WELCOME!

Please sign in here:

https://tinyurl.com/y7yh9lx4

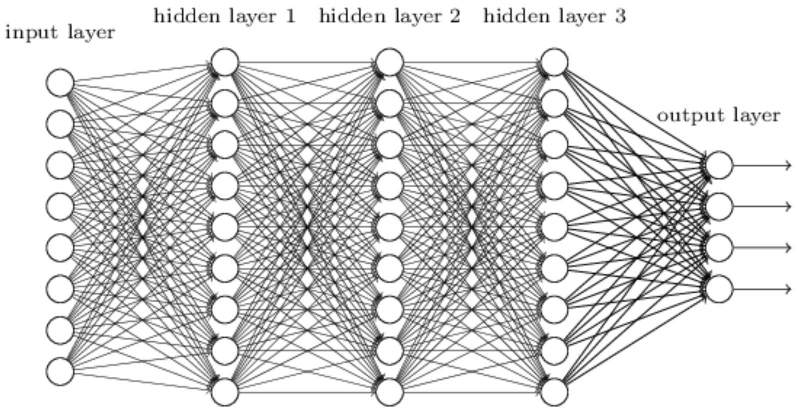
Regularization

BOSTON UNIVERSITY
MACHINE INTELLIGENCE
COMMUNITY

Chloe Kaubisch, Justin Chen
Oct. 10, 2017

Review

- Recall **neural networks**: layers of nodes capable of modeling high-dimensional data
- We use a **cost function** to compute our error, and **gradient descent** to update our weights accordingly
- **“Deep”** neural networks simply consist of more hidden layers

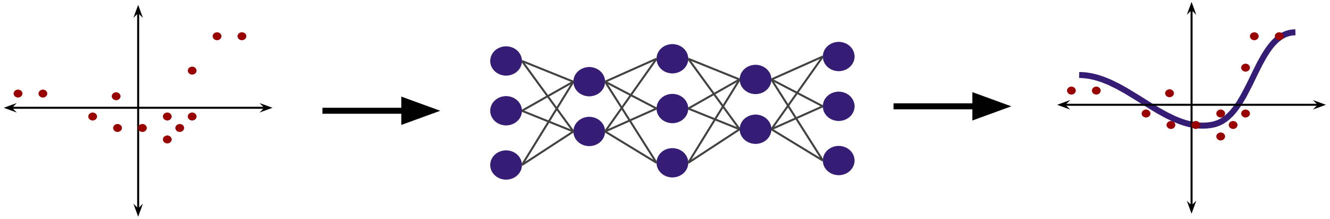


Overview:

- New Concepts:
    - Generalization
    - Data & Complexity
    - Bias & Variance
- Regularization
    - Weight Decay
    - Early Stopping
    - Dropout Layer
    - Dataset Augmentation

Generalization: the goal of machine learning

- **GENERALIZATION:** the ability of an algorithm to perform well on data it was not trained on
- **Key Questions:**
  - When and why does an algorithm fail to generalize?
  - How can this be addressed?



Dataset

Trained learning algorithm

Approximate function to generalize to new data

MIC

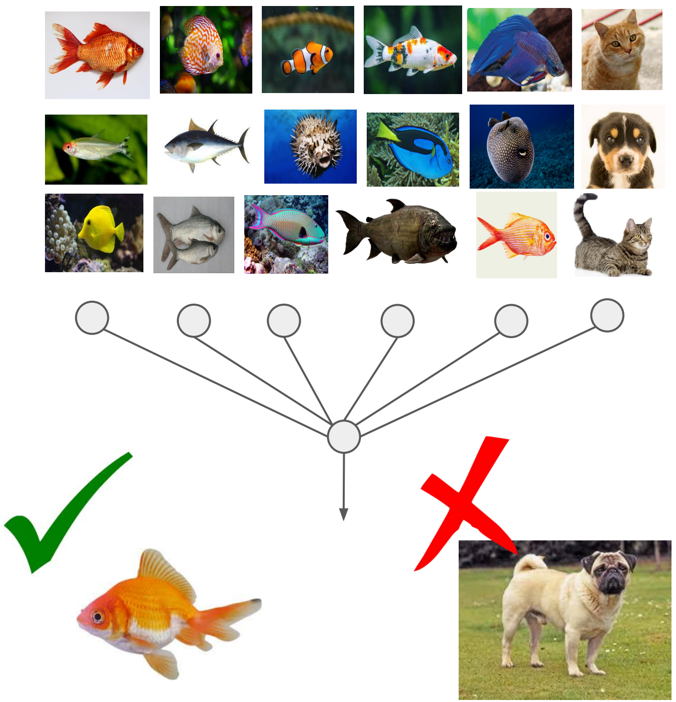
Dataset

Split - percentages depend on available data

Balance - data distribution close to uniform

M I C

Dataset Bias

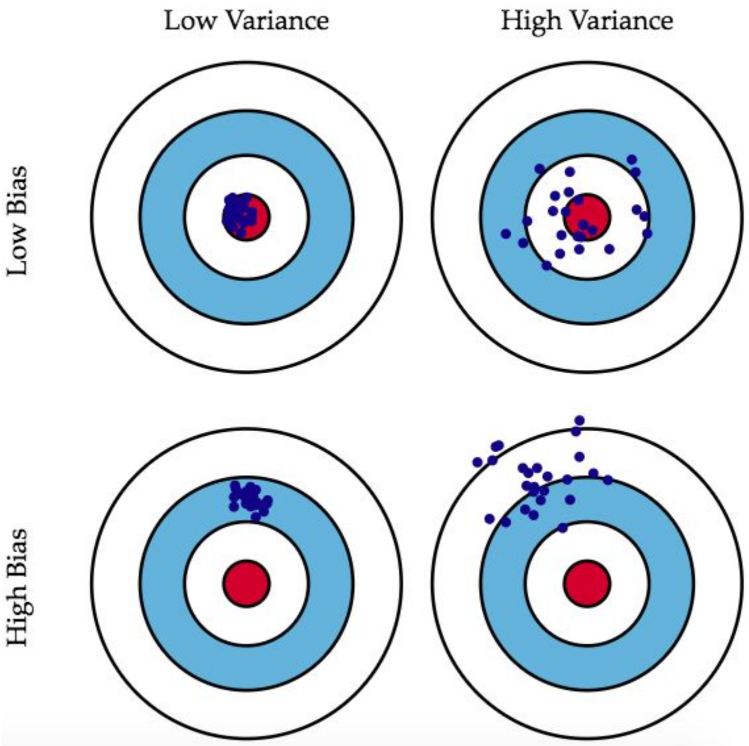


Src: @jackyalcine // https://www.dailymail.co.uk/sciencetech/article-3145887/Google-apologises-Photos-app-tags-black-people-gorillas-Fault-image-recognition-software-mislabelled-picture.html

M I C

Bias and Variance

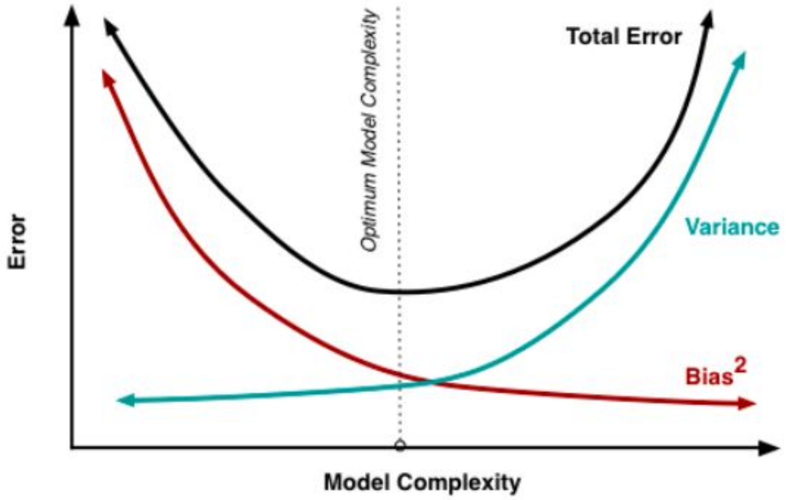
- **Bias:** How far are your results from the true values, e.g. how **inaccurate?**
  - Can indicate a lack of complexity in your model
- **Variance:** How dissimilar are your results from each other, e.g. how **inconsistent**?
  - Spread of your results across a given area
  - Can indicate excessive complexity in your model



M I C

Bias & Variance Tradeoff

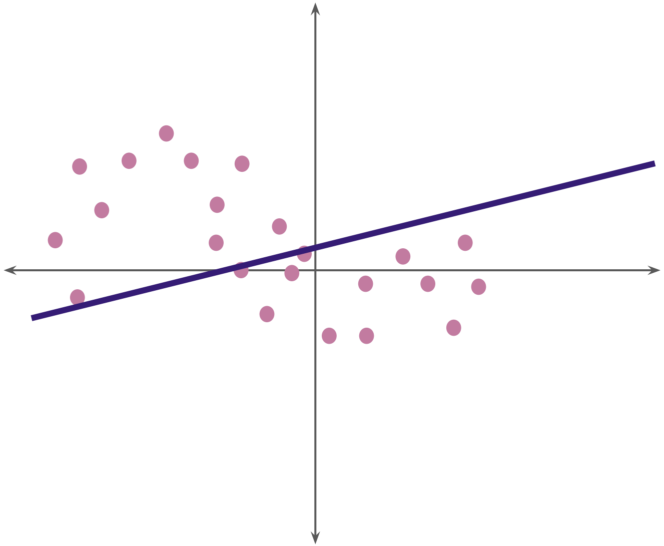
- Bias and variance are opposite problems that both lead to a failure to **generalize**
- Must minimize **total error**
- However, bias causes a model to **underfit** and variance causes a model to **overfit** - the difficulty is striking a balance

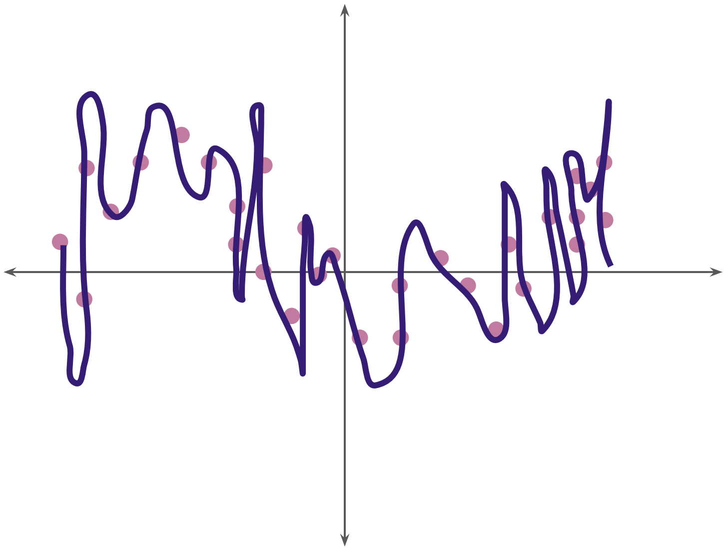


M I C

What is underfitting and overfitting?

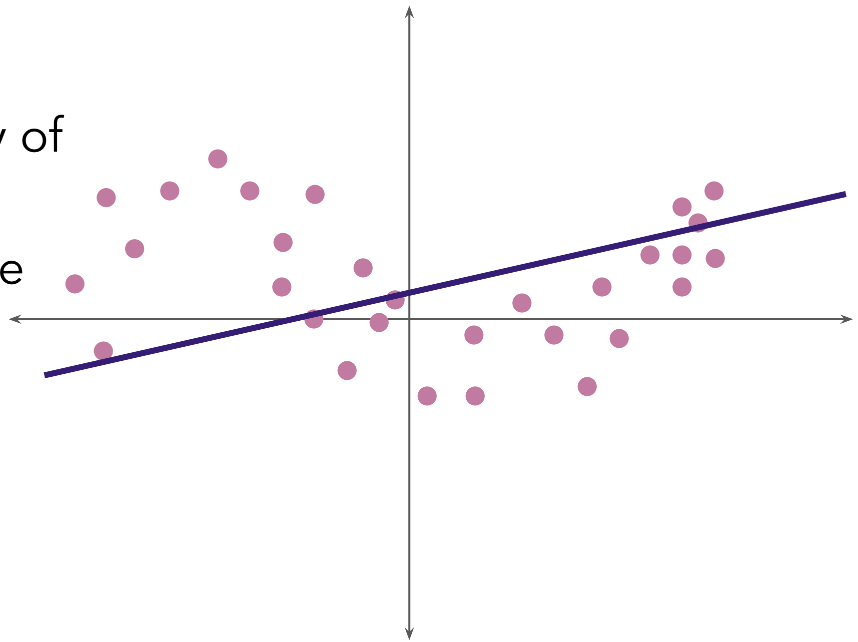
- Underfitting and overfitting occur when an algorithm learns but fails to **generalize**.





Underfitting

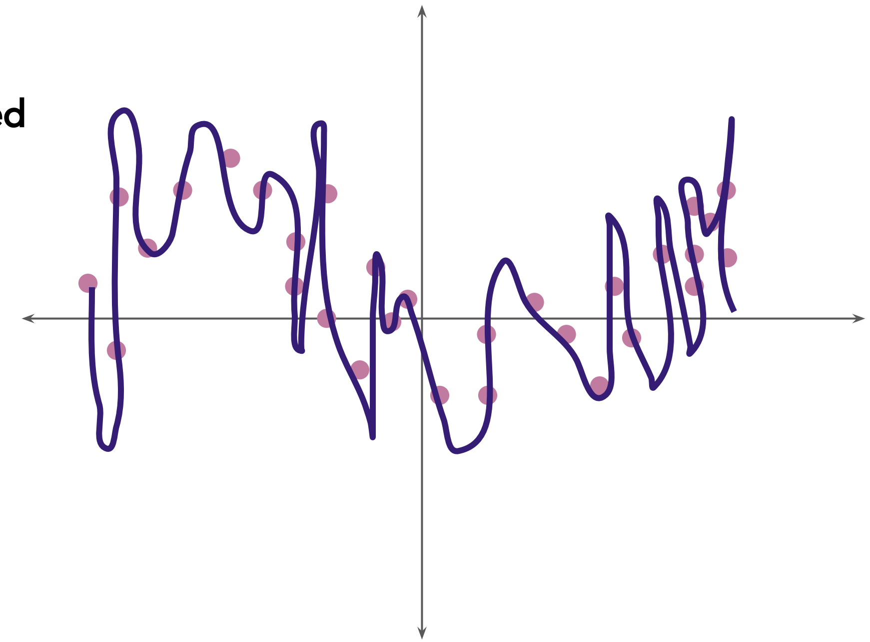
- High **bias** and low **variance**
- Low **training** and **testing** accuracy
- Underrepresentation of the complexity of the dataset
- Algorithm has not actually modeled the data



M I C

Overfitting

- High **variance** and low **bias**
- High **training** accuracy
  - But only because the model has **memorized** the data
- Low **testing** accuracy
- **Memorization:** fitting the training set perfectly or near-perfectly and failing to generalize

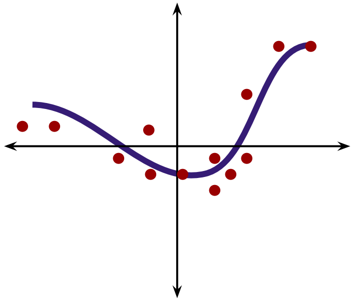


Regularization

“any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”

- Geoffrey Hinton, Yoshua Bengio, and Aaron Courville

- Weight Decay
- Early Stopping
- Dropout Layer
- Dataset Augmentation



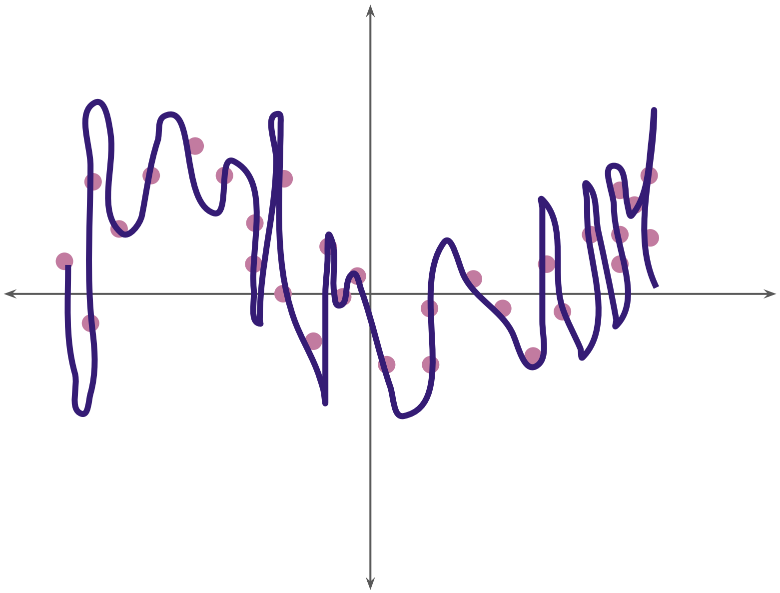
M I C

Weight Decay

$$J'(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \mathit{RegTerm}$$

- Adds a **regularization term** to your original cost function
- J'(θ) is the updated cost function
  - Now, you're trying to minimize both the error and the complexity
- Puts a constraint on the size of the weights by **penalizing** weights that get too large
- If your weights are too large, even a small change to your input will have a huge result

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(\underbrace{h_\theta(x^{(i)})}_{\text{model's prediction}} - \underbrace{y^{(i)}}_{\text{correct answer}})^2$$



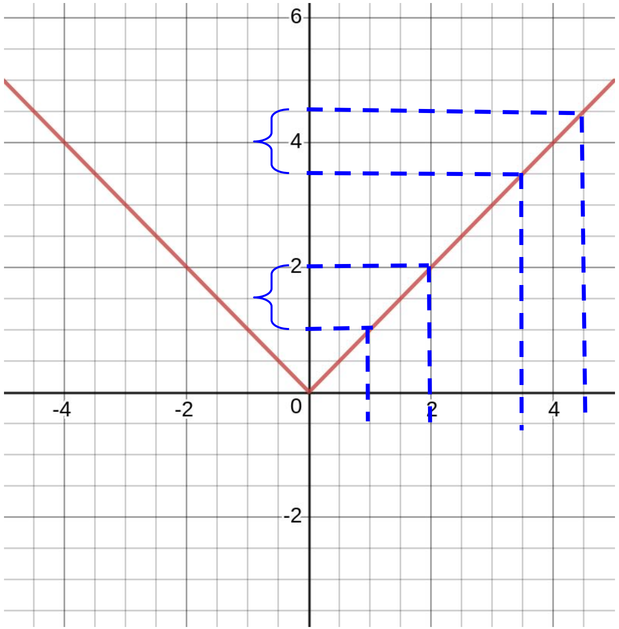
M I C

Weight Decay: L1 ("lasso")

$$L1_{Norm} = \|\theta\|_1 = \sum_i |\theta_i|$$

- LASSO = Least Absolute Shrinkage and Selection Operator
- Sum of the absolute values of the weights
- **Sparsifies** weight matrices
  - Some weights become zero
- "Built-in **feature selection**"
  - What happens when some weights become zero?

$$J'(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda L1_{Term}$$

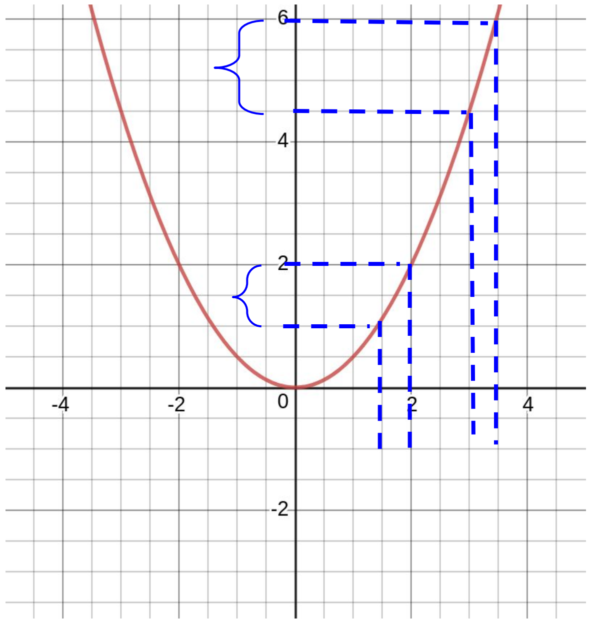


M I C

Weight Decay: L2 ("ridge")

$$L2_{Reg} = \|\theta\|_2^2 = \sum_i \theta_i^2$$

- Known as "ridge" or "Tikhonov" regression
- Sum of the square of the weights
- Drives weights closer to origins, but does **not** sparsify



$$J'(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda L2_{Term}$$

MIC

Weight Decay: L1 vs L2 gradients

- The derivative of L1 is a **constant k**
  - The value of **k** is independent of the value of θ_j
- The derivative of L2 is 2***weight**

Also written $\nabla J(\theta)$

Scalar learning rate

$$\theta_j := \theta_j - \alpha \underbrace{\frac{\partial}{\partial \theta_j} J(\theta)}$$

Individual weights

Cost/objective/loss function

Vector of weights

L1: $\sum_i |\theta_i|$ **Gradient:** $\theta := \theta - \frac{\partial}{\partial \theta}(J(x, y; \theta) + \lambda ||\theta||_1)$

L2: $\sum_i \theta_i^2$ **Gradient:** $\theta := \theta - \frac{\partial}{\partial \theta}(J(x, y; \theta) + \frac{\lambda}{2} ||\theta||_2^2)$

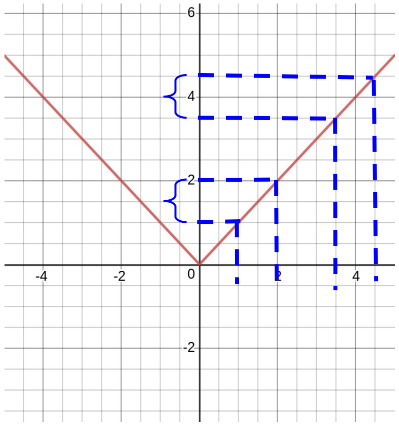
M I C

Weight Decay: L1 vs L2

- L1 reduces all weights evenly, while L2 will penalize larger weights more than smaller ones
- L2 is computationally easier to implement than L1, so it's more commonly used

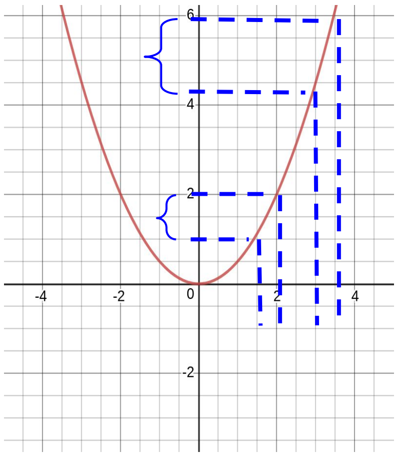
$$L1_{Norm} = \|\theta\|_1 = \sum_i |\theta_i|$$

L1



$$L2_{Reg} = \|\theta\|_2^2 = \sum_i \theta_i^2$$

L2



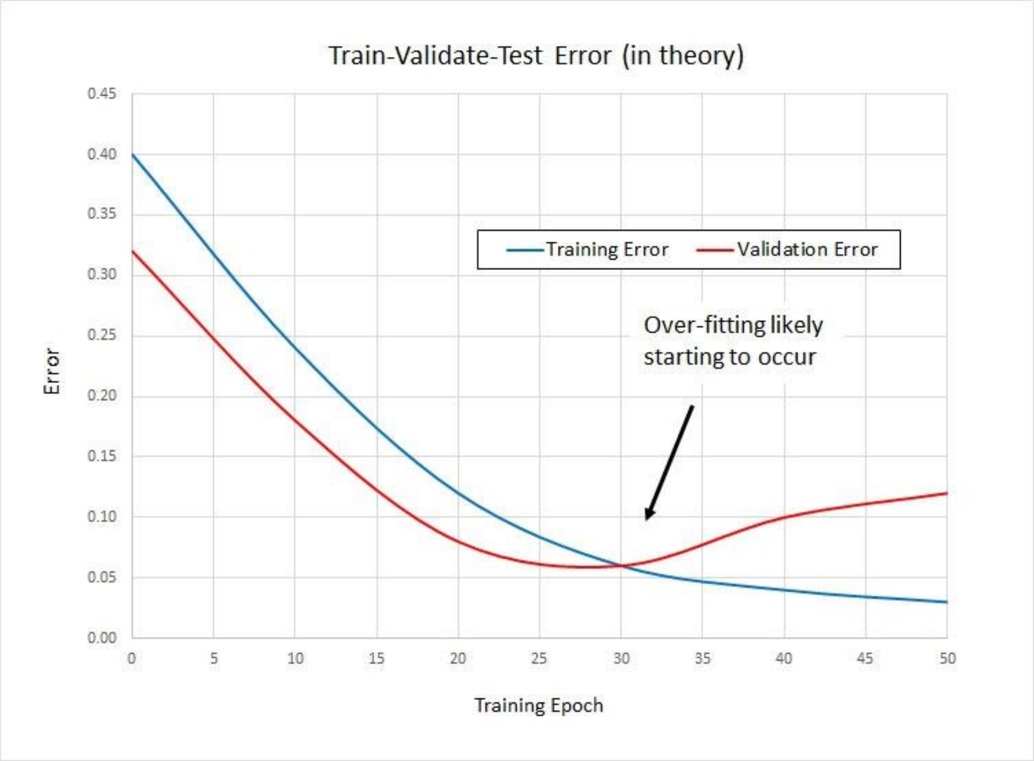
Regularization Parameter $J(\theta; X, y) = J(\theta; X, y) + \lambda L2_{Reg}$

- Regulates how much regularization we apply to our model
- How do you pick an appropriate regularization parameter?
- Too large:
  - High bias
  - Will penalize the parameters too much, and you will end up underfitting
- Too small:
  - High variance
  - Will not penalize the parameters enough, and you will overfit anyways

M I C

Early Stopping

- Plot **validation error** alongside your **training error**
- When validation error starts to increase, simply stop training
- Not this simple in practice, use **stopping criteria**
    - a threshold of error increase
    - a threshold over a period of epochs

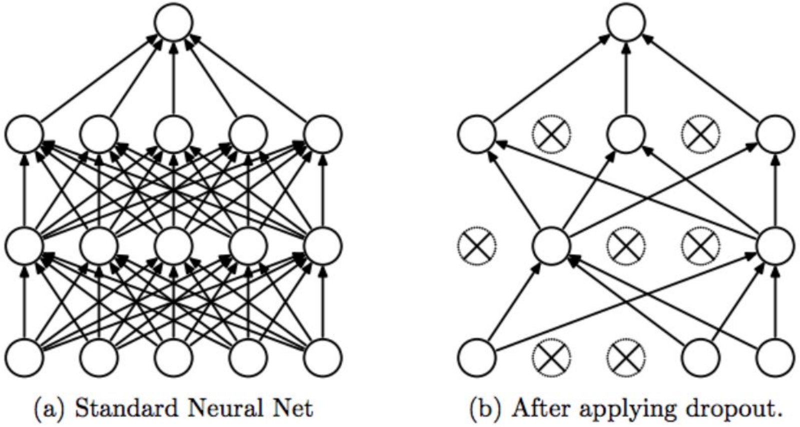


Src: https://visualstudiomagazine.com/articles/2015/05/01/~/media/ECG/visualstudiomagazine/Images/2015/05/0515vsm_McCaffreyFig1.ashx

M I C

Dropout

- Concept: randomly remove units from your neural network
- Intuition:
  - Forces the network to encode the same amount of information into less nodes
- Addresses **coadaptation:** the tendency of nodes to change to “fix the mistakes” of other nodes



Paper: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Dropout Implementation: “Inverted Dropout”

1. Set probability **p** of dropping a node for each layer
2. For a given layer **L**, create a vector **d** of randomly set elements to either 1 or 0, in accordance with your probability **p**
3. Multiply the activations of **L** by **d**, effectively reducing **p** (percentage) of node values to zero
4. Scale up by dividing remaining elements by keep probability (1 - **p**) to maintain expected output value

$L = [(w_0)/0.8, 0, 0, (w_3)/0.8, (w_4)/0.8]$

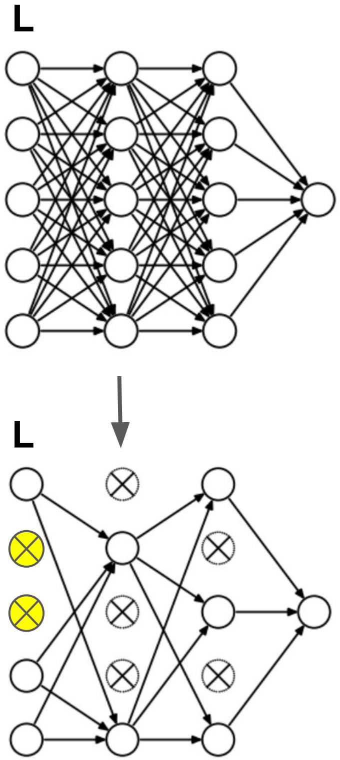
$p = 0.4$

$L = [w_0, w_1, w_2, w_3, w_4]$

$d = [1, 0, 0, 1, 1]$

$L * d^T = [w_0, 0, 0, w_3, w_4]$

$1 - p = 0.8$



Dataset Augmentation

- Most straightforward way to improve a machine learning algorithm?
  - Add more data
- Create new, fake data by making small edits to your existing dataset
- For example: apply a transformation to the data
  - Rotating or scaling images
  - Changing the lighting
  - Don't change the class, e.g. **9** -> **6**
- Not as effective as new data

Src: https://www.researchgate.net/figure/Effects-of-data-augmentation-on-some-example-images-of-the-AgrilPlant-dataset_fig2_319990090

Summary

- **Weight decay:**
  - Penalizing weights that get too large by adding a regularization term to your cost function
- **Early Stopping:**
  - Tracking training and validation error and stopping before overfitting occurs
- **Dropout:**
  - Randomly killing neurons in your neural network to prevent coadaptation
- **Dataset Augmentation:**
  - Creating new data by making minor changes to pre-existing data

That’s all, folks

Thanks for coming!

Resources

- Papers:
  - Co-adaptation: https://arxiv.org/abs/1207.0580
  - Dropout: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
  - Dropout Implementation: https://www.youtube.com/watch?v=D8PJAL-MZv8
  - Early stopping: http://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf
- Other:
  - Algorithmic Justice League: https://www.ajlunited.org/
  - Joy Buolamwini's Ted Talk: https://www.youtube.com/watch?v=UG_X_7g63rY

Feedback

https://goo.gl/forms/U9K3KdeNHatHyZEs1