Regularization

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

Justin Chen
Oct. 10, 2017

Brief Recap: Neural Networks & Gradients

- Neural networks are composition of matrices and non-linear functions
- Train networks on data to approximate functions
- Use gradient descent to search parameter space to minimize cost
- Cost function measures distance between hypothesis and label

MACHINE INTELLIGENCE COMMUNITY

Vectorization in PYTORCH

Create a Random 2x2 Image

```
>> from torch import randn
>> image = randn(2,2)
>> image

 2.3042 -0.3380
-0.2713  0.5415
[torch.FloatTensor of size 2x2]
```

Vectorize Image & Create Class Label

```
>> from torch.autograd import
Variable
>> image = Variable(image.view(1,4))

Variable containing:
 2.3042 -0.3380 -0.2713  0.5415
[torch.FloatTensor of size 1x4]

>> from torch import LongTensor
>> label = Variable(LongTensor([1]))

Variable containing:
 1
[torch.LongTensor of size 1]
```

MACHINE INTELLIGENCE COMMUNITY

MLP in PYTORCH

Define Fully-Connected Network

```
>> import torch.nn as nn
>> net =
nn.Sequential(nn.Linear(4, 2),
nn.Sigmoid(), nn.Linear(2, 2))
>> net
Sequential (
  (0): Linear (4 -> 2)
  (1): Sigmoid ()
  (2): Linear (2 -> 2)
)
```

Feedforward the Feature Vector

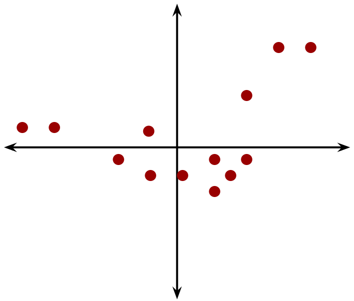
```
>> net(image)
Variable containing:
 0.0817  0.3724
[torch.FloatTensor of size 1x2]
```

Define Optimizer

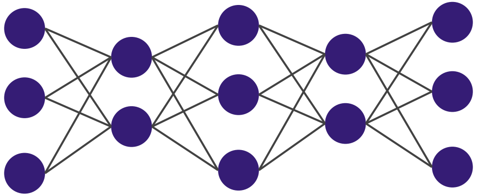
```
>> from torch import optim
>> optimizer =
optim.SGD(net.parameters(), lr=1)
```

MACHINE INTELLIGENCE COMMUNITY

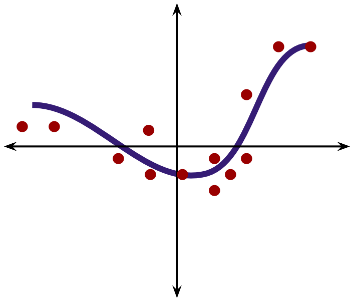
Goal of Learning: Data-Driven Generalization



Dataset



Trained learning algorithm



Approximate function to generalize to new data

MACHINE INTELLIGENCE COMMUNITY

Dataset

Split - percentages depend on available data

Balance - data distributed close to uniformly

MACHINE INTELLIGENCE COMMUNITY

How much data is enough data?

- In practice, the size of real datasets can be on the orders of 10^5, 10^6, 10^7...
- Depends on dimensionality, population, linearity of data

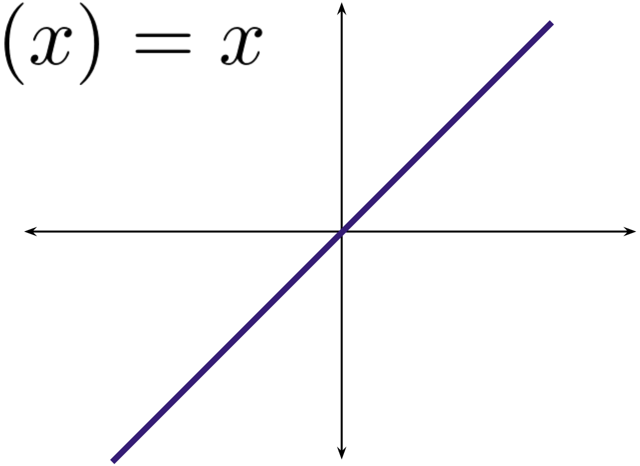
Example:

- Imagine we want to model single pixel images with one color channel
- Each pixel can takes on values in the rangen [0, 255]
- 256 total colors, and in this case 256 total images

How much data is enough data?

- Assume uniform distribution for this example
- Can model with a simple linear classifier over a compact set and only need to collect 256 images to model entire space
- x = 1, f(x) = 1, …

$$f(x) = x$$



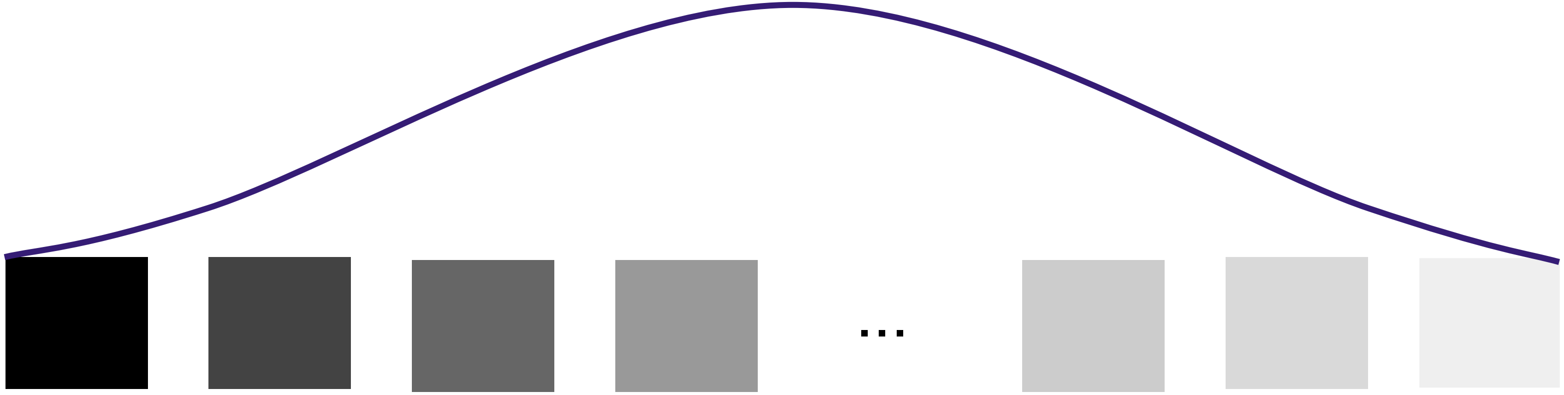
$x =$  $f(x) = 254$

$\frac{1}{256}$ $\frac{1}{256}$ $\frac{1}{256}$ $\frac{1}{256}$ … $\frac{1}{256}$ $\frac{1}{256}$ $\frac{1}{256}$

MACHINE INTELLIGENCE COMMUNITY

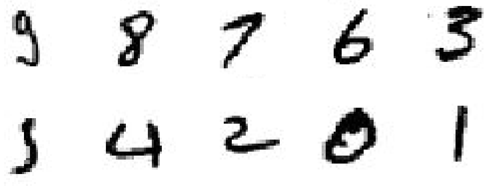
How much data is enough data?

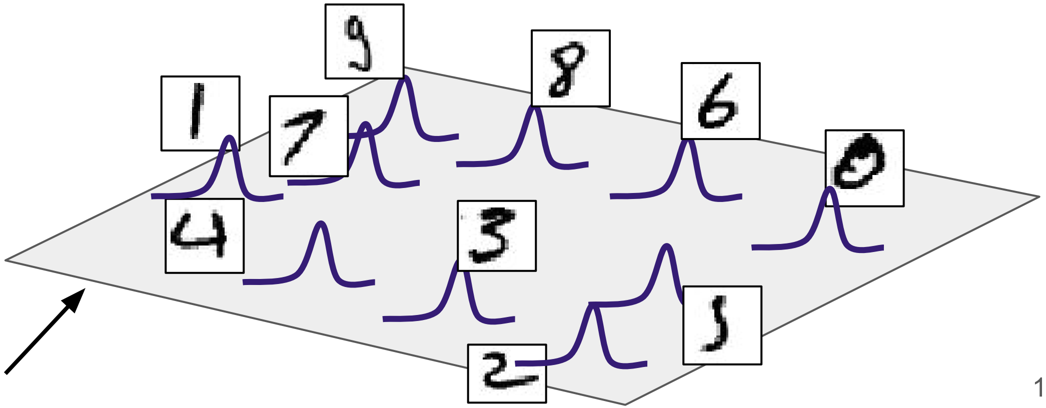
- Now assume Gaussian for this example
- Images towards the center are more likely to occur than those near the tails
- Real images: [117, 137]   21/256 = 8.20% of space
- Random noise: [0, 117) U (137, 255]   235/256 = 91.80% of space



How much data is enough data?

- Now consider 28x28 images with 1 color channel
- 784-dimensional space
- 28 x 28 x 256 x 1 = **200,704 possible images**
- MNIST dataset = 70,000/200,704 = **34.88% of total space**
- **Manifold** = connected regions of space that locally appears Euclidean space



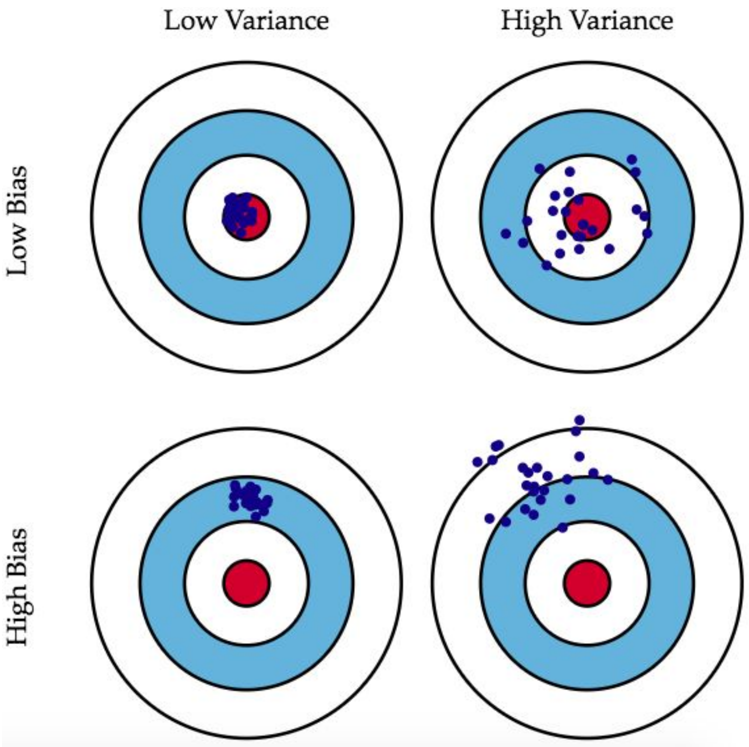


Conceptual illustration of space of possible 28x28x256 images

MACHINE INTELLIGENCE COMMUNITY

Bias-Variance Tradeoff

- What happens if our dataset is **unbalanced**?
- Learning algorithms are only as good as your data
- **Bias** = concentration around a particular area/ set of points
- **Variance** = spread/distance each point is from each other in an area



MACHINE INTELLIGENCE COMMUNITY

Bias-Variance Tradeoff

Bias **<u>error</u>** says "when you predict things how far are you from the expected **true values**"

$$B = E[f(x;D) - E[y|x]]$$

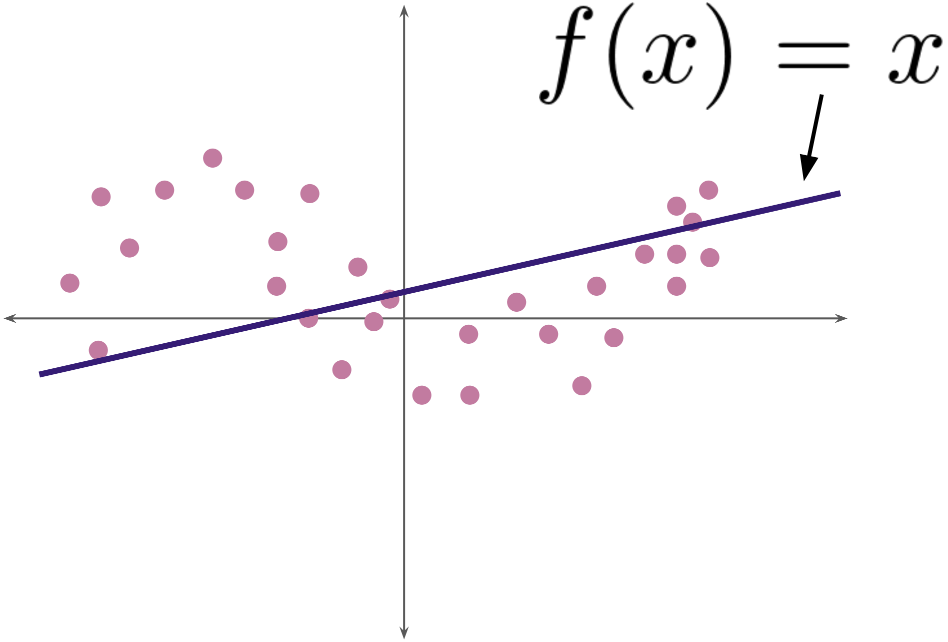
Variance **<u>error</u>** says "when you predict things how far are you from **other predictions** you've made"

$$V = E[(f(x;D) - E[f(x;D)])^2]$$

MACHINE INTELLIGENCE COMMUNITY

Underfitting

- **High bias error**
- Low **training** <u>**accuracy**</u>
- Low **testing** <u>**accuracy**</u>
- Learning algorithm lacks capacity to model data

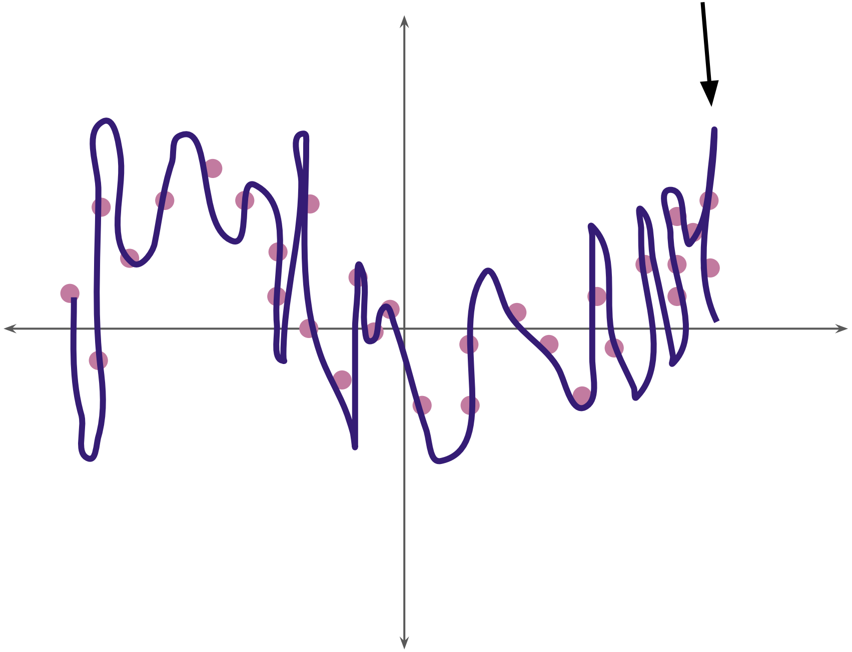


Data from $f(x) = sin(x - \pi)$

Overfitting

$$f(x) = \theta_n x_n^k + \ldots + \theta_0 x_0$$

- **High variance error**
- High **training accuracy**
- Low **testing accuracy**
- Model is **overparameterized**
- Fitting the training set perfectly (**memorization**)



Data from $f(x) = sin(x - \pi)$

MACHINE INTELLIGENCE COMMUNITY

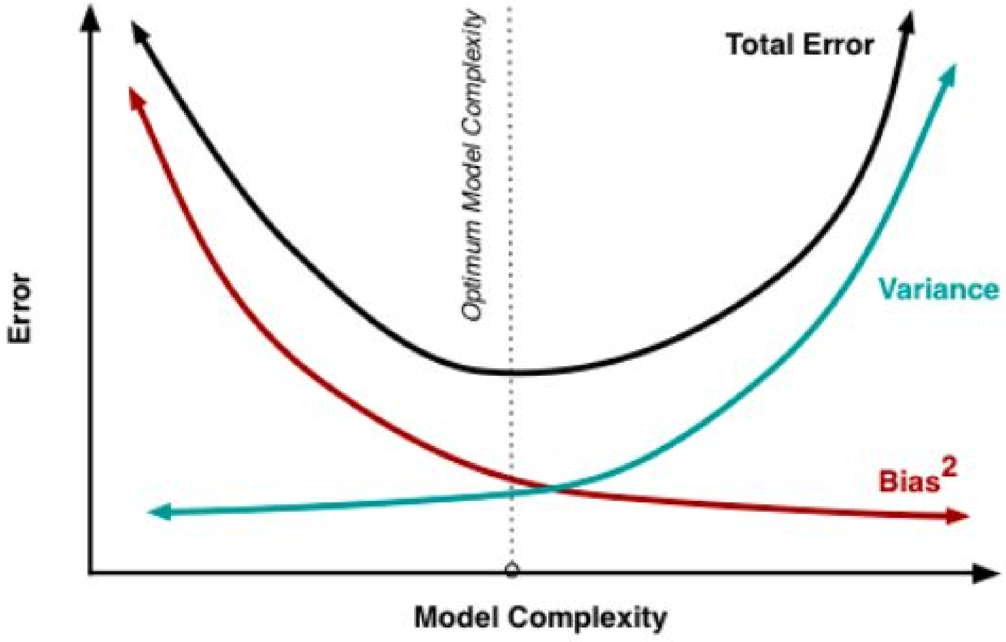
Capacity

- **Representational capacity** - set of hypotheses that can be **expressed** by a trained model's parameters
- **Effective capacity** - set of hypotheses that can be **reached by training** on a particular set of data
- **Model capacity** - number of trainable model parameters

$$\underbrace{EC}_{\text{Effective capacity}}(\underbrace{\mathcal{A}}_{\text{Learning algorithm}}) = \{\underbrace{h}_{\text{Hypothesis}} | \exists \underbrace{\mathcal{D}}_{\text{Dataset}} \ s.t. \ h \in \mathcal{A}(\mathcal{D})\}$$

MACHINE INTELLIGENCE COMMUNITY

Bias-Variance Tradeoff



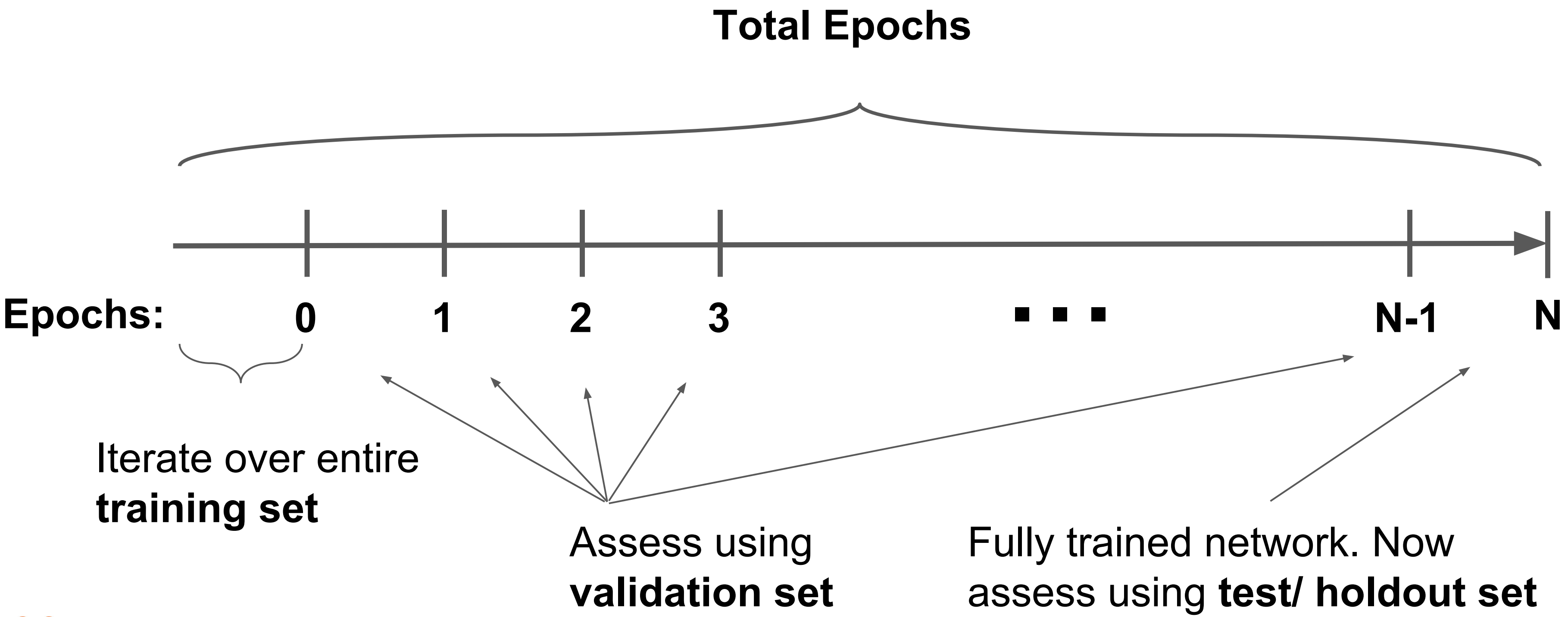
MACHINE INTELLIGENCE COMMUNITY

Priors

- A prior knowledge can help generalization
- Independent features
  - Don’t want to over represent features
  - e.g. Modeling house prices
    - Features: length and width of property, and area
- Model selection
  - Selecting an architecture that is better suited for particular data
    - e.g. CNN for compositional data or RNN for sequential data, or logistic classifier for binary classification
- No Free Lunch Theorem
  - There is no single learning algorithm that works best for all problems.

MACHINE INTELLIGENCE COMMUNITY

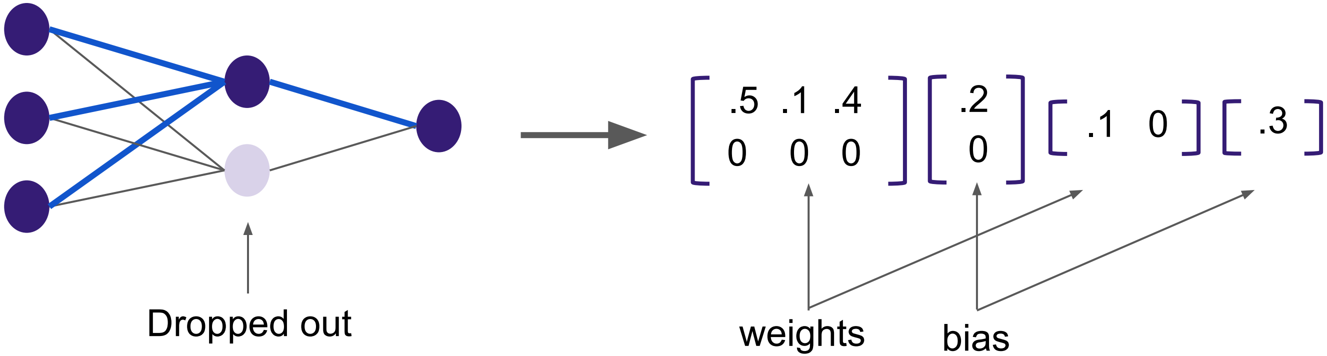
Early Stopping



MACHINE INTELLIGENCE COMMUNITY

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

- *Srivastava et al*, 2014
- Randomly disable neurons during training with Bernoulli random variable
- Scale output of dropped out of neuron during inference
- Causes networks to become sparse (lots of zeros in matrices)



Co-adaptation

- *Hinton et. al, 2012*
- Co-adaptation - neurons tend to rely on features learned by connected neurons
- To break that adaptation, neurons are dropped during training
- Forces each neuron to try to learn something interesting
- Effectively, learns exponentially many subnetworks that are averaged during inference

MACHINE INTELLIGENCE COMMUNITY

Dropout in PYTORCH

Define Dropout Network

```
>> net = nn.Sequential(nn.Linear(4, 2), nn.Sigmoid(), nn. Dropout(),
nn.Linear(2, 2), nn.Softmax())
>> net
Sequential (
   (0): Linear (4 -> 2)
   (1): Sigmoid ()
   (2): Dropout (p = 0.5)
   (3): Linear (2 -> 2)
   (4): Softmax ()

)
```

MACHINE INTELLIGENCE COMMUNITY

Network Parameter in PYTORCH

```
>> [x for x in net.parameters()]
[Parameter containing:
-0.0046 -0.1379 -0.4034  0.0733
-0.4190  0.1449  0.2962  0.2015
[torch.FloatTensor of size 2x4]
, Parameter containing:
-0.1390
 0.4595
[torch.FloatTensor of size 2]
, Parameter containing:
-0.5009  0.4981
-0.4290  0.6568
[torch.FloatTensor of size 2x2]
, Parameter containing:
-0.4264
-0.3450
[torch.FloatTensor of size 2]
]
```

Layer 1 Weights

Layer 1 Biases

Layer 2 Weights

Layer 2 Biases

MACHINE INTELLIGENCE COMMUNITY

Update Parameters in PYTORCH

Local Gradients

```
>> [x.grad for x in
net.parameters()]
[None, None, None, None]
```

Feedforward

```
>> optimizer.zero_grad()
>> hypo = net(image)
Variable containing:
 0.4322  0.5678
[torch.FloatTensor of size 1x2]
```

Backpropagation

```
>> import torch.nn.functional as F
>> loss = F.nll_loss(hypo, label)
>> loss
Variable containing:
-0.5497
[torch.FloatTensor of size 1]

>> loss.backward()
```

Gradient Descent

```
>> optimizer.step()
```

zero_grad()	Clears the gradients of all optimized variables.
nll_loss()	The negative log likelihood loss
step()	Updates the parameters

MACHINE INTELLIGENCE COMMUNITY

L1 Weight Decay

- Sum the distances between each parameter and the origin
- Sparsifies weight matrices - some weights become zero
- Each of theta is a weight vector - a row in a weight matrix (**neuron**)
- L2 tends to work better in practice

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

$$\theta := \theta - \frac{\partial}{\partial \theta}(J(x, y; \theta) + \|\theta\|_1)$$

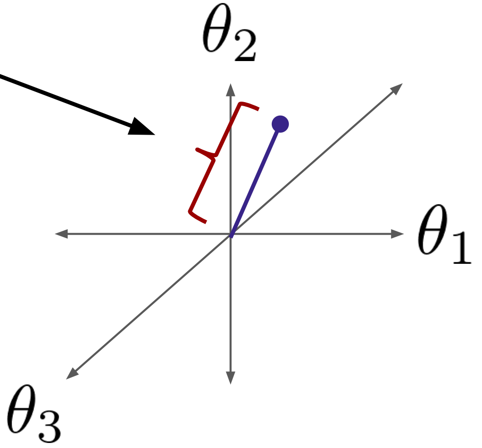
MACHINE INTELLIGENCE COMMUNITY

L2 Regularization

- L2-norm a.k.a. Euclidean norm

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

$$\bar{\theta}^\top = [\theta_1 \ \theta_2 \ \theta_3]$$



- L2 regularization a.k.a. weight decay
- Weight decay is L2-norm squared

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

MACHINE INTELLIGENCE COMMUNITY

L2 Weight Decay

- Add regularization term to cost function
- λ controls how much weights should be decayed
- ½ and square for mathematical convenience
- Pulls all weights towards origin/ zero
- Forces each dimension to contribute

$$\theta := \theta - \frac{\partial}{\partial \theta}\left(J(x, y; \theta) + \frac{\lambda}{2}\|\theta\|_2^2\right)$$

$$\theta := \theta - \frac{\partial}{\partial \theta} J(x, y; \theta) - \lambda\theta$$

MACHINE INTELLIGENCE COMMUNITY

Data Augmentation

- Apply some natural domain-dependent transformation
    - e.g. change the lighting in an image
- Useful when you need more data
- Regularizes training by showing network more examples from manifold

Image Data

original

augmented

Text Data

I like doggos

I love doggos

I really love doggos

I really like doggos

MACHINE INTELLIGENCE COMMUNITY

Shoutout to our Sponsors

Boston University Computer Science

Boston University SPARK!

Boston University Software Application and Innovation Lab

MACHINE INTELLIGENCE COMMUNITY

Upcoming Events

MIC Paper signup: https://goo.gl/iAm6TL
BUMIC Projects signup: https://goo.gl/GmP9oK

MIT MIC reading group:

Paper: Decoupled Neural Interfaces Using Synthetic Gradients
Location: MIT 56-154 (building 56, room 154)
Date: 10.12.17 Time: 5 PM

BU MIC reading group:

Paper: Self-Normalizing Neural Networks
Location: Fishbowl Conference Room
Date: 10.11.17 Time: 5 PM

Next workshop:

Topic: Compositional Data
Location: BU Hariri Seminar Room
Date: 10.17.17 Time: 7 PM

MACHINE INTELLIGENCE COMMUNITY

References & Further Reading

[1] Fortmann-Roe, Scott. "Understanding the bias-variance tradeoff." (2012).

[2] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* 15.1 (2014): 1929-1958.

[3] Ba, Jimmy, and Brendan Frey. "Adaptive dropout for training deep neural networks." *Advances in Neural Information Processing Systems*.2013.

[4] Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization." arXiv preprint arXiv:1611.03530 (2016).

[5] Krueger, David, et al. "Deep Nets Don't Learn via Memorization." (2017).

[6] Huang, Gao, et al. "Deep networks with stochastic depth." European Conference on Computer Vision. Springer International Publishing, 2016.

[7] Singh, Saurabh, Derek Hoiem, and David Forsyth. "Swapout: Learning an ensemble of deep architectures." Advances in Neural Information Processing Systems. 2016.

[8] Krueger, David, et al. "Zoneout: Regularizing rnns by randomly preserving hidden activations." arXiv preprint arXiv:1606.01305 (2016).

[9] Sun, Xu, et al. "meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting." arXiv preprint arXiv:1706.06197 (2017).

[10] https://medium.com/towards-data-science/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2

MACHINE INTELLIGENCE COMMUNITY