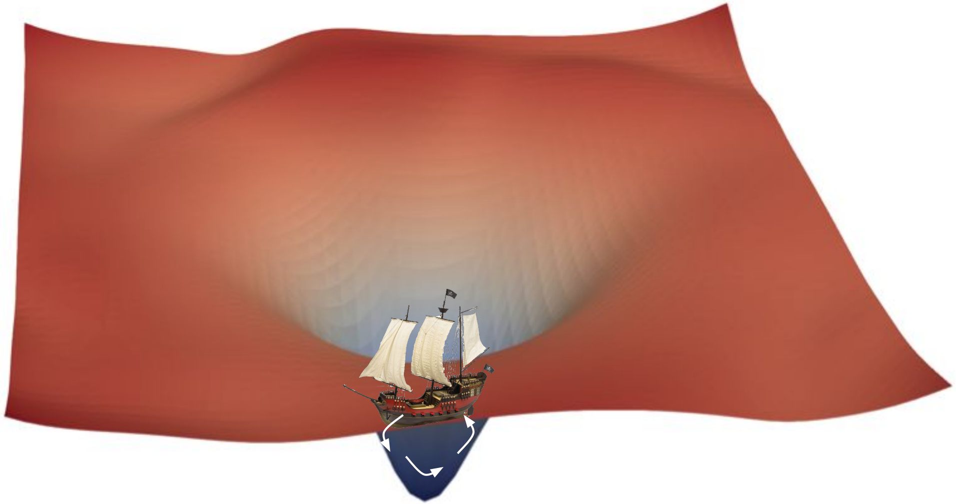
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

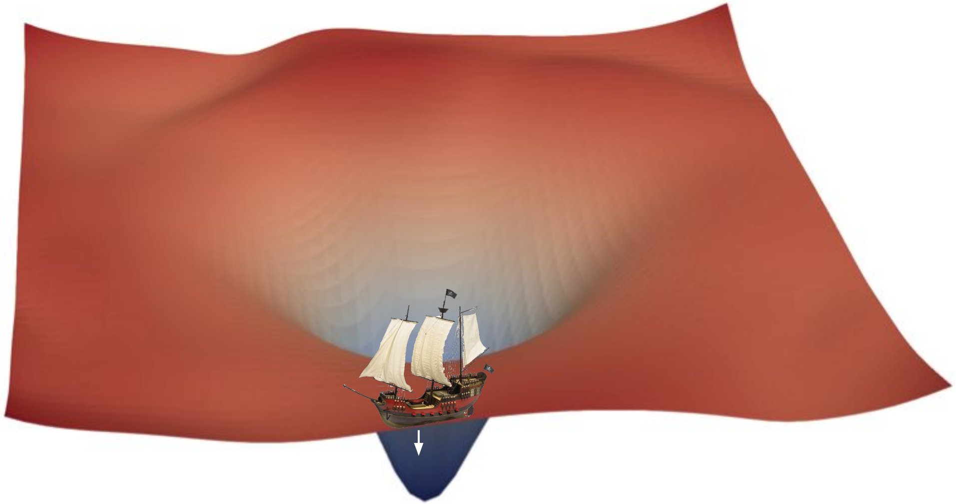
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

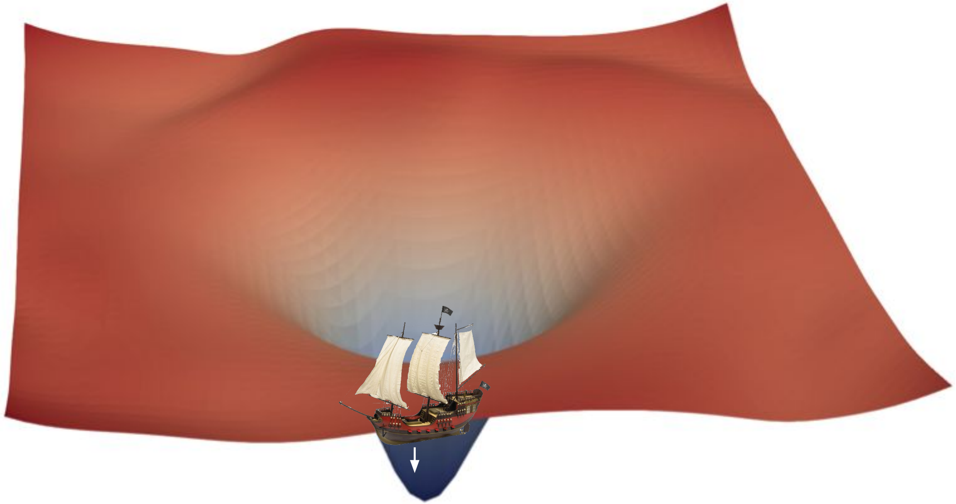
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

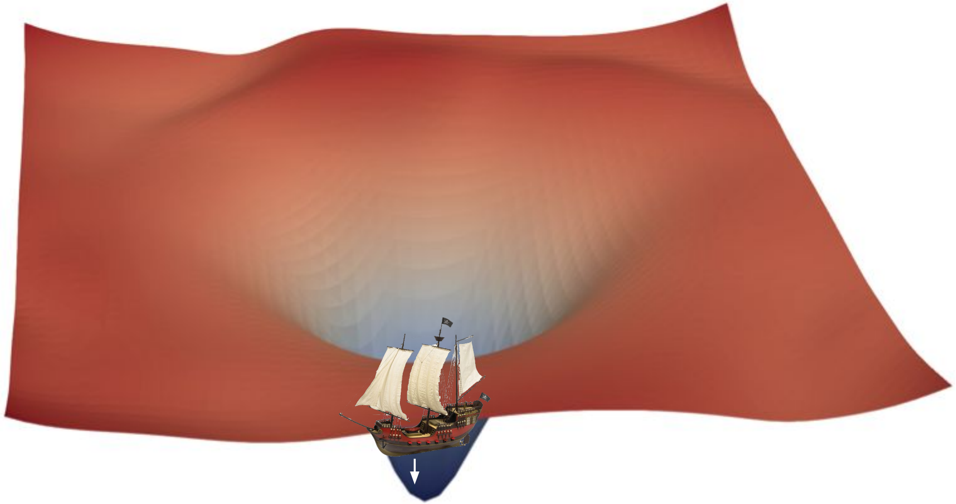
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

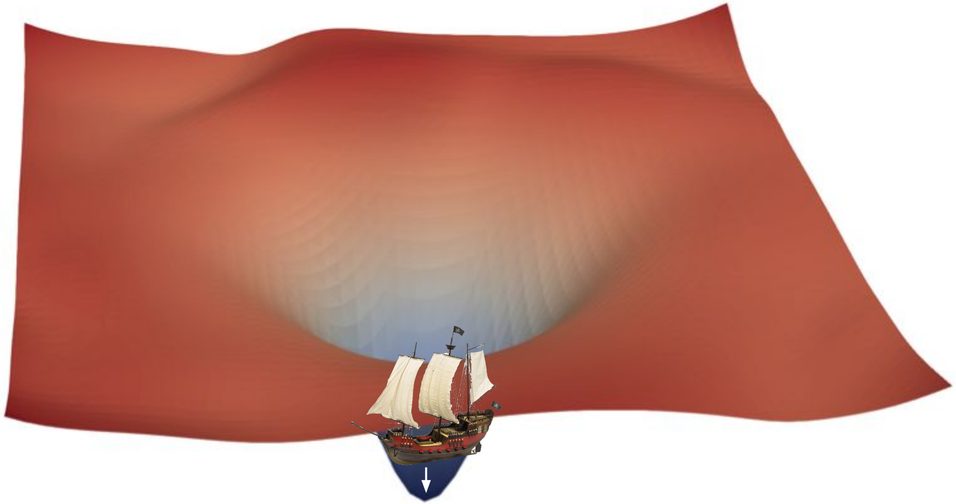
- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time



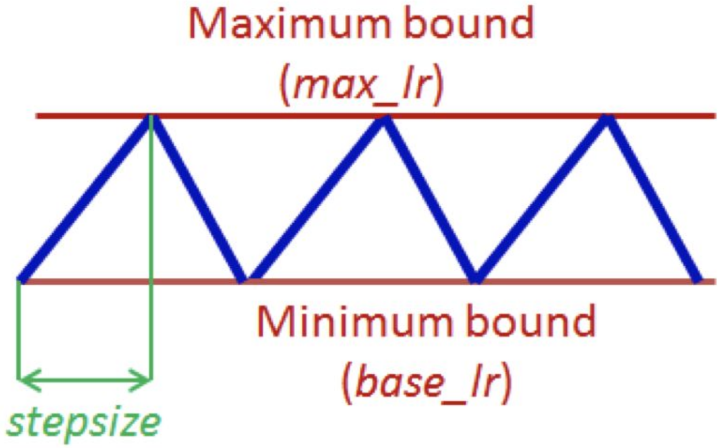
From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Adjusting Learning Rates

- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time
- Warm-up: low -> high -> low
    - Intuition is that warm-up improves stability

Adjusting Learning Rates

- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time
- Warm-up: low -> high -> low
    - Intuition is that warm-up improves stability
- Cyclical LRs



Cyclical Learning Rates for Training Neural Networks by Leslie Smith

Adjusting Learning Rates

- Choosing a learning rate is an art-form
- LR Test: Train while cranking up learning rate and plot loss
- Learning rate *annealing*: Start with High LR, then decrease over time
- Warm-up: low -> high -> low
    - Intuition is that warm-up improves stability
- Cyclical LRs
- And more!

Regularization

- There many techniques we can use to help prevent overfitting

Regularization

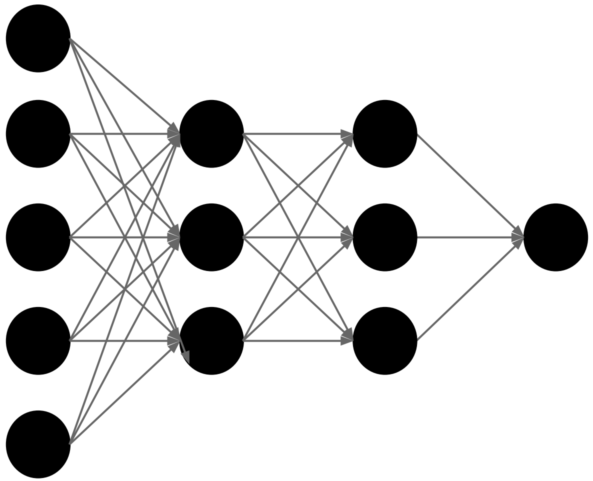
- There many techniques we can use to help prevent overfitting
- Early Stopping: Fit to training data, but not too much!

Regularization

- There many techniques we can use to help prevent overfitting
- Early Stopping: Fit to training data, but not too much!
- Weight Decay: Overfit models often have high magnitude coefficients

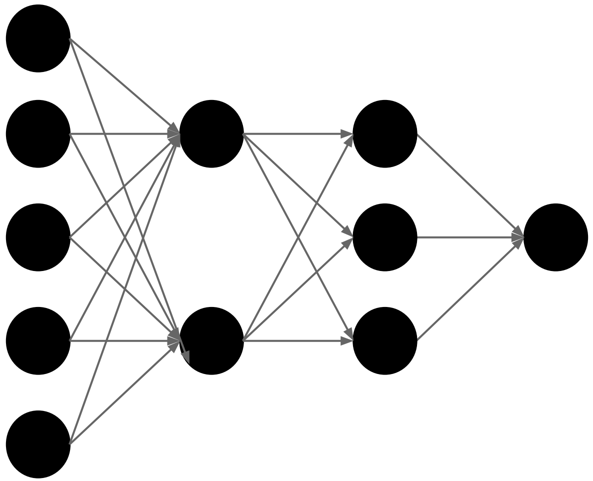
Regularization

- There many techniques we can use to help prevent overfitting
- Early Stopping: Fit to training data, but not too much!
- Weight Decay: Overfit models often have high magnitude coefficients
- Dropout: Reduces reliance on a single node/feature



Regularization

- There many techniques we can use to help prevent overfitting
- Early Stopping: Fit to training data, but not too much!
- Weight Decay: Overfit models often have high magnitude coefficients
- Dropout: Reduces reliance on a single node/feature



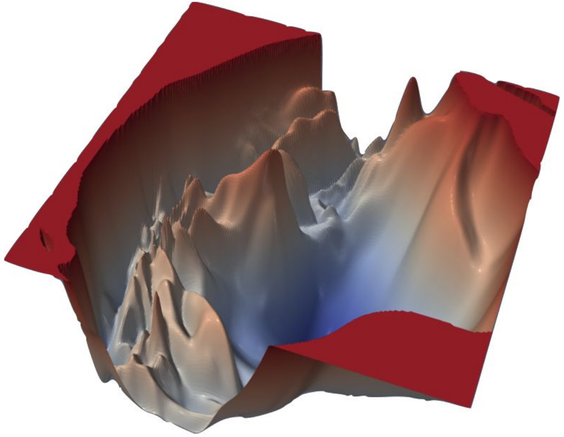
Regularization

- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
  - Only compute gradient with respect to a small batch of your data
  - Data might be too big to load onto GPU
  - Form of regularization (adds noise)
  - Model updates more frequently

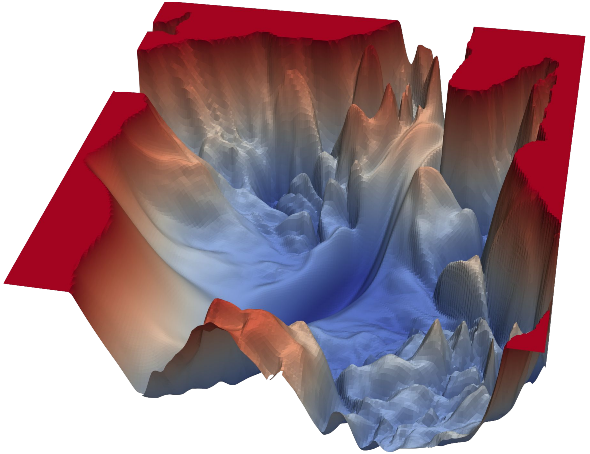
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

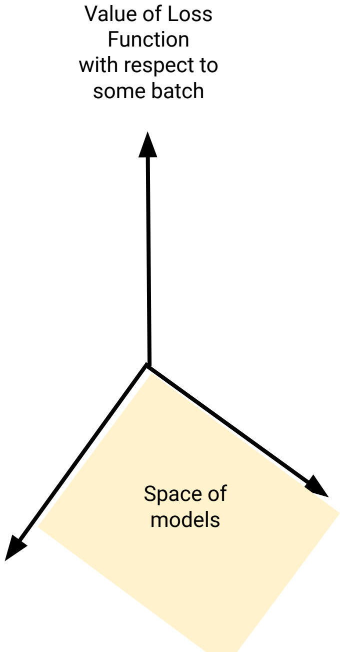
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
  - Only compute gradient with respect to a small batch of your data
  - Data might be too big to load onto GPU
  - Form of regularization (adds noise)
  - Model updates more frequently



Loss values for first batch



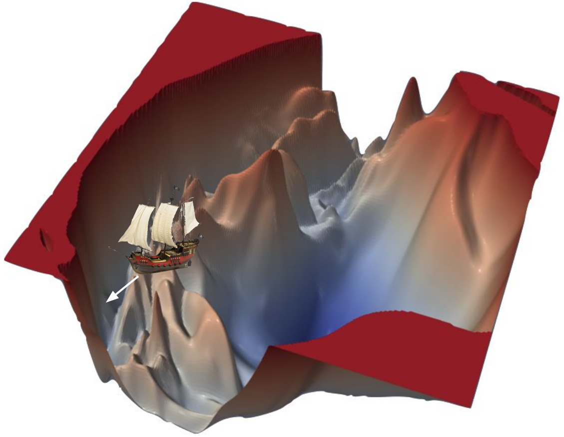
Loss values for second batch



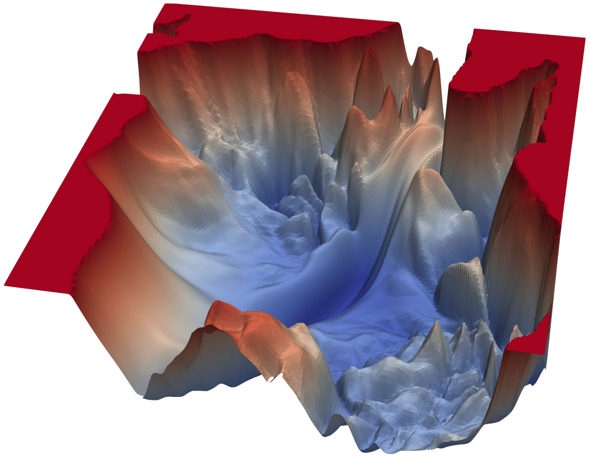
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

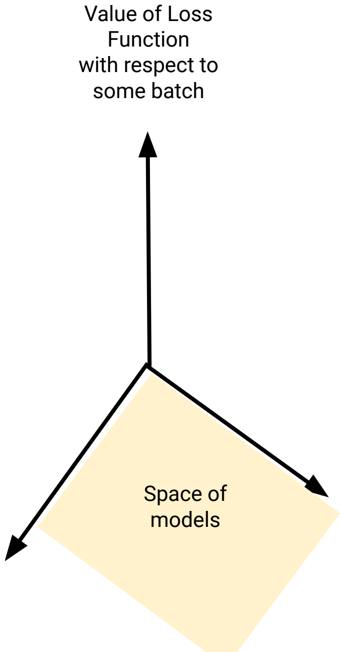
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
  - Only compute gradient with respect to a small batch of your data
  - Data might be too big to load onto GPU
  - Form of regularization (adds noise)
  - Model updates more frequently



Loss values for first batch



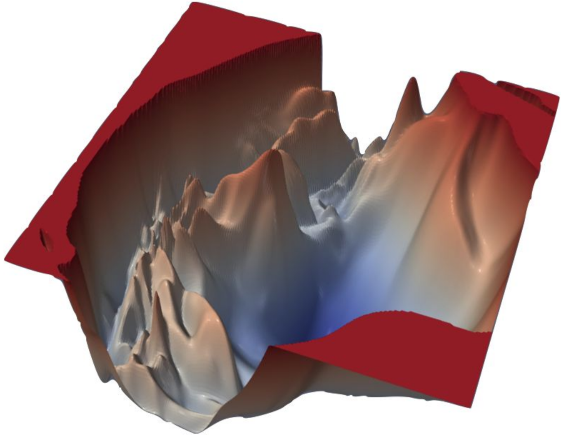
Loss values for second batch



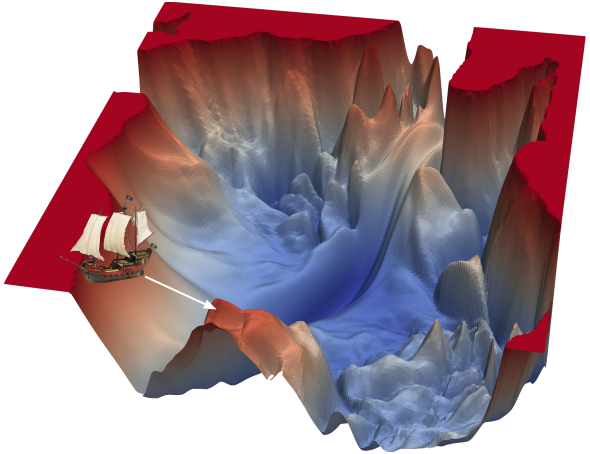
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

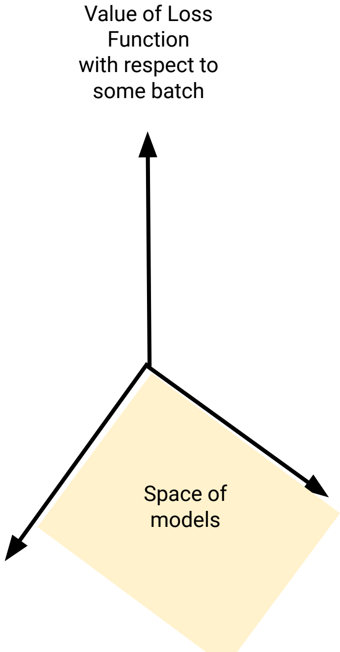
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
    - Only compute gradient with respect to a small batch of your data
    - Data might be too big to load onto GPU
    - Form of regularization (adds noise)
    - Model updates more frequently



Loss values for first batch



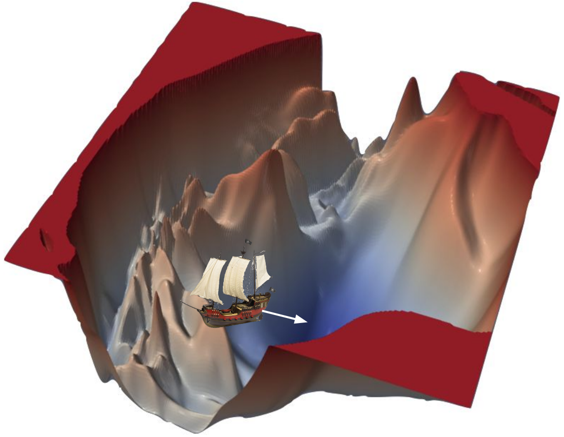
Loss values for second batch



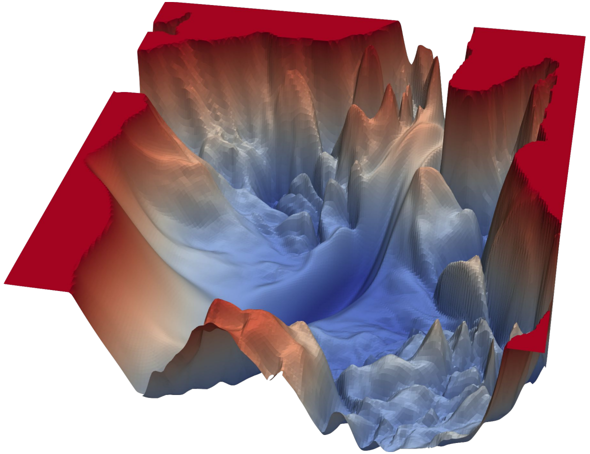
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

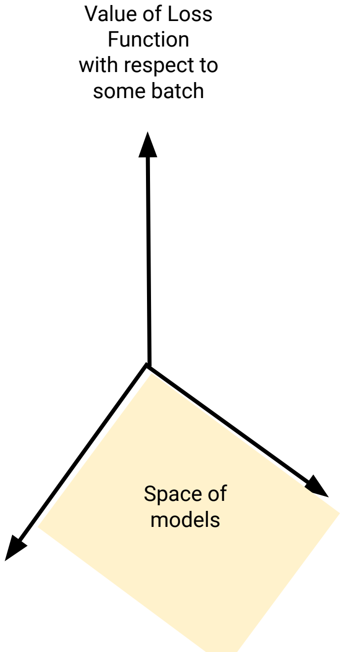
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
    - Only compute gradient with respect to a small batch of your data
    - Data might be too big to load onto GPU
    - Form of regularization (adds noise)
    - Model updates more frequently



Loss values for first batch



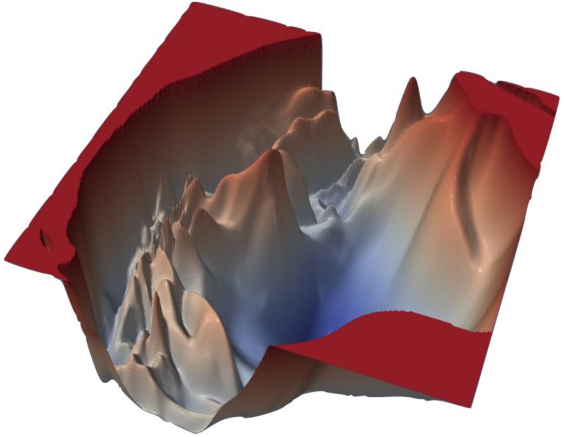
Loss values for second batch



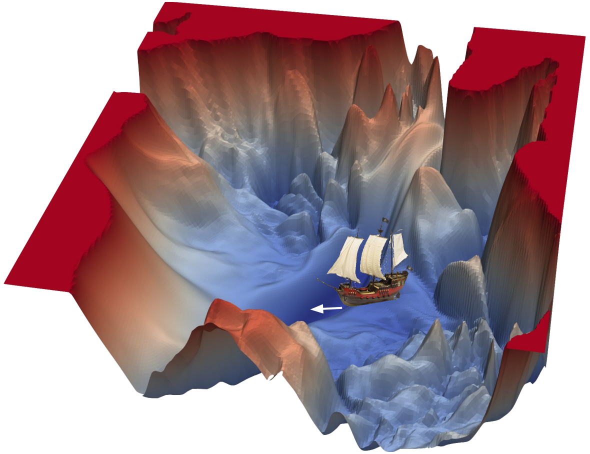
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

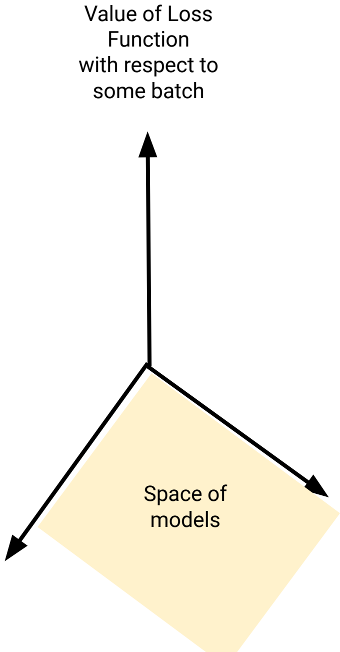
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
  - Only compute gradient with respect to a small batch of your data
  - Data might be too big to load onto GPU
  - Form of regularization (adds noise)
  - Model updates more frequently



Loss values for first batch



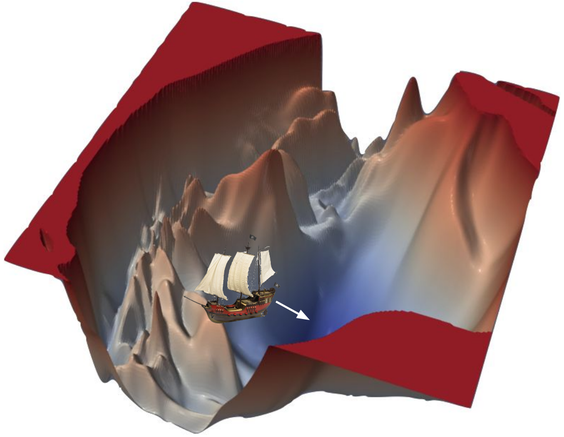
Loss values for second batch



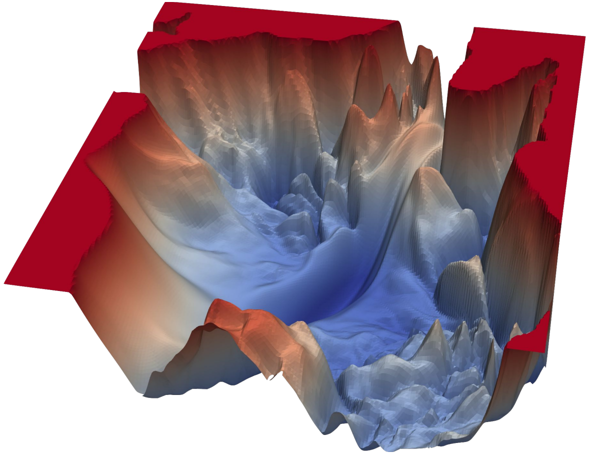
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

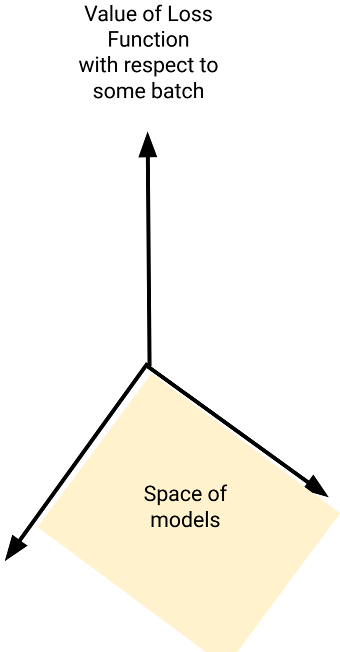
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
  - Only compute gradient with respect to a small batch of your data
  - Data might be too big to load onto GPU
  - Form of regularization (adds noise)
  - Model updates more frequently



Loss values for first batch



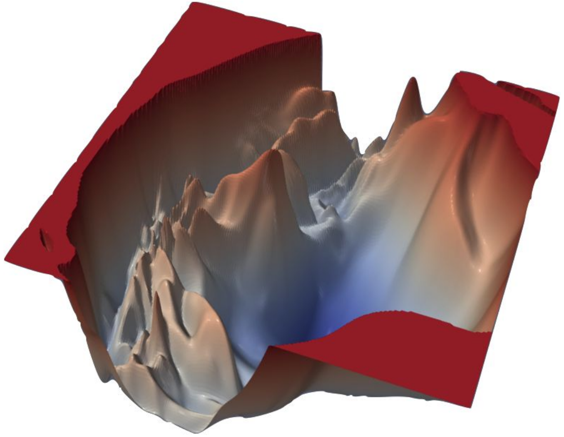
Loss values for second batch



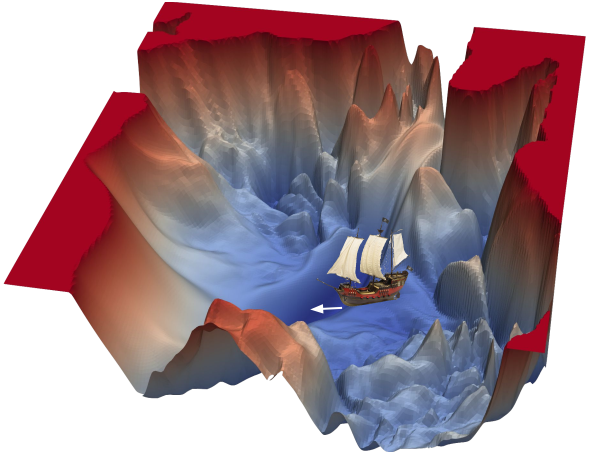
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

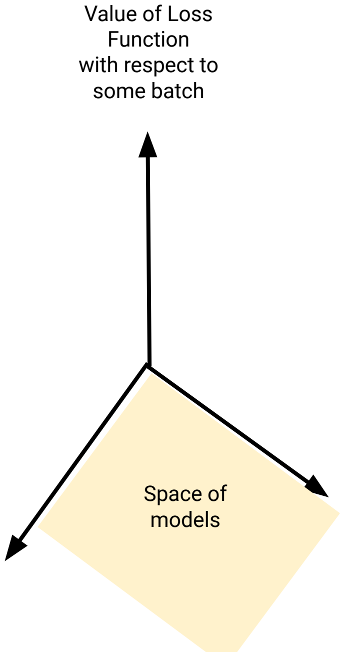
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
  - Only compute gradient with respect to a small batch of your data
  - Data might be too big to load onto GPU
  - Form of regularization (adds noise)
  - Model updates more frequently



Loss values for first batch



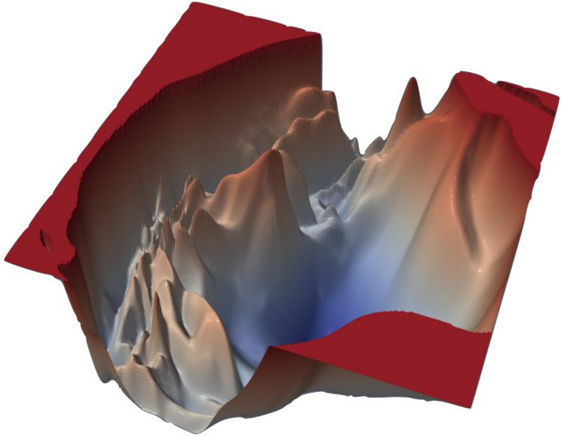
Loss values for second batch



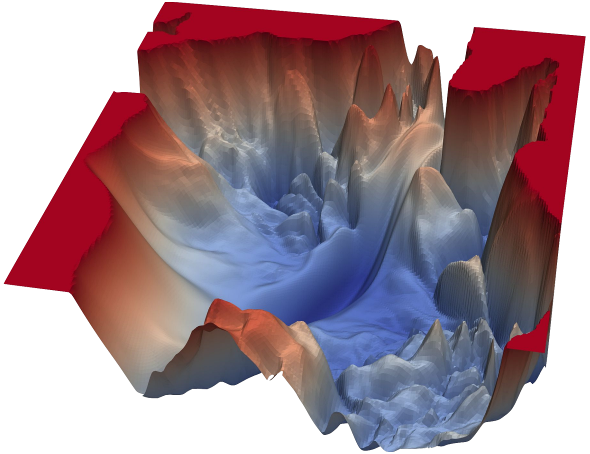
Regularization

Pictures from *Visualizing the Loss Landscape of Neural Nets* by H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

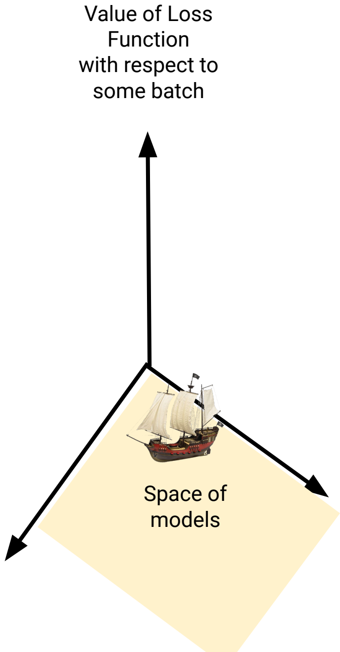
- We usually train NN using *mini-batches*. (not a fixed loss landscape!)
    - Only compute gradient with respect to a small batch of your data
    - Data might be too big to load onto GPU
    - Form of regularization (adds noise)
    - Model updates more frequently



Loss values for first batch



Loss values for second batch



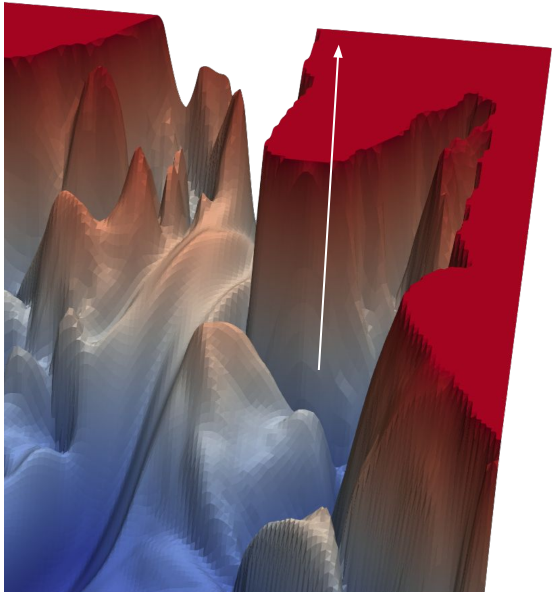
Normalization

- Normalization of continuous variables can be extremely helpful for optimization, particularly for deep neural networks
  - Puts features on a similar scale
  - Potentially avoid vanishing/exploding gradients

$$\bar{x} = \frac{x - \mu}{\sigma}$$

Normalization

- Vanishing/exploding gradients can become even worse in a deep network
    - Think chain rule



From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

Normalization

- Vanishing/exploding gradients can become even worse in a deep network
  - Think chain rule
- *Batch normalization* normalizes, in some sense, the output of each layer
  - "Normalizing along the way"

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Output of some hidden layer we will apply batch normalization to

Normalization

- Vanishing/exploding gradients can become even worse in a deep network
  - Think chain rule
- *Batch normalization* normalizes, in some sense, the output of each layer
  - "Normalizing along the way"

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \overline{x} = \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_n \end{bmatrix} \qquad \overline{x}_i = \frac{x_i - \mu_i}{\sigma + \epsilon}$$

Normalization

- Vanishing/exploding gradients can become even worse in a deep network
  - Think chain rule
- *Batch normalization* normalizes, in some sense, the output of each layer
  - “Normalizing along the way”

Normalize over the mini-batch

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \overline{x} = \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_n \end{bmatrix} \qquad \overline{x}_i = \frac{x_i - \mu_i}{\sigma + \epsilon}$$

Normalization

- Vanishing/exploding gradients can become even worse in a deep network
  - Think chain rule
- *Batch normalization* normalizes, in some sense, the output of each layer
  - "Normalizing along the way"

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \overline{x} = \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_n \end{bmatrix} \qquad \gamma\overline{x} + \beta = \begin{bmatrix} \gamma_1 \overline{x}_1 + \beta_1 \\ \gamma_2 \overline{x}_2 + \beta_2 \\ \vdots \\ \gamma_n \overline{x}_n + \beta_n \end{bmatrix}$$

Normalization

- Vanishing/exploding gradients can become even worse in a deep network
  - Think chain rule
- *Batch normalization* normalizes, in some sense, the output of each layer
  - "Normalizing along the way"

Gamma, beta
initialize as all ones
and zeros vectors
respectively!

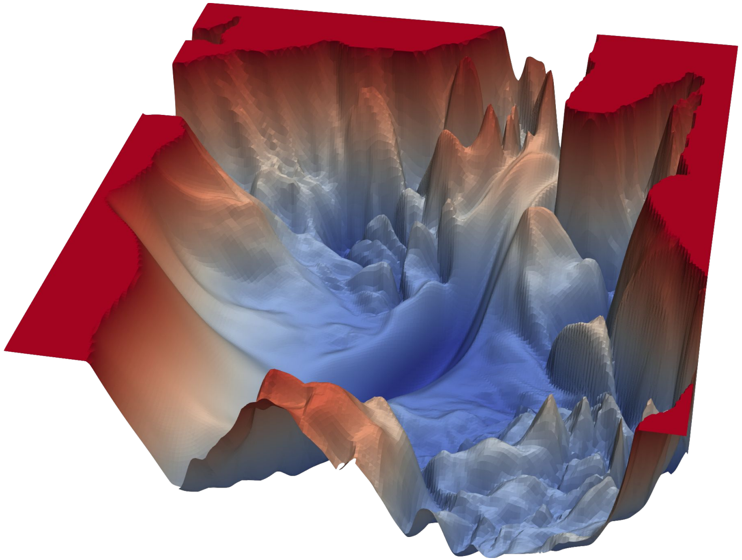
$$\gamma\overline{x} + \beta = \begin{bmatrix} \gamma_1 \overline{x}_1 + \beta_1 \\ \gamma_2 \overline{x}_2 + \beta_2 \\ \vdots \\ \gamma_n \overline{x}_n + \beta_n \end{bmatrix}$$

Smoothing the loss landscape

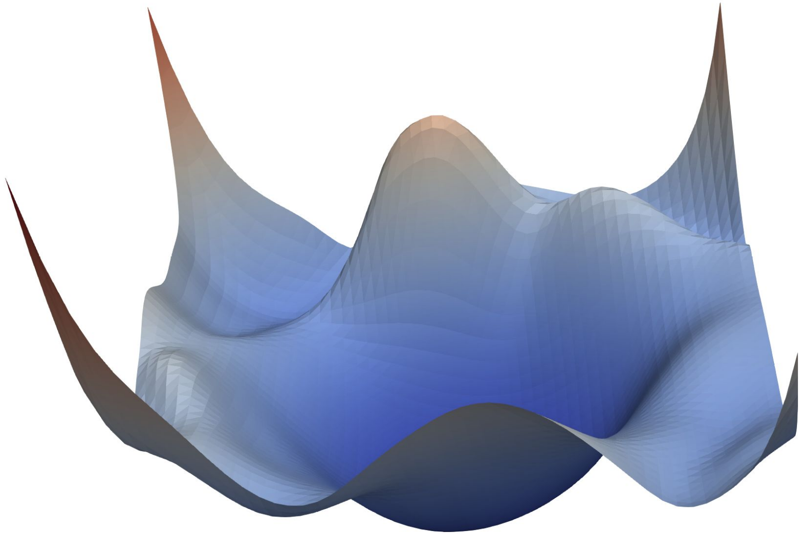
- Intuition: making the loss landscape easier to traverse

Smoothing the loss landscape

- Intuition: making the loss landscape easier to traverse
- Skip Connections (more on these later)



w/o skips



w/ skips

From *Visualizing the Loss Landscape of Neural Nets* by
H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein

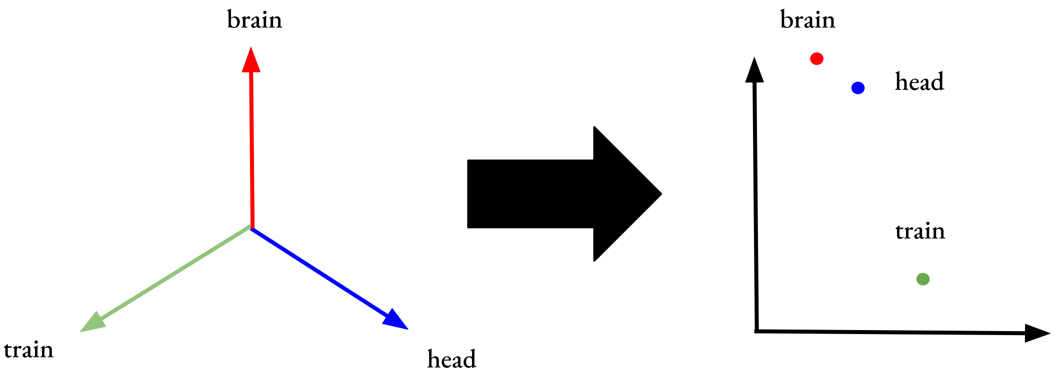
Other things to tweak

- Tons of different optimization algorithms
  - RMSprop
  - Adam
  - AdamW (adam with weight decay)
  - Adadelta
- Different methods for weight initialization
  - Idea: better/more stable starting points
- Change batch size
  - Spectrum from stochastic to one batch
  - Smaller batches usually results in noisier training

Embeddings

- For categorical variables we often use an *embedding* as a first step
- Categorical values can be one-hot encoded (meaning agnostic) then embedded into a meaningful feature space
- Similar to word embeddings
    - Go from one-hot encoded dictionary to word vectors

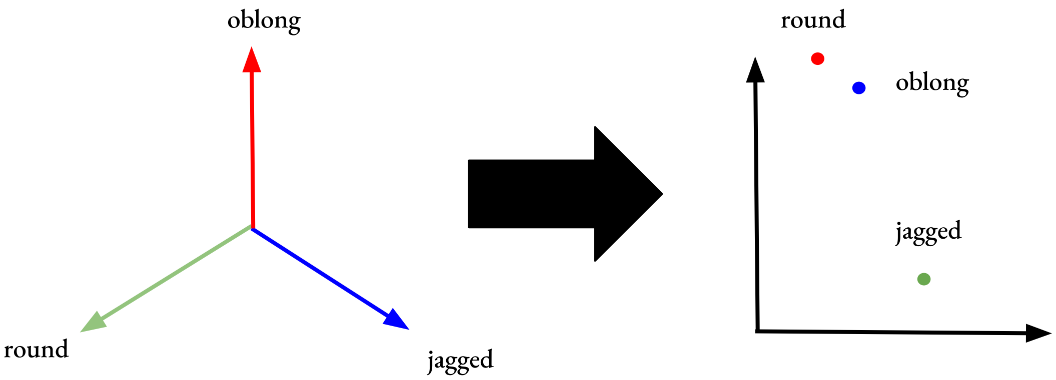
$$\begin{array}{ccc} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \text{brain} & \text{head} & \text{train} \end{array}$$



Embeddings

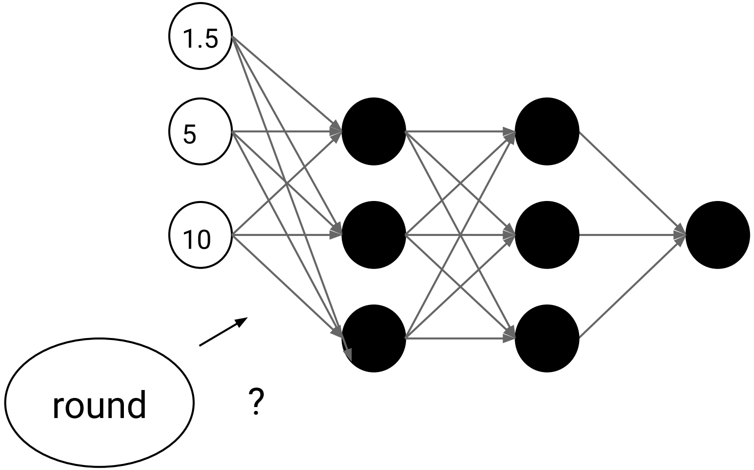
- For categorical variables we often use an *embedding* as a first step
- Categorical values can be one-hot encoded (meaning agnostic) then embedded into a meaningful feature space
- Doesn't have to be words
    - Go from one-hot encoded possible values to feature vectors

$$\begin{array}{ccc} \begin{bmatrix}1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\end{bmatrix} \\ \text{round} & \text{oblong} & \text{jagged} \end{array}$$



Embeddings

- Suppose you have a mix of numerical and categorical variables for your input layer: x = [1, .5, 10, round]



Embeddings

- Suppose you have a mix of numerical and categorical variables for your input layer: x = [1, .5, 10, round]

One hot encoding

$$\text{round} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Embeddings

- Suppose you have a mix of numerical and categorical variables for your input layer: x = [1, .5, 10, round]

One hot encoding

Embedding matrix

$$\text{round} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Embeddings

- Suppose you have a mix of numerical and categorical variables for your input layer: x = [1, .5, 10, round]

One hot encoding

$$\text{round} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Embedding matrix

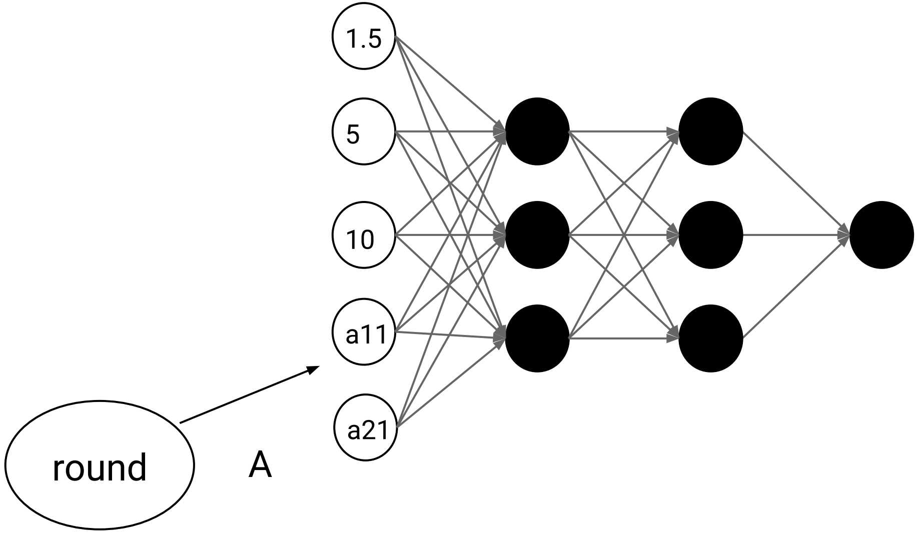
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Embedding of round

Embeddings

x = [1, .5, 10, round]

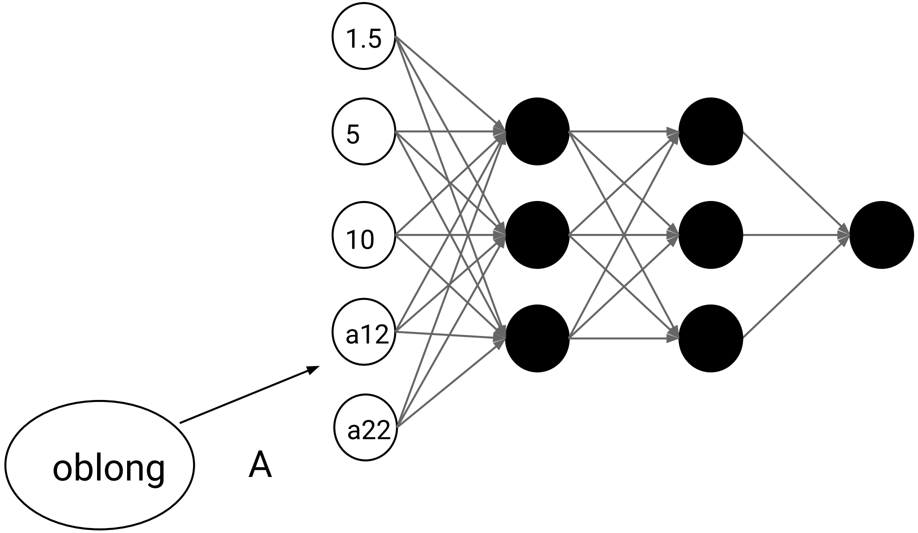
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$



Embeddings

x = [1, .5, 10, oblong]

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$



Why this or that architecture for a given problem?

- What architecture you use and other hyperparameters you choose depend heavily on
  - The task
  - Your available computing problem
  - How your model will be used
  - How interpretable you want your model to be

Why this or that architecture for a given problem?

- What architecture you use and other hyperparameters you choose depend heavily on
  - The task
  - Your available computing problem
  - How your model will be used
  - How interpretable you want your model to be
- The answer to “how many layers” or “how many nodes” is usually determined by
  - What other people have had success with
  - Your own experiments with different architectures