Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

Revisiting Cross Validation
A “Too Early” History of Deep Learning

Logistics and Agenda

- Logistics
  - Grading update
- Agenda
  - Revisiting Cross Validation
  - Town Hall
  - “Deep Learning” History

Class Overview, by topic

Table Data Visualization
Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing
Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks

Convolutional Networks

Recurrent Networks

Keras, Tensorflow
Intuition, Detailed implement.

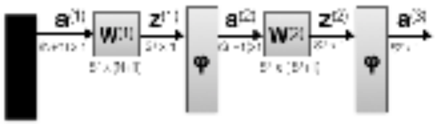
Ethics in Language Models

ConceptNet
Case studies

Last time:

Back propagation summary

$$J(\mathbf{W}) = \sum_k^M (\mathbf{y}^{(k)} - \mathbf{a}^{(L)})^2$$



$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{z}_i^{(l)}} a_j^{(l)}$$

1. Forward propagate to get **z**, **a** for all layers
2. Get final layer gradient
3. Update back propagation variables
4. Update each $\mathbf{W}^{(l)}$

for each $\mathbf{y}^{(i)}$

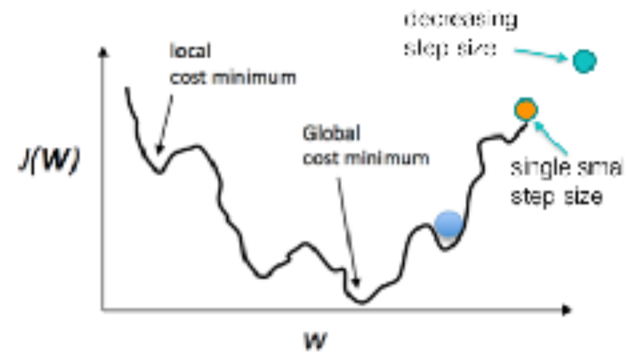
$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = -2(\mathbf{y}^{(i)} - \mathbf{a}^{(3)}) * \mathbf{a}^{(3)} * (1 - \mathbf{a}^{(3)})$$

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} = \text{diag}[\mathbf{a}^{(l+1)} * (1 - \mathbf{a}^{(l+1)})] \cdot \mathbf{W}^{(l+1)} \frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l+1)}}$$

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial J(\mathbf{W}^{(l)})}{\partial \mathbf{z}^{(l)}} \cdot \mathbf{a}^{(l)}$$

Problems with Advanced Architectures

- Space is no longer convex
  - **One solution:**
    - start with large step size
    - "cool down" by decreasing step size for higher iterations



Practical Implementation of Architectures

- A new cost function: **Cross entropy**

$$J(\mathbf{W}) = -[\mathbf{y}^{(i)} \ln \mathbf{a}^{(L)} + (1 - \mathbf{y}^{(i)}) \ln(1 - \mathbf{a}^{(L)})]$$

speeds up initial training

$$\frac{\partial J(\mathbf{W})}{\mathbf{z}^{(L)}} = ({}_i\mathbf{a}^{(L+1)} - \mathbf{y}^{(i)})$$

$$\frac{\partial J(\mathbf{W})}{\mathbf{z}^{(2)}} = ({}_i\mathbf{a}^{(3)} - \mathbf{y}^{(i)})$$

```
# vectorized backpropagation
sigma3 = (A3-Y_enc) # <- this is only line
sigma2 = (W2.T @ sigma3)*A2*(1-A2)

grad1 = sigma2[1:,:] @ A1
grad2 = sigma3 @ A2.T
```

new update

```
# vectorized backpropagation
sigma3 = -2*(Y_enc-A3)*A3*(1-A3)
sigma2 = (W2.T @ sigma3)*A2*(1-A2)

grad1 = sigma2[1:,:] @ A1
grad2 = sigma3 @ A2.T
```

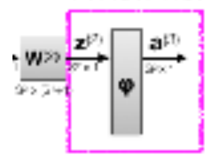
$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(2)}} = -2(\mathbf{y}^{(i)} - \mathbf{a}^{(3)}) * \mathbf{a}^{(3)} * (1 - \mathbf{a}^{(3)})$$ old update

bp-5

Neural Networks and Deep Learning, Michael Nielsen, 2015

Practical Implementation of Architectures

- A new nonlinearity: **recitifed linear units**



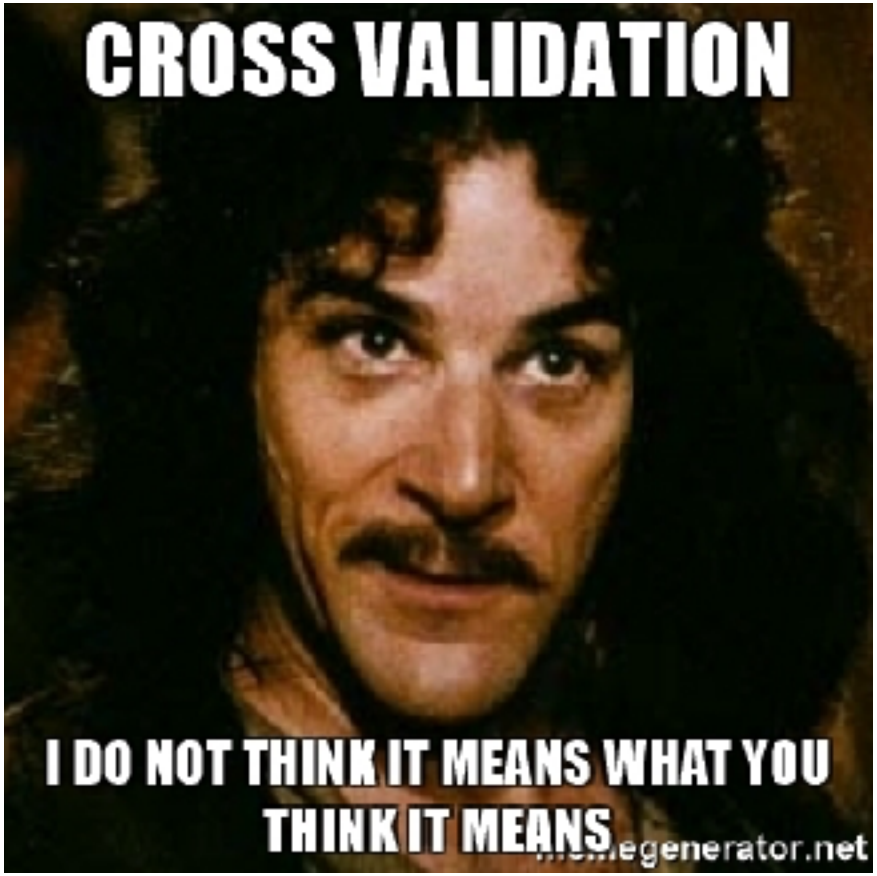
$$\phi(\mathbf{z}^{(i)}) = \begin{cases} \mathbf{z}^{(i)}, \text{if } \mathbf{z}^{(i)} > 0 \\ 0, \text{else} \end{cases}$$

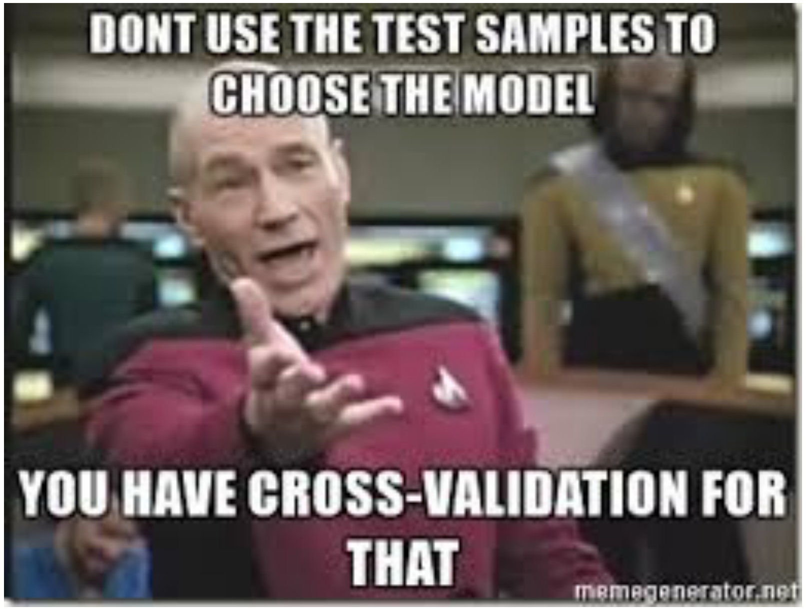
it has the advantage of **large gradients** and **extremely simple** derivative

$$\frac{\partial \phi(\mathbf{z}^{(i)})}{\partial \mathbf{z}^{(i)}} = \begin{cases} 1, \text{if } \mathbf{z}^{(i)} > 0 \\ 0, \text{else} \end{cases}$$

Neural Networks and Deep Learning, Michael Nielsen, 2015

Revisiting Cross Validation





Review: Grid Searching

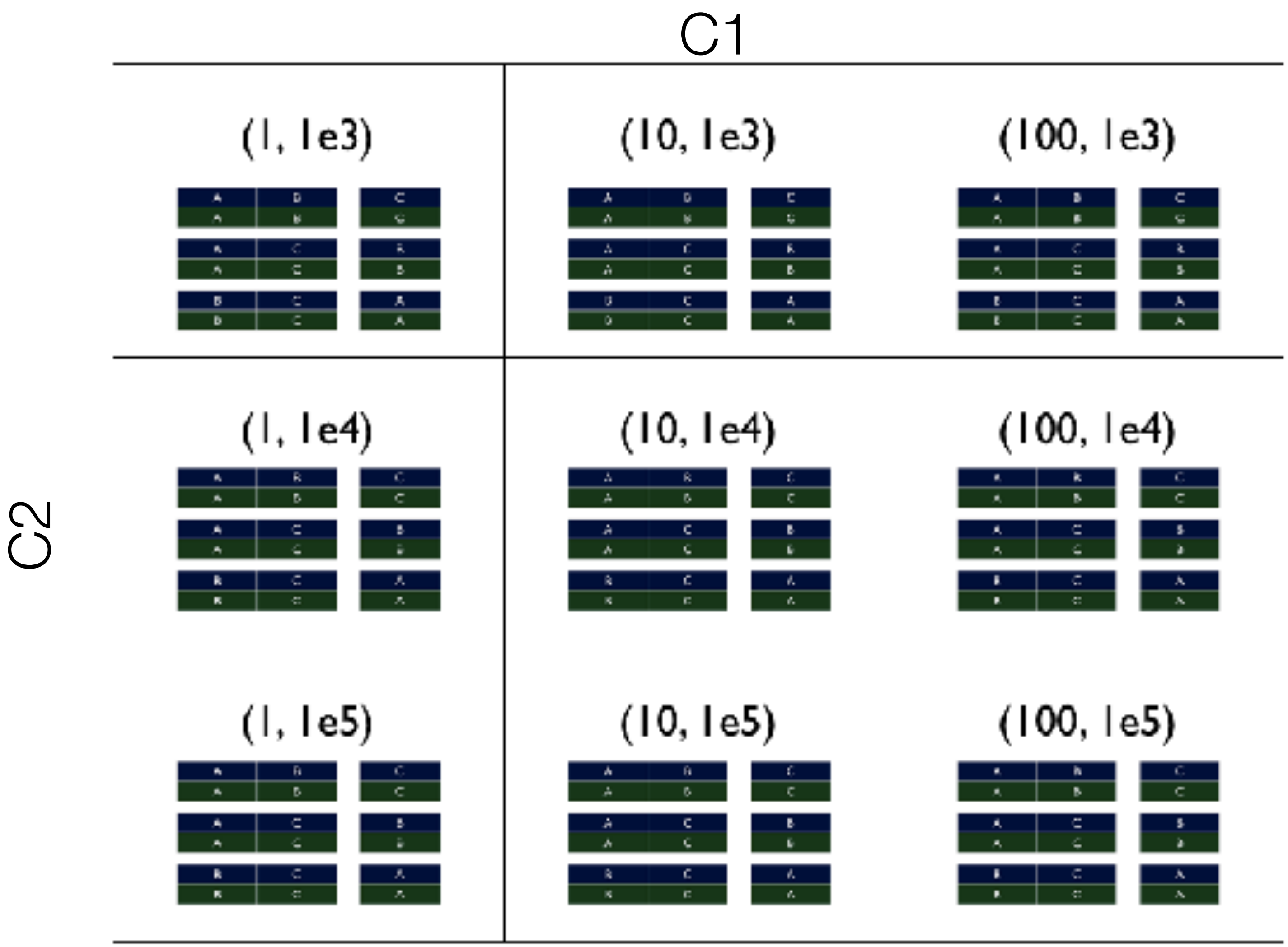
Trying to find the best parameters

`NN: C1=[1, 10, 100] C2=[1e3,1e4,1e5]`

	C1		
C2	(1, 1e3)	(10, 1e3)	(100, 1e3)
	(1, 1e4)	(10, 1e4)	(100, 1e4)
	(1, 1e5)	(10, 1e5)	(100, 1e5)

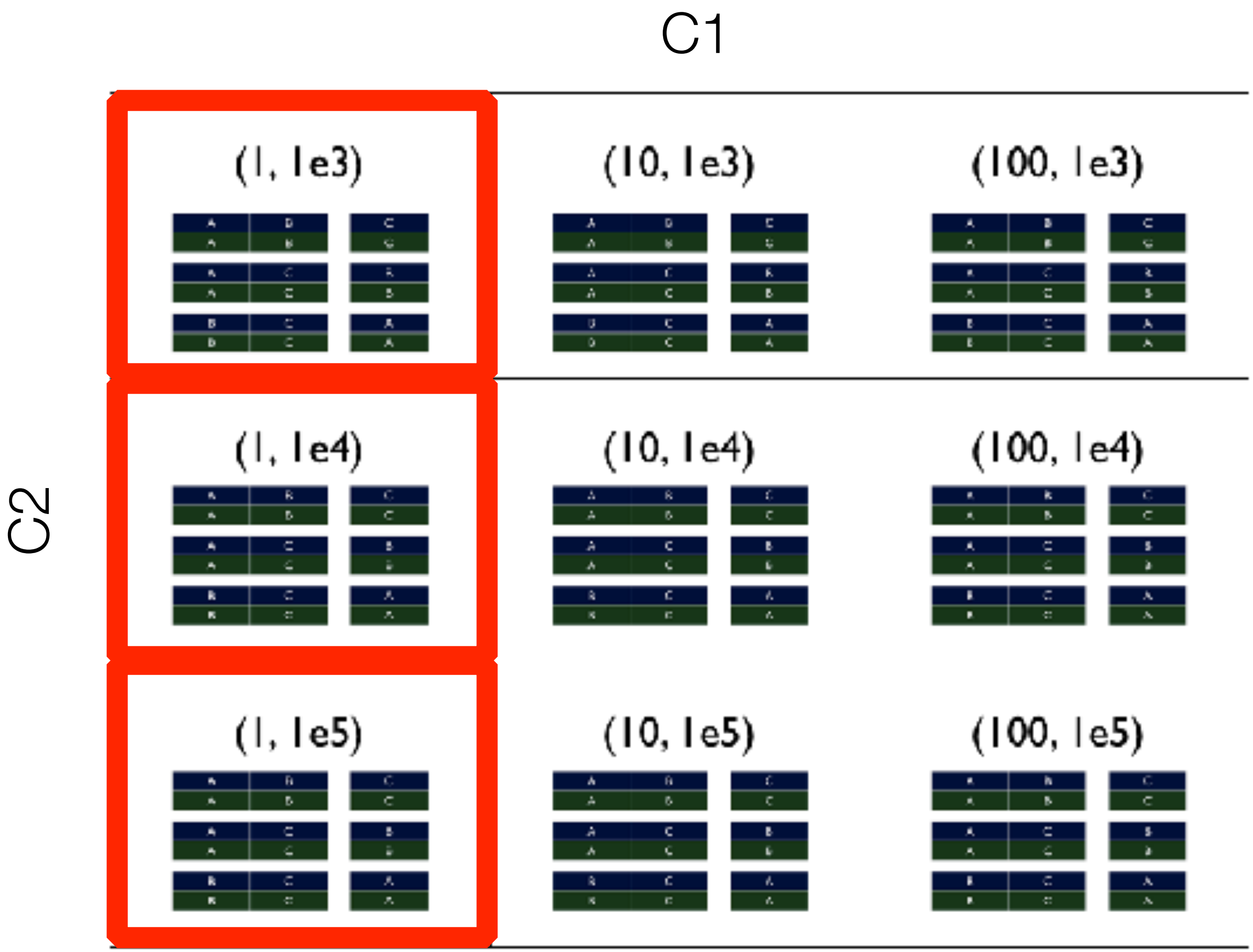
Review: Grid Searching

For each value, want to run cross validation…



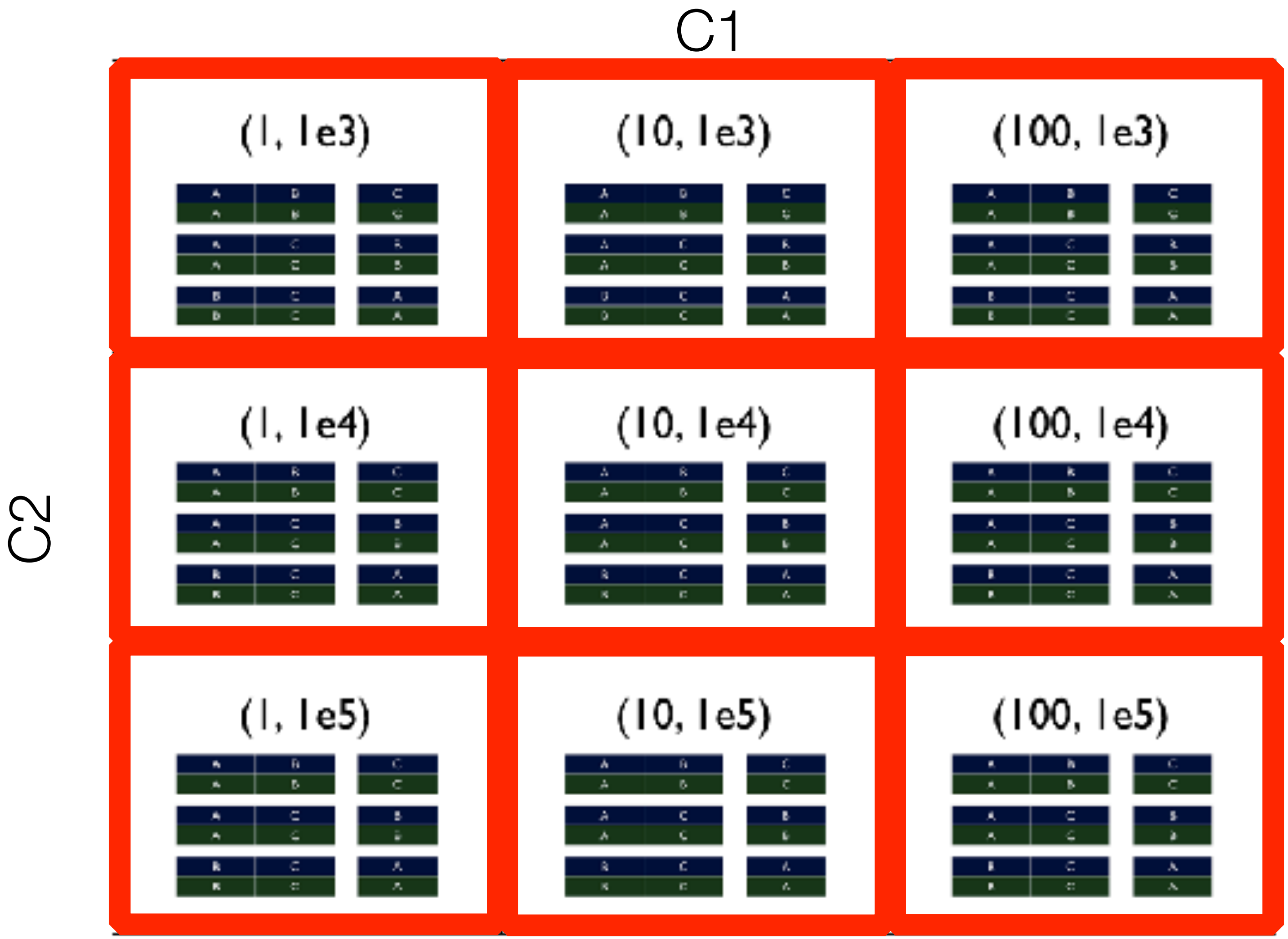
Review: Grid Searching

Could perform iteratively



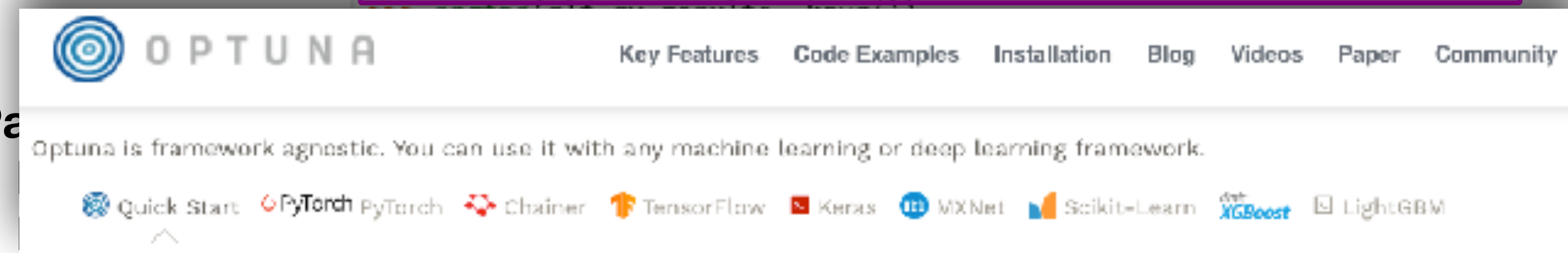
Review: Grid Searching

or at random…



Review: Grid Searches in Scikit-learn

```
>>> from sklearn import svm, datasets
>>> from sklearn.model_selection import GridSearchCV
>>> iris = datasets.load_iris()
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC()
>>> clf = GridSearchCV(svc, parameters)
>>> clf.fit(iris.data, iris.target)
GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 10], 'kernel': ('linear', 'rbf')})
```



Pa

-
-

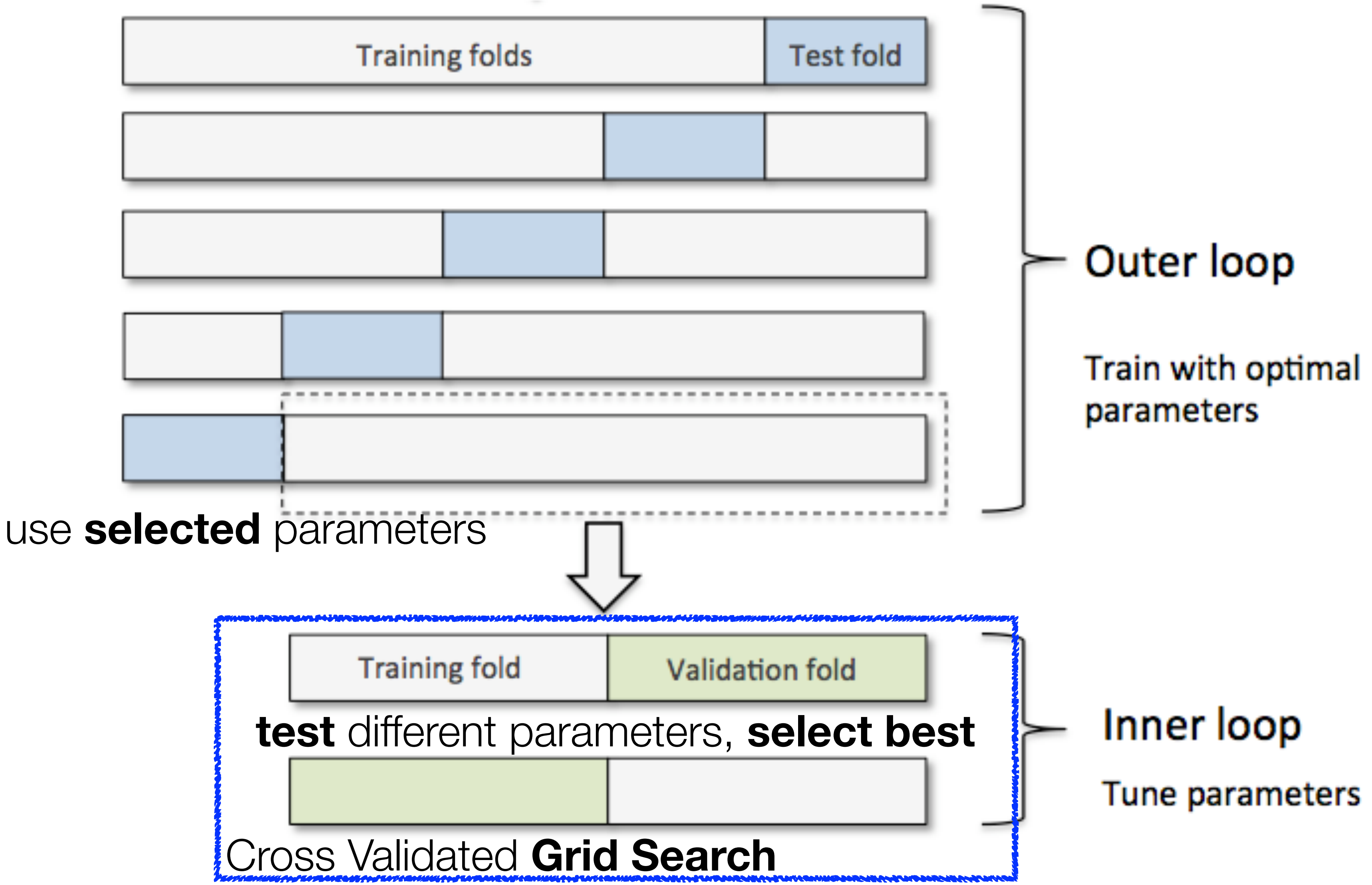
values, sampled

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import RandomizedSearchCV
>>> from scipy.stats import uniform
>>> iris = load_iris()
>>> logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
...                               random_state=0)
>>> distributions = dict(C=uniform(loc=0, scale=4),
...                      penalty=['l2', 'l1'])
>>> clf = RandomizedSearchCV(logistic, distributions, random_state=0)
>>> search = clf.fit(iris.data, iris.target)
>>> search.best_params_
{'C': 2..., 'penalty': 'l1'}
```

Review: Data Snooping

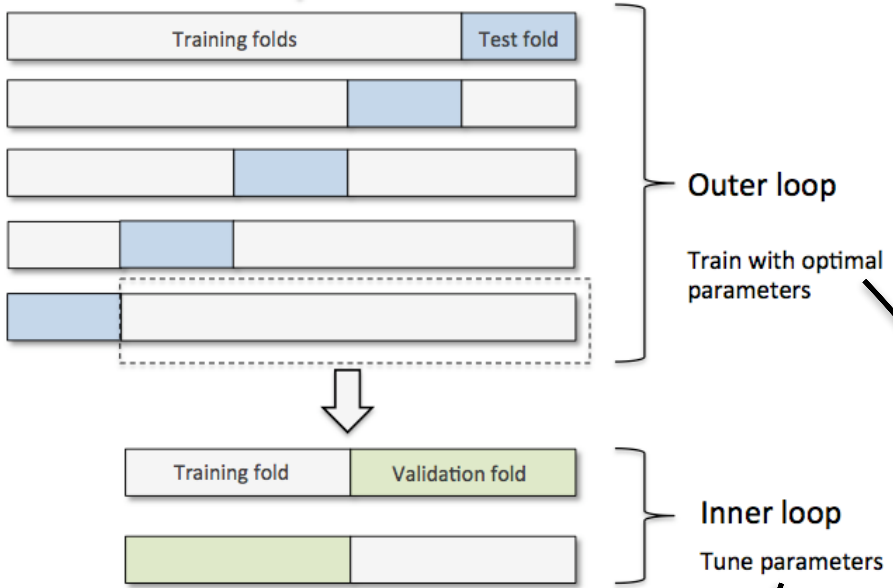
- Using the grid search parameters and testing on the same set…
    - the **performance on the dataset** could now be **biased**
    - cannot determine the **expected performance** on **new data**
    - this is **data snooping**

Review: Solution: Nested Cross Validation



https://github.com/rasbt/python-machine-learning-book/

Review: Nested Cross Validation: Hyper-parameters




```
gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=2)

# Note: Optionally, you could use cv=2
# in the GridSearchCV above to produce
# the 5 x 2 nested CV that is shown in the figure.

scores = cross_val_score(gs, X_train, y_train, scoring='accuracy', cv=5)
print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

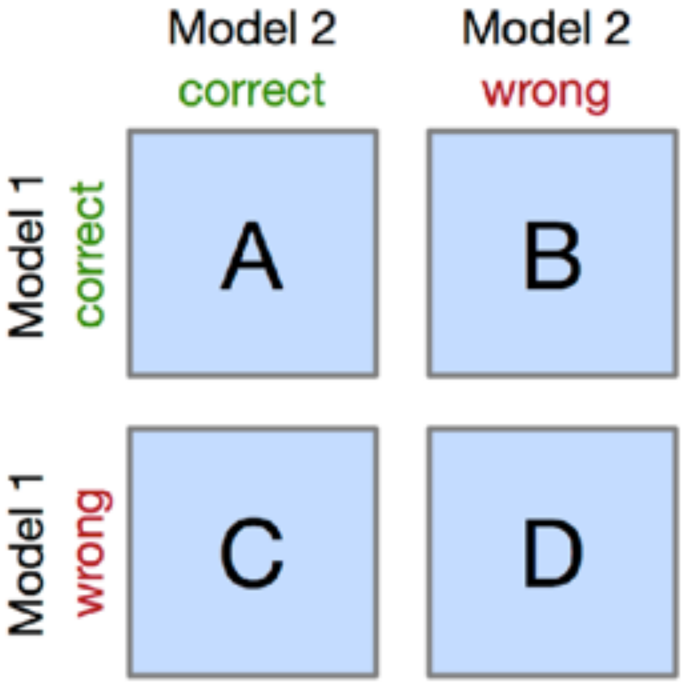
https://github.com/rasbt/python-machine-learning-book/

Self Test

- **What is the end goal of nested cross-validation?**
  - A. To determine hyper parameters
  - B. To estimate generalization performance
  - C. To estimate generalization performance when performing hyper parameter tuning
  - D. To estimate the variation in tuned hyper parameters

McNemar Testing for Comparing Performance

Few assumptions, **Null hypothesis**: predictions are not different!



McNemar and Edwards, 1948

$$\chi^2 \approx \frac{(|B - C| - 1)^2}{B + C}$$

If predictions are drawn from the same distributions, then this equation follows χ **squared statistic with one DOF**

Steps:

1. Compare each model's predictions on the same test data (2x2 matrix)
2. Calculate χ^2 statistic
3. Look up *critical value* associated with χ^2 statistic for given confidence
4. Are you confident enough to **reject the null hypothesis** that the performance is the same (p<0.05)?

One caveat: Statistical power depends upon B+C, which might be small, even with lots of test data.

https://sebastianraschka.com/blog/2018/model-evaluation-selection-part4.html

McNemar Example

Model 1	Model 2	Label	Matrix
T-shirt	T-shirt	T-shirt	A
Sneaker	T-shirt	Sneaker	B
T-shirt	Pullover	Pullover	C
Sneaker	Sneaker	Sneaker	A
T-shirt	Sneaker	Sneaker	C
Pullover	Pullover	T-shirt	D
Pullover	T-shirt	Pullover	B
Sneaker	Sneaker	Sneaker	A
Sneaker	Sneaker	Sneaker	A

	Model 2 correct	Model 2 wrong
Model 1 correct	4 A	2 B
Model 1 wrong	2 C	1 D

McNemar and Edwards, 1948

$$\chi^2 \approx \frac{(|B - C| - 1)^2}{B + C}$$

$$\chi^2 = \frac{(|2 - 2| - 1)^2}{2 + 2} = 0.25$$

Confidence	0.90	0.95	0.99
1 DOF, Critical Value	2.706	3.841	6.635

https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm

Since 0.25 < 3.841, we cannot reject the null hypothesis. This means **we should not say the models' performance are different** based on the evidence.

Town Hall

Some History of Deep Learning

Neural Networks: Where we left it

- Before 1986: AI Winter
- 1986: *Rumelhart*, *Hinton*, and *Williams* popularize gradient calculation for multi-layer network
    - *technically* introduced by Werbos in 1982
- **difference**: Rumelhart *et al.* validated ideas with a computer
- until this point no one could train a multiple layer network consistently
- algorithm is popularly called **Back-Propagation**
- wins pattern recognition prize in 1993, becomes de-facto machine learning algorithm in the 90's

David Rumelhart

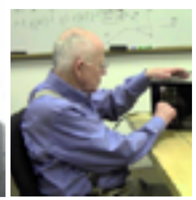
Geoffrey Hinton

Machine Learning Timeline (Neural Nets)

- Up to this point: back propagation saved AI winter
- 80’s, 90’s, 2000’s: neural networks for image processing start to get deeper
  - but back propagation no longer efficient for training
  - Back propagation gradient **stagnates** research—can’t train **deeper** networks

- Second AI winter begins, research in NN plummets
- Funding for and accepted papers with Neural Networks asymptotically approaches zero

1949, Hebb’s Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network


1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

1940 — 1960 — 1980 — 2000 — 2020

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind


1957, Rosenblatt
Perceptron

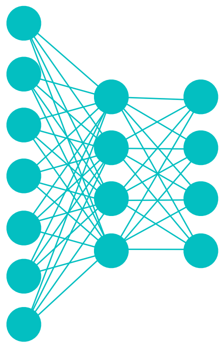
1969, Minsky & Papert
Linear Models are Doomed


2001, Breiman
Random Forests

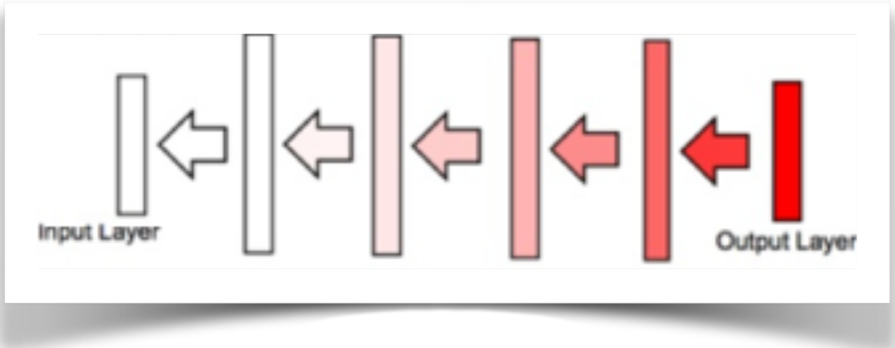
Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

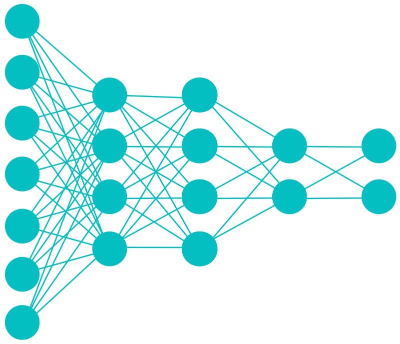
History of Deep Learning: Winter

- AI Winter is coming:



Easy to train, performs on par with other methods





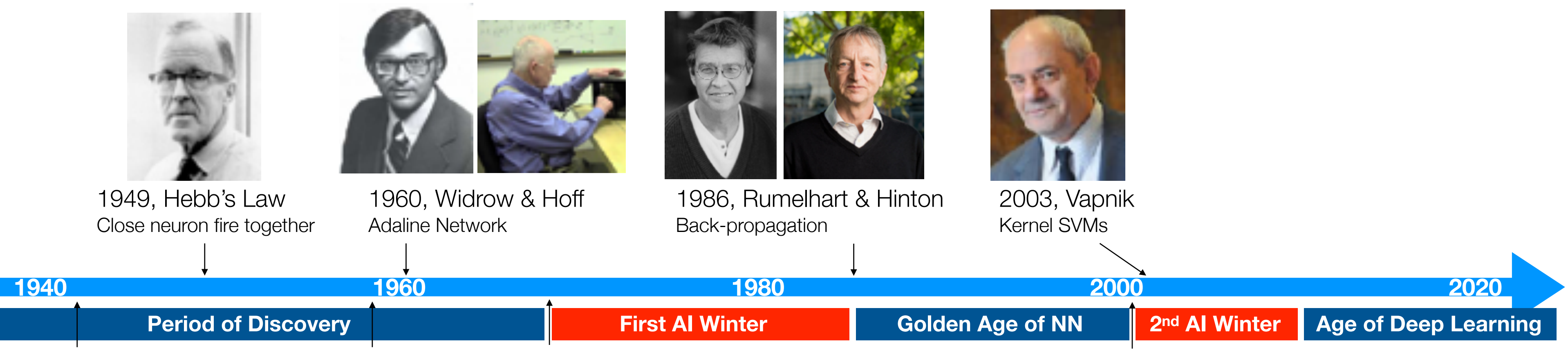
Hard to train, performs worse than other methods
~chance (untrainable)

Researcher have difficulty reconciling expressiveness with performance

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Machine Learning Timeline (Neural Nets)

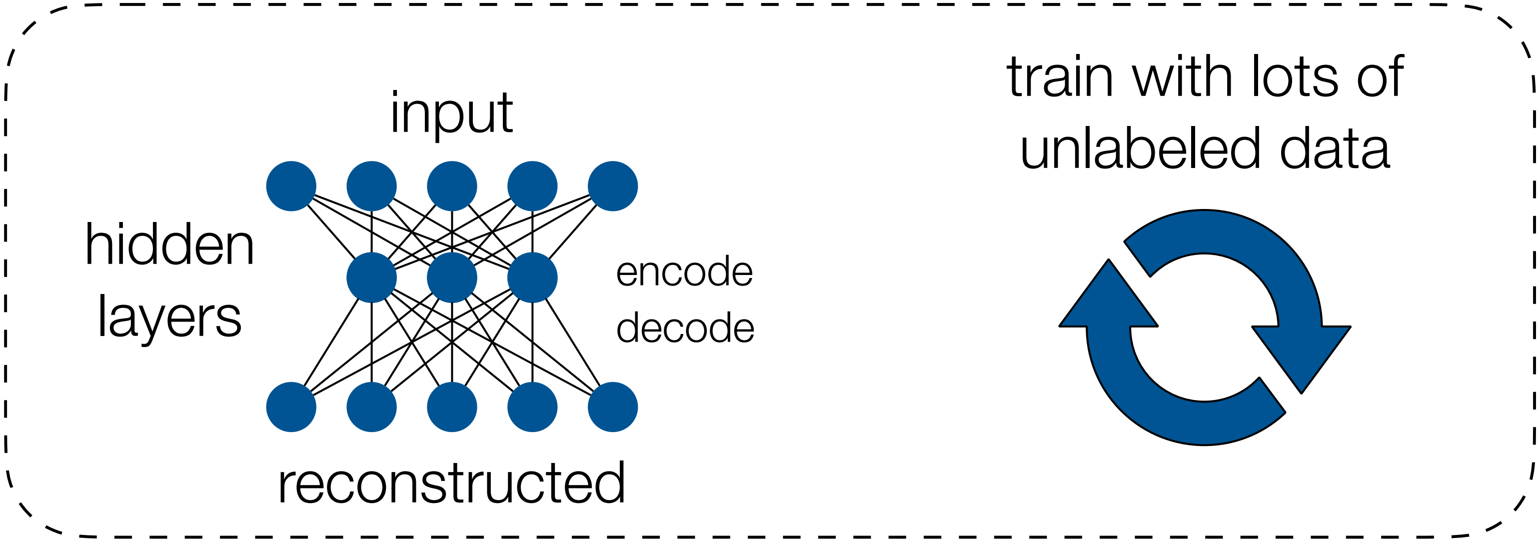
- 2004: Hinton secures funding from CIFAR based on his reputation
  - *eventually*: Canada would be savior for neural networks
  - Hinton rebrands: **Deep Learning**
- 2006: Hinton publishes paper on using pre-training and Restricted Boltzmann Machines
- 2007: Another paper: Deep networks are more efficient when pre-trained
  - RBMs not really the important part

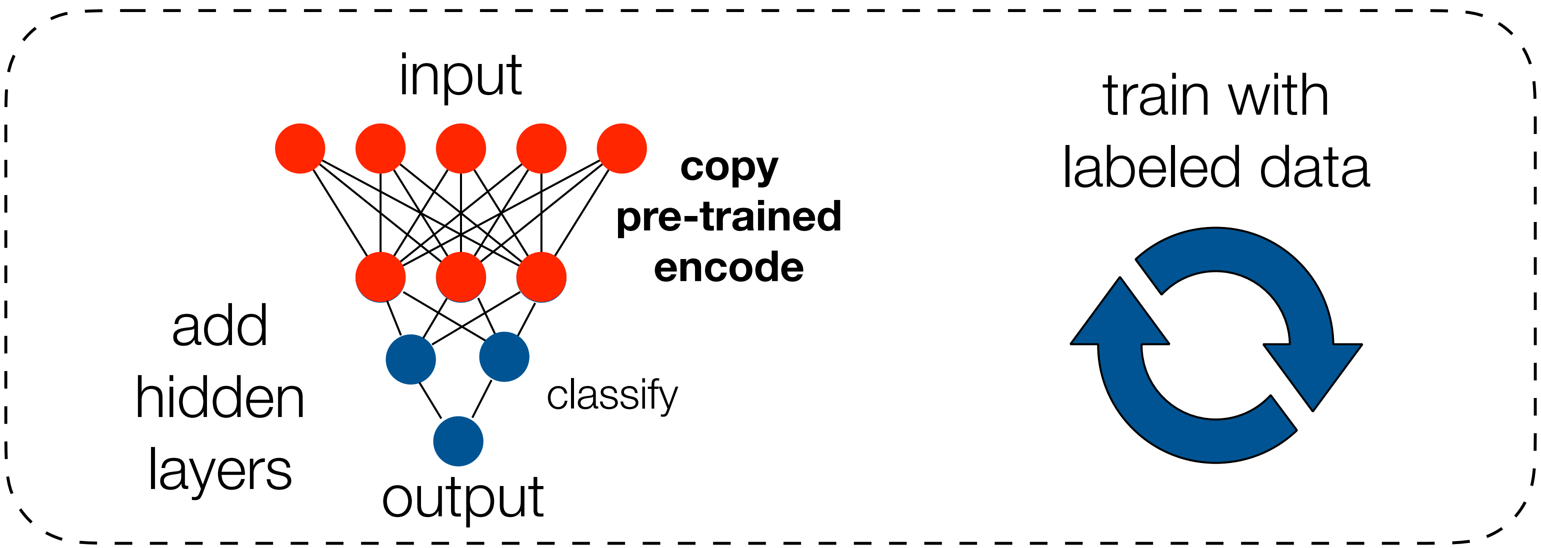


Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Pre-training: still in the long winter

- auto-encoding (a form of pre-training)



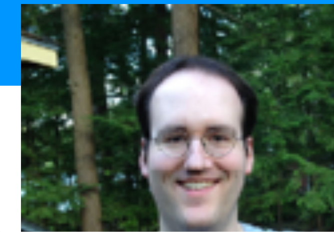


Still in the Long Winter

- 2009: Hinton's lab starts using GPUs, Also Andrew Ng
  - GPUs decrease training time by 70 fold…
- 2010: Hinton's and Ng's students go to internships with Microsoft, Google, IBM, and Facebook

Research at Google


Navdeep Jaitly
Google Brain Team


George Dahl
Google Brain Team


Abdel-rahman Mohamed
Microsoft Research
Redmond, Washington | Computer Software
Current Microsoft
Previous University of Toronto, IBM, Microsoft
Education University of Toronto

- Xbox Voice
- Android Speech Recognition
- IBM Watson
- DeepFace
- All of Baidu


1949, Hebb's Law
Close neuron fire together



1960, Widrow & Hoff
Adaline Network



1986, Rumelhart & Hinton
Back-propagation

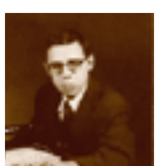

2003, Vapnik
Kernel SVMs

1940 1960 1980 2000

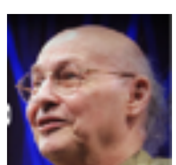
Period of Discovery | **First AI Winter** | **Golden Age of NN** | **2nd AI Winter**

1943, McCulloch & Pitts
Logic Gates of The Mind


1957, Rosenblatt
Perceptron



1969, Minsky & Papert
Linear Models are Doomed

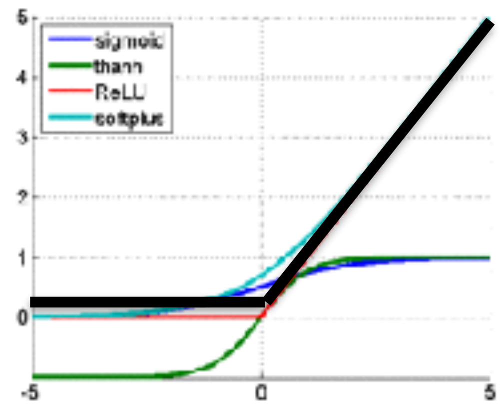



2001, Breiman
Random Forests

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Getting out of the long Winter

- 2011: Glorot and Bengio investigate more systematic methods for why past deep architectures did not work
  - **discover some interesting, simple fixes**: the type of neurons chosen and the selection of initial weights
  - do not require pre-training to get deep networks properly trained, just sparser representations and less complicated derivatives



ReLU: $f(x) = \max(0,x)$

$f'(x) = 1$ if $x>0$ else 0

1949, Hebb's Law
Close neuron fire together


1960, Widrow & Hoff
Adaline Network


1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

1940 — 1960 — 1980 — 2000 — 2020

Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind


1957, Rosenblatt
Perceptron

1969, Minsky & Papert
Linear Models are Doomed


2001, Breiman
Random Forests

2011, Bengio
Init and ReLU

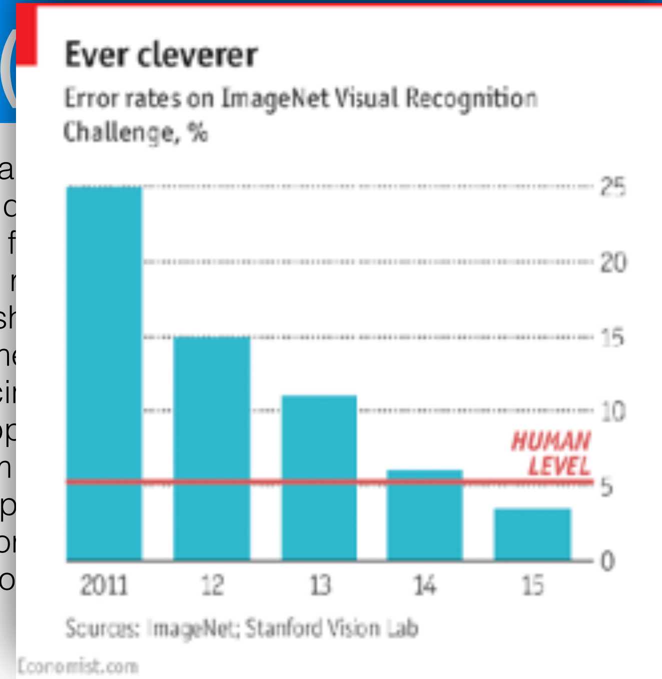
Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Machine Learning Timeline (

- **ImageNet competition occurs**
- **Second place**: 26.2% error rate
- **First place**:
  - From Hinton's lab, uses convolutional network with ReLU and dropout
  - 15.2% error rate
- Computer vision adopts deep learning with convolutional neural networks en mass

"I ha
hanc
last f
so I r
as sh
come
pacir
happ
from
skep
a cor
Visio

first
ie
),
ef
on
is
ust
t as
er



1949, Hebb's Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network

1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs

2012, Hinton, Fei-Fei Li
CNNs win ImageNet

1940 | 1960 | 1980 | 2000 | 2020

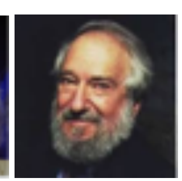
Period of Discovery | First AI Winter | Golden Age of NN | 2nd AI Winter | Age of Deep Learning

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron

1969, Minsky & Papert
Linear Models are Doomed

2001, Breiman
Random Forests

2011, Bengio
Init and ReLU

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Machine Learning Timeline (Neural Nets)

- 2013: Andrew Ng and Google (BrainTeam)
  - run unsupervised feature creation on YouTube videos (becomes computer vision benchmark)

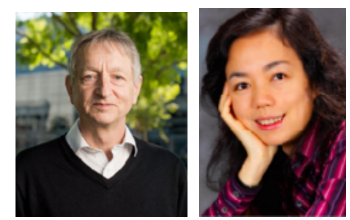
The work resulted in unsupervised neural net learning of an unprecedented scale - 16,000 CPU cores powering the learning of a whopping 1 billion weights. The neural net was trained on Youtube videos, entirely without labels, and learned to recognize the most common objects in those videos.

1949, Hebb's Law
Close neuron fire together

1960, Widrow & Hoff
Adaline Network

1986, Rumelhart & Hinton
Back-propagation

2003, Vapnik
Kernel SVMs



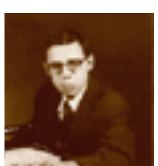
2012, Hinton, Fei-Fei Li
CNNs win ImageNet

1940 1960 1980 2000 2020

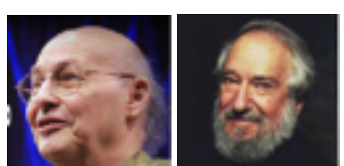
Period of Discovery | **First AI Winter** | **Golden Age of NN** | **2nd AI Winter** | **Age of Deep Learning**

1943, McCulloch & Pitts
Logic Gates of The Mind

1957, Rosenblatt
Perceptron



1969, Minsky & Papert
Linear Models are Doomed



2001, Breiman
Random Forests

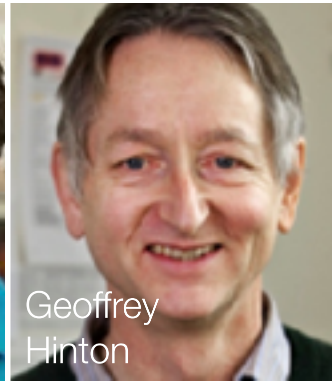
2011, Bengio
Init and ReLU

Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

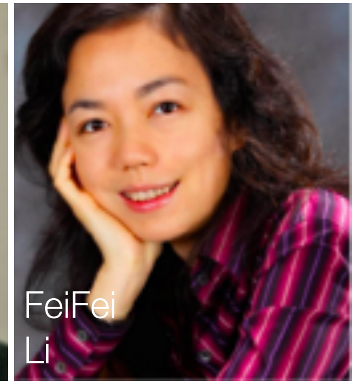
A summary of the Deep Learning people:

Stayed at Univ. Montreal
Advises IBM

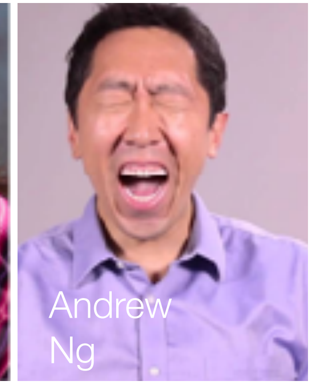
Heads Facebook AI Team



Univ. Toronto
Google



Stanford (HAI)
Former Chief Scien., AI/MLGoogle Cloud



Coursera
Baidu
Google

Stanford
Founded Coursera
MacArthur Genius

Made Deep Learning Instruction Accessible

deep learning

doi:10.1038/nature14539

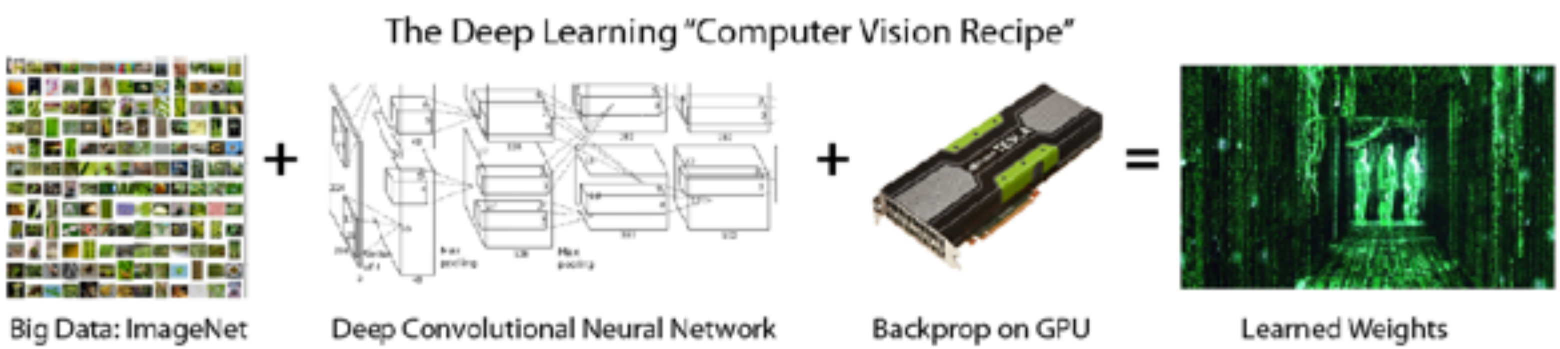
ning

Geoffrey Hinton[4,5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and

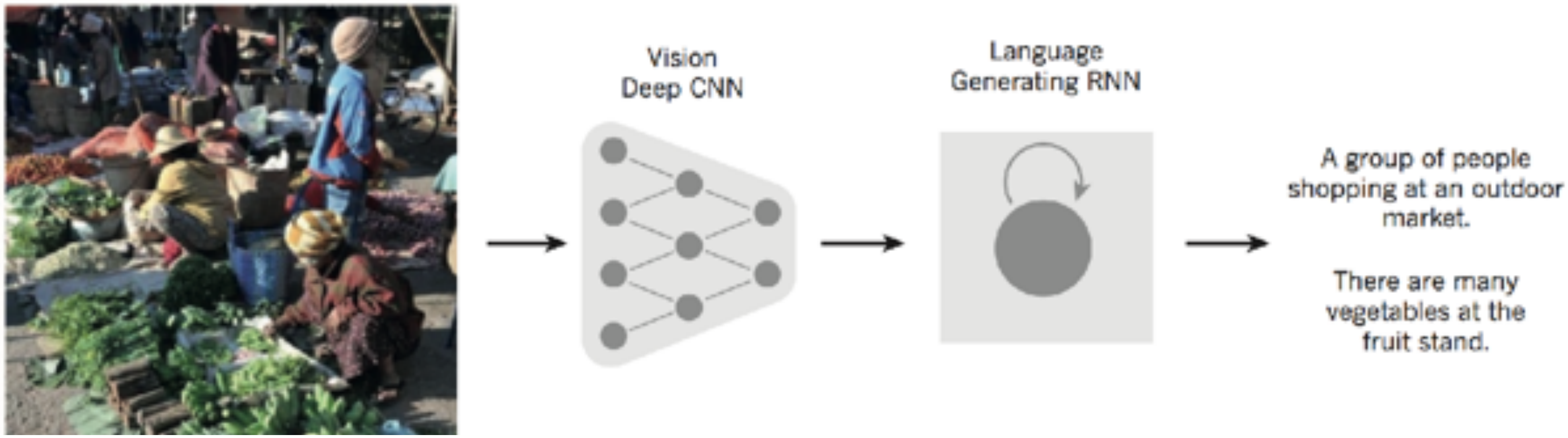
History of Deep Learning

- **Hinton** summarized what we learned in deep learning from the 2006 to present. Where we went wrong before present day:
  - labeled dataset were 1000s of times too small
  - computers were millions of times too slow
  - weights were initialized in stupid ways
  - we used the wrong non-linearities
- Or **Larson's Laws**:
  - use a GPU when possible, init weights for consistent gradient magnitude, ReLU/SiLU where it makes sense (like in early feedforward layers), clip the gradient magnitude, and use adaptive learning to learn more quickly!



Read this: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

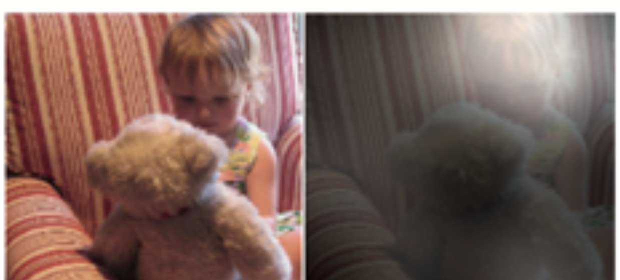
Famous examples:



A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

End of Session

- Next Time:
  - Introduction to TensorFlow
  - Wide and Deep Networks