Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

Professor Eric Larson

Neural Network, Wrap Up

Class Logistics and Agenda

- Logistics
  - hiring undergraduates!
  - grading update
- Agenda:
  - Demo and Review
  - Universality
  - Parameter Searching
  - Statistical Testing
  - Lab 4 Town Hall

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

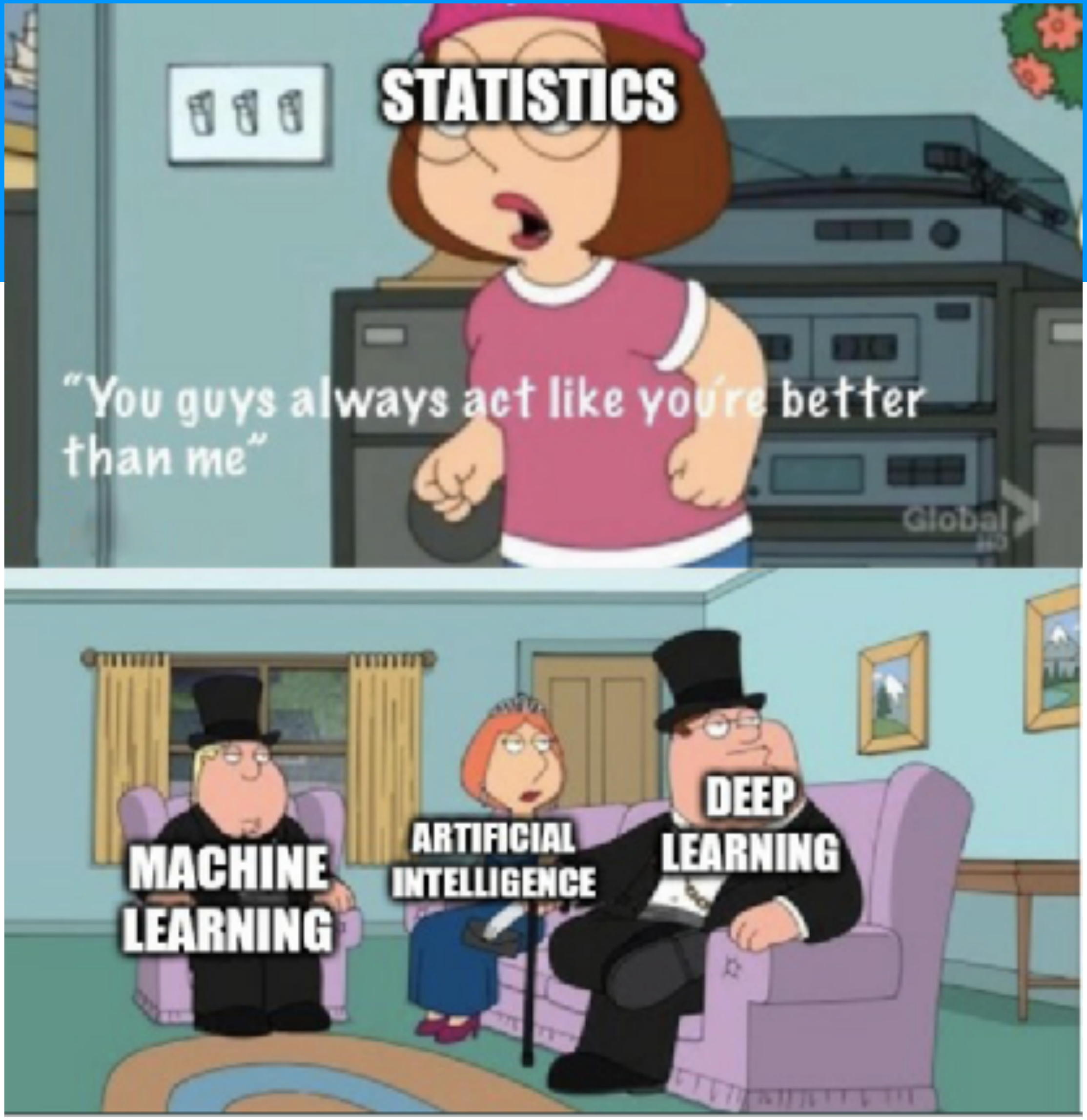
Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Sequential Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models

ConceptNet
Case studies

Review


STATISTICS
"You guys always act like you're better than me"
Global
MACHINE LEARNING
ARTIFICIAL INTELLIGENCE
DEEP LEARNING

Review

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \rho_k$$

- Cross entropy

$$\mathbf{V}^{(2)} = \mathbf{A}^{(3)} - \mathbf{Y}$$

new final layer update

- Momentum

$$\rho_k = \alpha \nabla J(\mathbf{W}_k) + \beta \nabla J(\mathbf{W}_{k-1})$$

- Nesterov's Momentum

$$\rho_k = \underbrace{\beta \nabla J \left(\mathbf{W}_k + \alpha \nabla J(\mathbf{W}_{k-1}) \right)}_{\text{step twice}} + \alpha \nabla J(\mathbf{W}_{k-1})$$

- Mini-batching

←all data→

	batch 1	batch 2	batch 3	batch 4	batch 5	batch 6	batch 7	batch 8	batch 9
Epoch 1									
Epoch 2									
Epoch 3									
Epoch 4									
...									

shuffle ordering each epoch and update W's after each batch

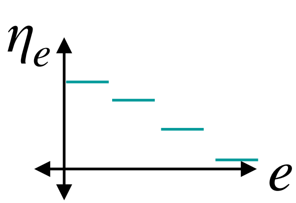
- Learning rate adjustment (eta)

Stair step

$$\eta_e = \eta_0 \cdot d^{\lfloor \frac{e}{e_d} \rfloor}$$

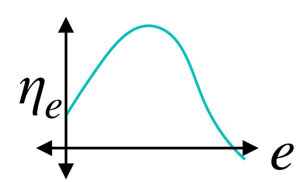
e_d epochs between reductions

$0 < d < 1$

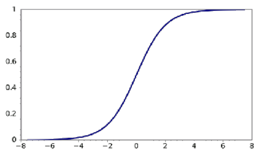
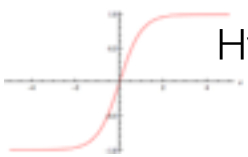
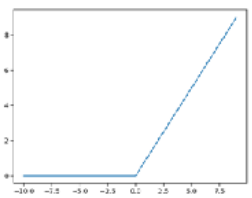
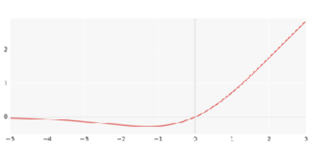


$$\eta_e = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{e}{e_{max}}\pi\right)\right)$$

Cosine Schedule



Review: Activations Summary

	Definition	Derivative	Weight Init *(Uniform Bounds)*
 Sigmoid	$\phi(z) = \frac{1}{1+e^{-z}}$	$\nabla\phi(z) = a(1-a)$	$w_{ij}^{(L)} \sim \pm 4\sqrt{\frac{6}{n^{(L)}+n^{(L+1)}}}$ $b^{(l)} = [-2]$
 Hyperbolic Tangent	$\phi(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$\nabla\phi(z) = \frac{4}{(e^{z}+e^{-z})^2}$	$w_{ij}^{(L)} \sim \pm\sqrt{\frac{6}{n^{(L)}+n^{(L+1)}}}$ $b^{(l)} = [0]$
 ReLU	$\phi(z) = \begin{cases} z, \text{ if } z > 0 \\ 0, \text{ else} \end{cases}$	$\nabla\phi(z) = \begin{cases} 1, \text{ if } z > 0 \\ 0, \text{ else} \end{cases}$	$w_{ij}^{(L)} \sim \pm\sqrt{2}\sqrt{\frac{6}{n^{(L)}+n^{(L+1)}}}$
 SiLU	$\phi(z) = \frac{z}{1+e^{-z}}$	$\nabla\phi(z) = \phi(z) + \sigma(z)\cdot(1-\phi(z))$	$b^{(l)} = [0]$

Review of Adaptive Strategies $\mathbf{W}_{k+1} = \mathbf{W}_k - \eta \cdot \rho_k$

Adjust each element of gradient by the steepness (for each layer):

- AdaGrad

$$\rho_k = \frac{1}{\sqrt{\mathbf{G}_k + \epsilon}} \odot \nabla J(\mathbf{W}_k)$$

all operations are per element

where

$$\mathbf{G}_k = \gamma \cdot \mathbf{G}_{k-1} + \nabla J(\mathbf{W}_k) \odot \nabla J(\mathbf{W}_k)$$

- RMSProp

$$\rho_k = \frac{1}{\sqrt{\mathbf{V}_k + \epsilon}} \odot \nabla J(\mathbf{W}_k)$$

all operations are per element

$$\mathbf{G}_k = \nabla J(\mathbf{W}_k) \odot \nabla J(\mathbf{W}_k)$$

$$\mathbf{V}_k = \gamma \cdot \mathbf{V}_{k-1} + (1-\gamma) \cdot \mathbf{G}_k$$

- AdaDelta

$$\rho_k = \frac{\mathbf{M}_k}{\sqrt{\mathbf{V}_k + \epsilon}}$$

all operations are per element

$$\mathbf{M}_{k+1} = \gamma \cdot \mathbf{M}_k + (1-\gamma) \cdot \nabla J(\mathbf{W}_k)$$

- AdaM

update momentum

$$\mathbf{M}_{k+1} \leftarrow \beta_1 \cdot \mathbf{M}_k + (1-\beta_1) \cdot \nabla J(\mathbf{W}_k)$$

$$\hat{\mathbf{M}}_k \leftarrow \frac{\mathbf{M}_k}{(1-[\beta_1]^k)}$$

normalizer momentum

$$\mathbf{V}_{k+1} \leftarrow \beta_2 \cdot \mathbf{V}_k + (1-\beta_2) \cdot \nabla J(\mathbf{W}_k) \odot \nabla J(\mathbf{W}_k)$$

$$\hat{\mathbf{V}}_k \leftarrow \frac{\mathbf{V}_k}{(1-[\beta_2]^k)}$$

full update

$$\mathbf{W}_{k+1} \leftarrow \mathbf{W}_k - \eta \cdot \frac{\hat{\mathbf{M}}_k}{\sqrt{\hat{\mathbf{V}}_k + \epsilon}}$$

exploitation, boosting

Demo

08a. Practical_NeuralNetsWithBias.ipynb

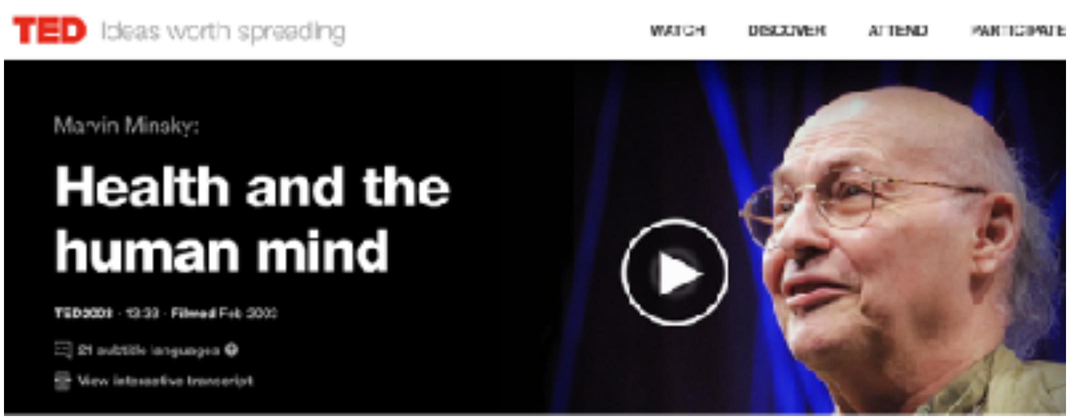
~~Momentum~~
~~Learning Rate Adaptation~~
~~Cross Entropy~~
~~Smarter Weight Initialization~~
~~Adaptive training with AdaGrad~~
ReLU Nonlinearities
Sklearn Comparison

Revisiting Universality (if time)

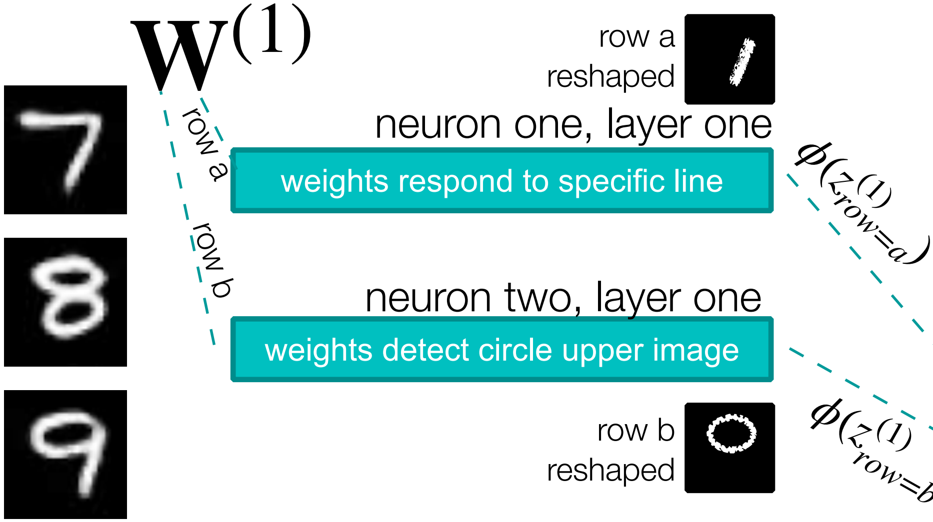
- Neural networks can separate any data through multiple layers. The true realization of Rosenblatt:

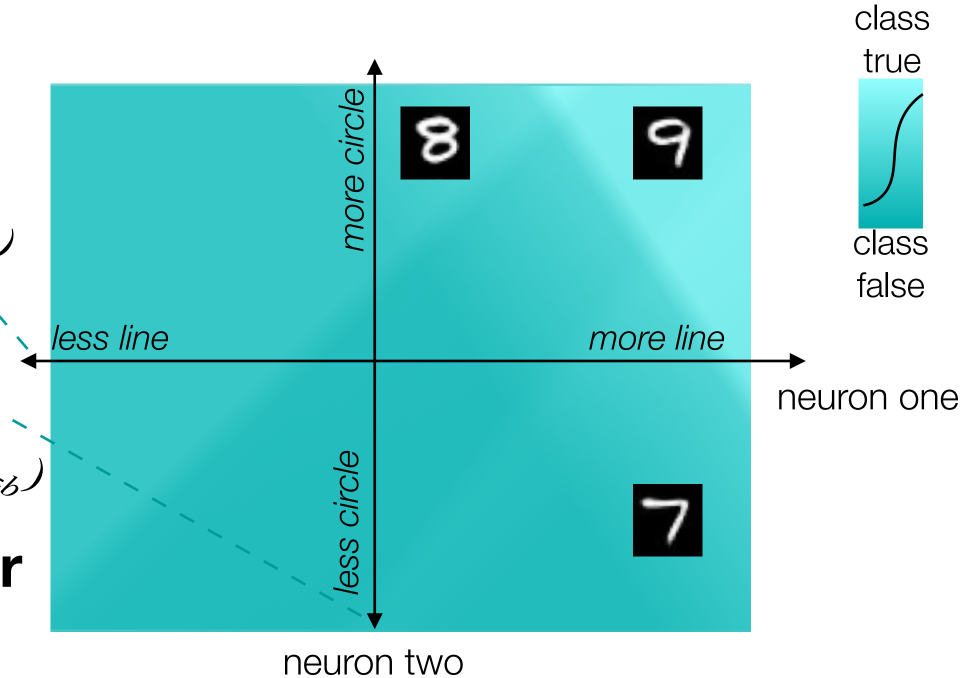
  "Given an elementary α-perceptron, a stimulus world W, and any classification C(W) for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to C(W) in finite time…"

- **Universality**: No matter what function we want to compute, we know that there is a neural network which can do the job.



Universality

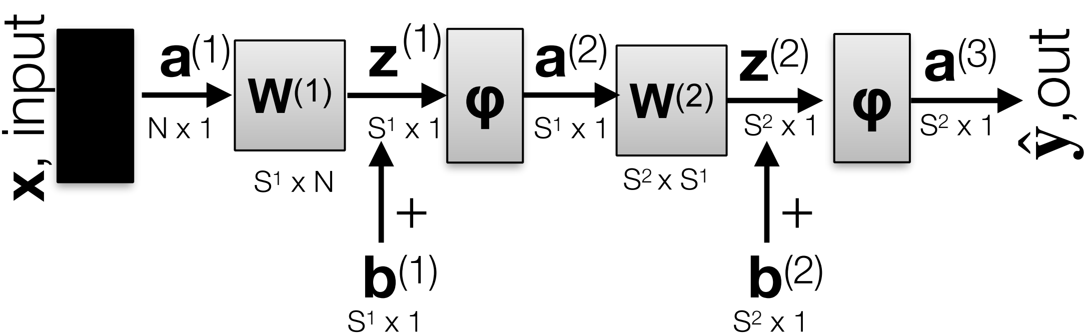




$\mathbf{W}^{(1)}$ is a **featurization layer**

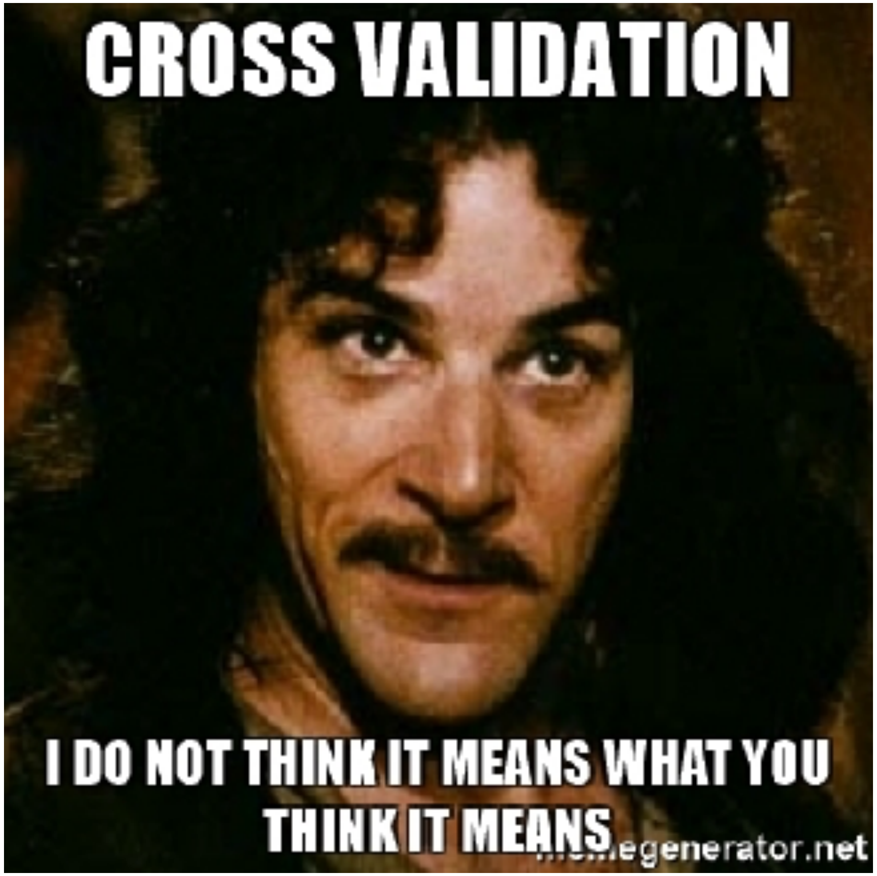
adding layers and convolution help obtain desired behavior…

$\mathbf{W}^{(2)}$ is **classification layer,** like one versus all logistic regression



- One nonlinear hidden layer with an output layer can ***perfectly train any problem with enough data***, but might be memorizing…
- … could be better to have ***even more layers*** for more generalizing features

Grid Searching



Review: Grid Searching

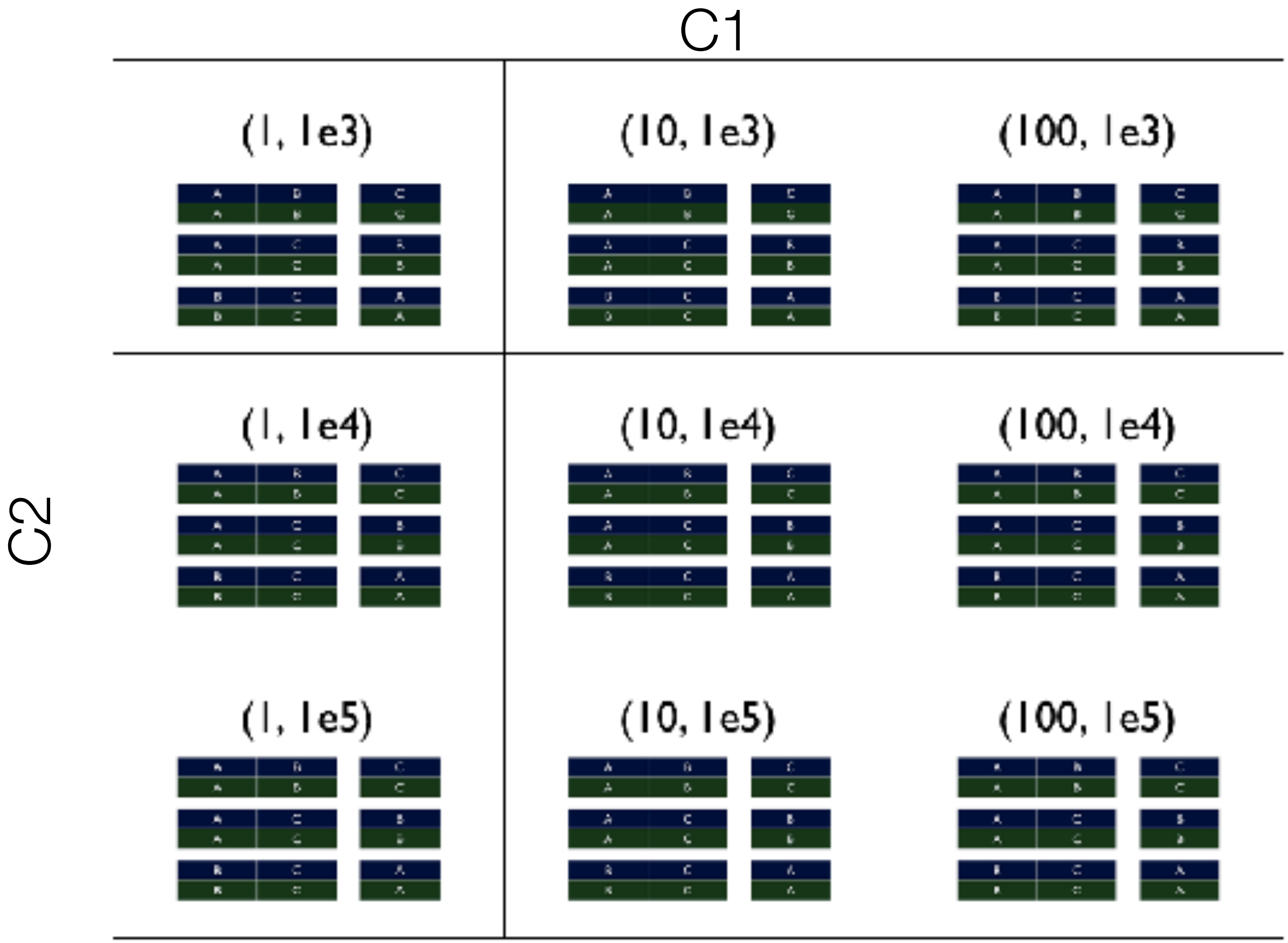
Trying to find the best parameters

`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)

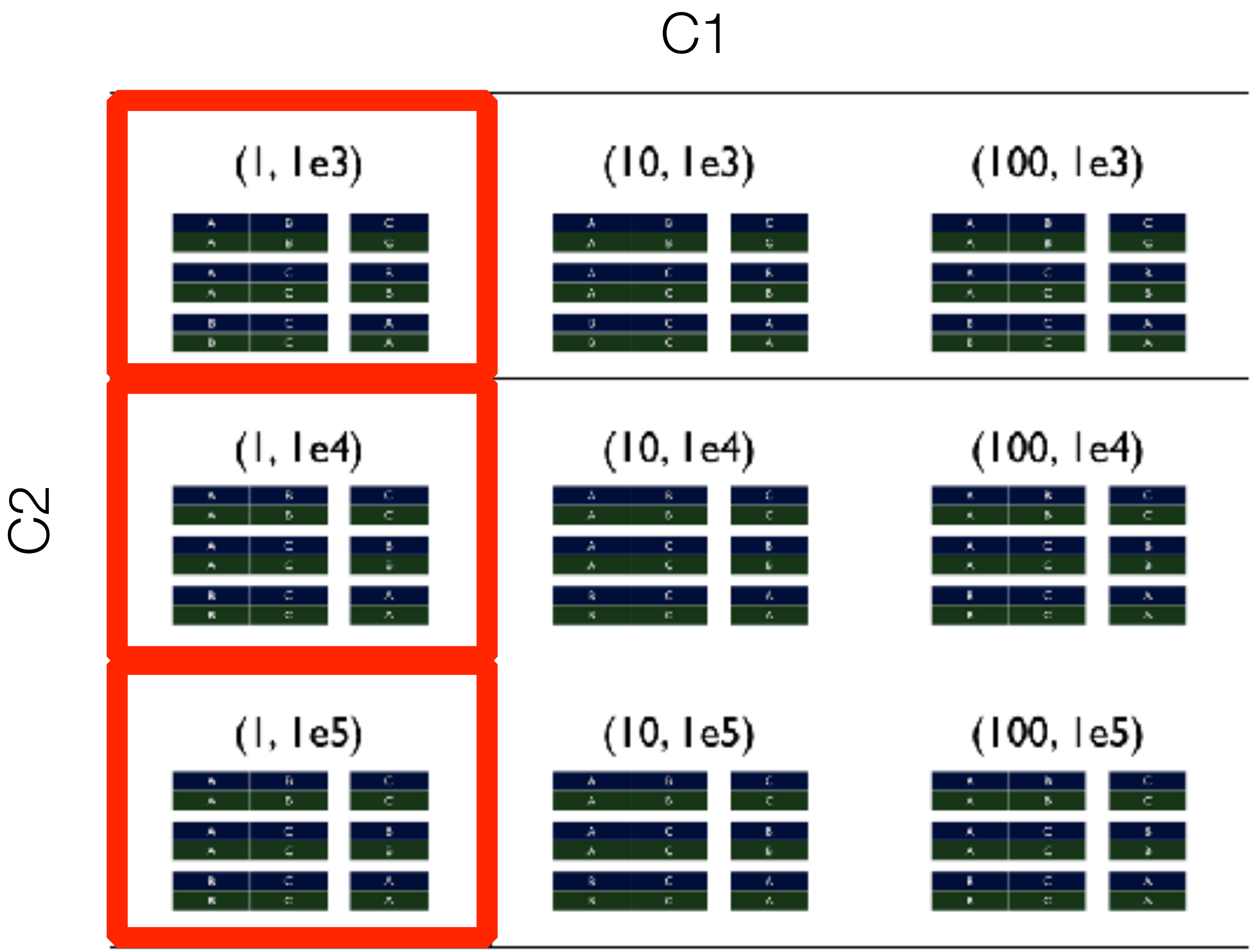
Review: Grid Searching

For each value, want to run cross validation…

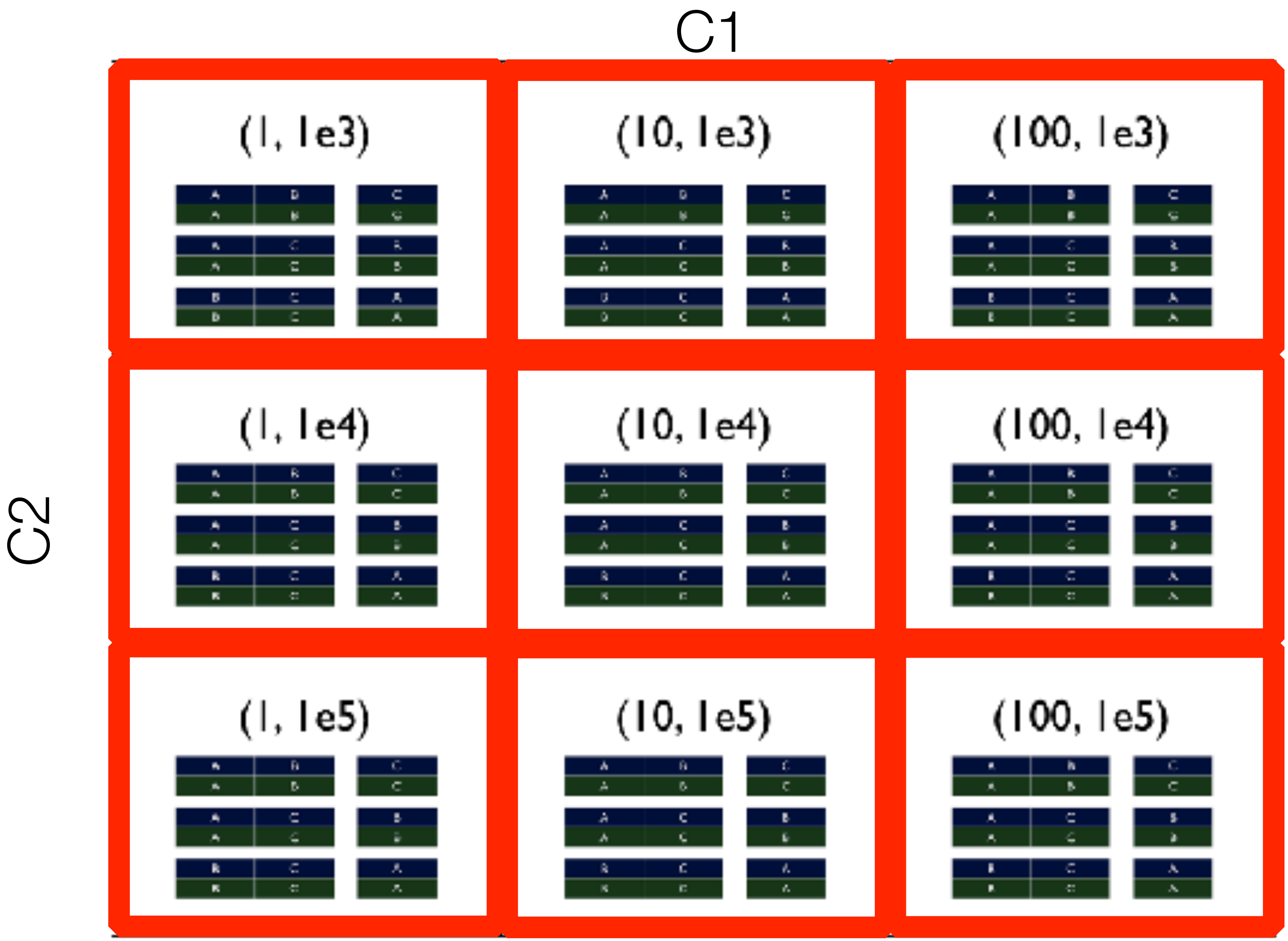


Review: Grid Searching

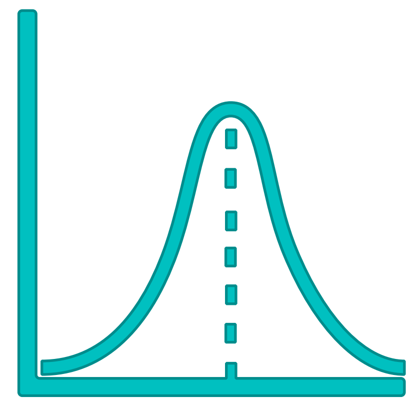
Could perform iteratively



or at random…



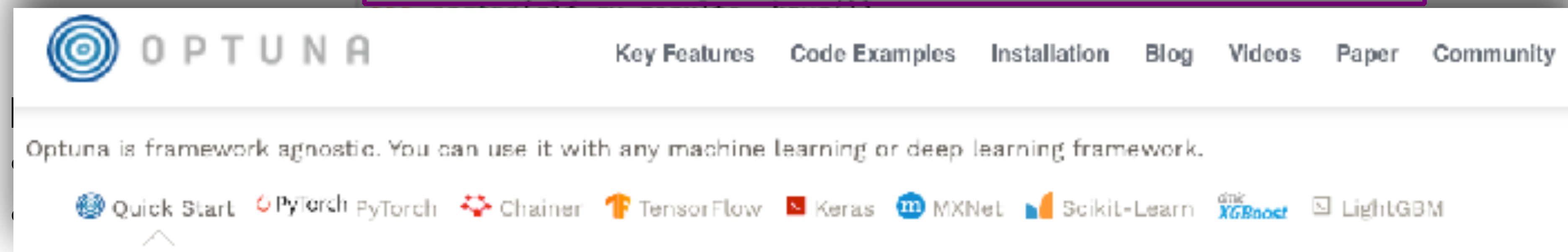
If random can also draw at random from a distribution!



C1 drawn from $\mathcal{N}(\mu = 50, \sigma = 20)$

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')})
```



values, sampled

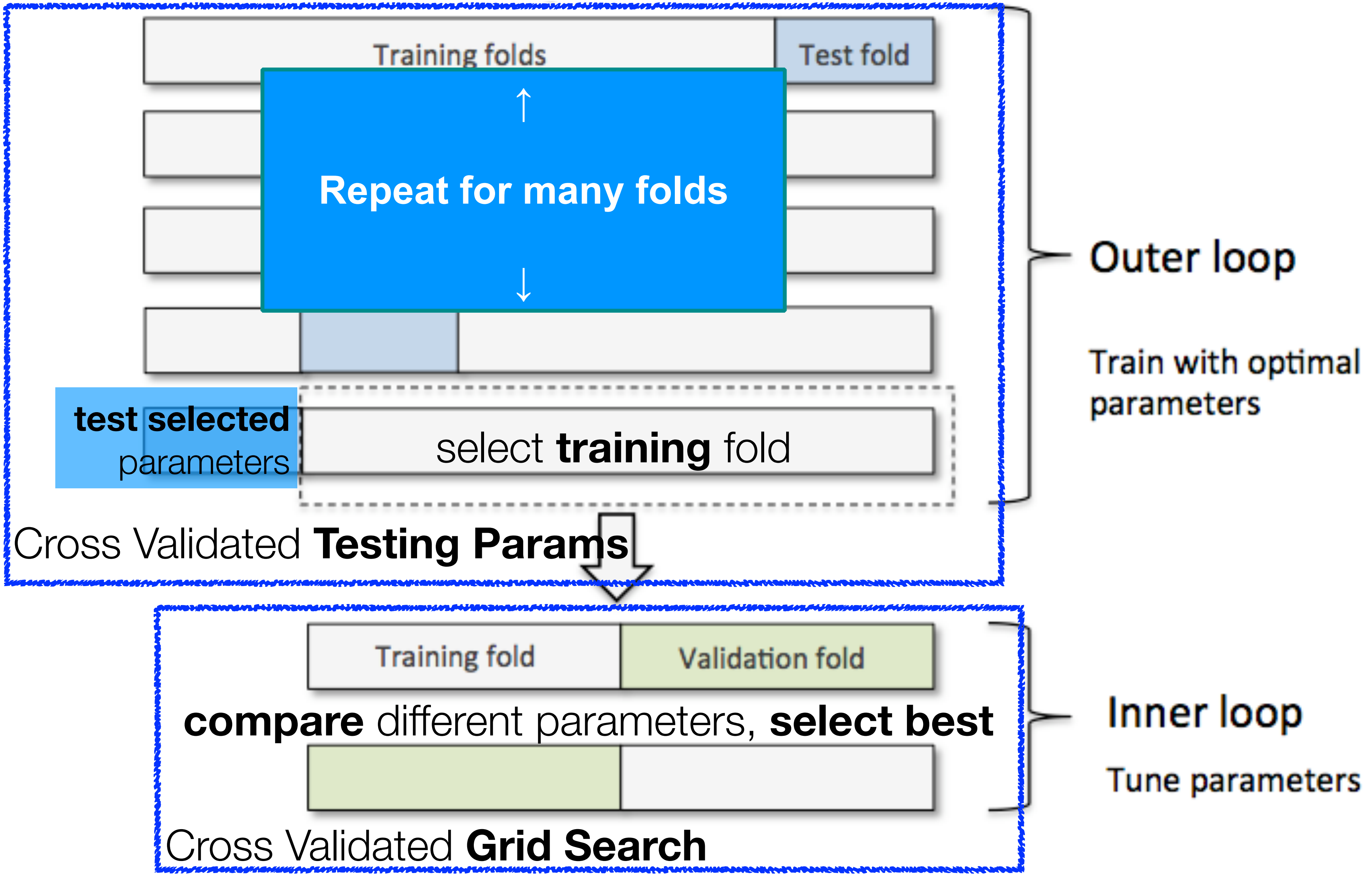
```
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import RandomizedSearchCV
>>> from scipy.stats import uniform
>>> iris = load_iris()
>>> logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
...                               random_state=0)
>>> 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: Self Test

- Using the grid search parameters and testing on the same set…
- Is this **data snooping**?
  - **A. True**, this is snooping because it uses test set to define parameters
  - **B. True**, this is snooping because we can no longer reliably define the expected performance on new data

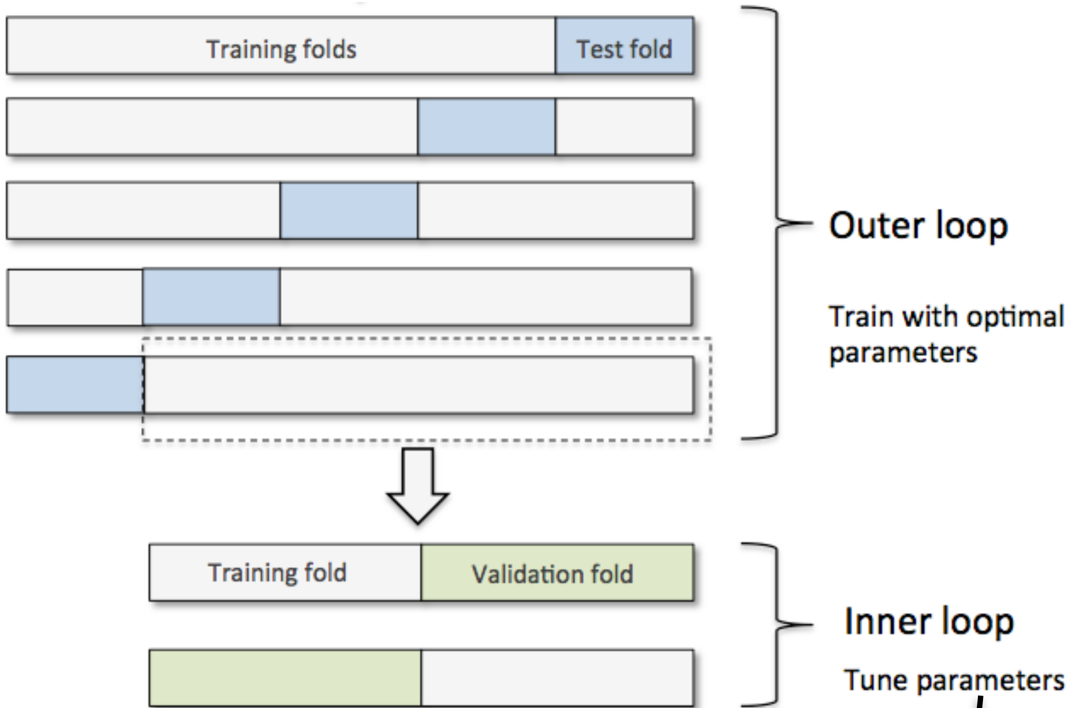
How can we define expected performance when using cross validation in a grid search?

Review: Nested Cross Validation



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

Review: Nested Cross Validation: Hyper-parameters



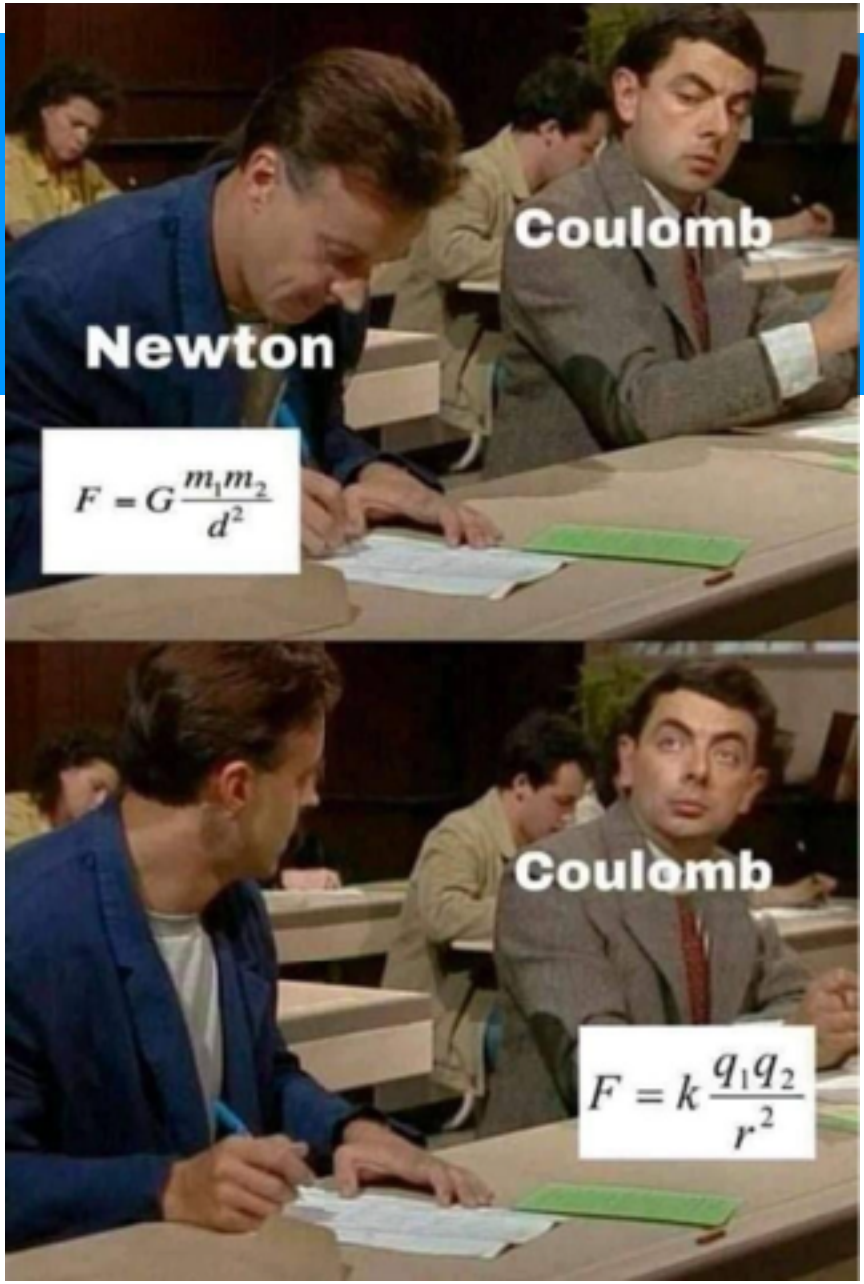

```python
gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=2)
```

```python
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

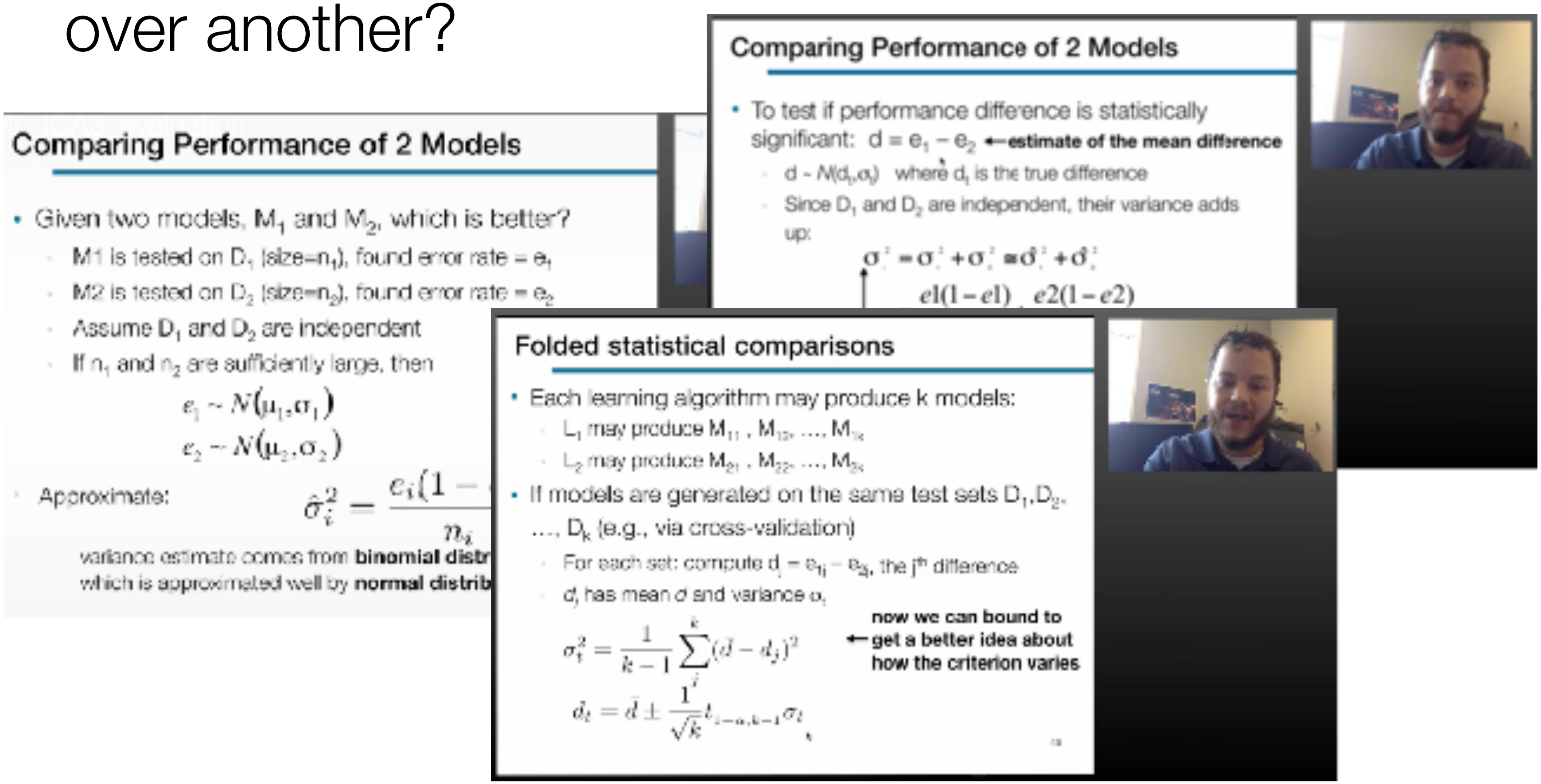
- **What is the end goal of nested cross-validation?**
  - A. To determine hyper parameters
  - B. To estimate generalization performance
  - C. To estimate generalization performance when performing hyper parameter tuning
  - D. To estimate the variation in tuned hyper parameters



Statistical Tests

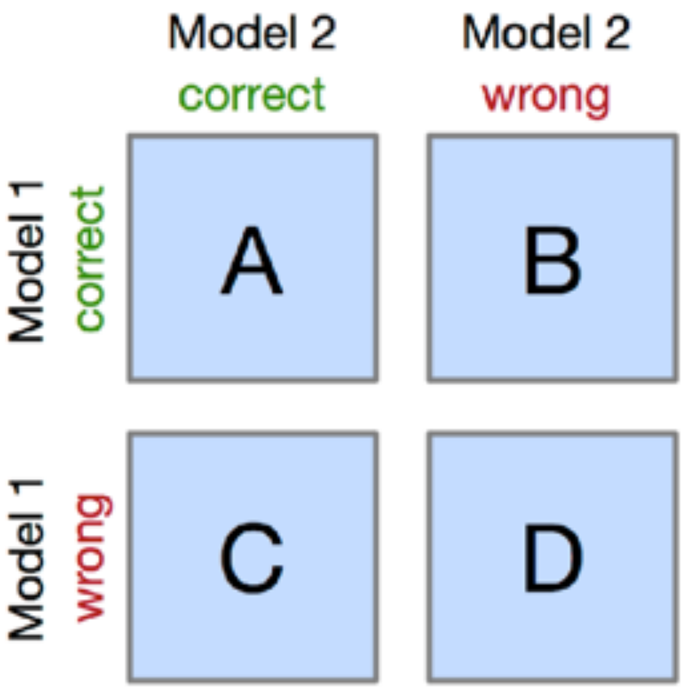
Flipped Module, Model Comparison

- How do we compare two (or more) trained models to one another?
- Are models different enough to prefer one model over another?



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

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

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

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.

Self Test:

- You have trained **three different models** on prediction of child poverty ratings. Each model is trained on the same data and **tested on the same single split** of the data.

- **Can you use a McNemar test for selecting a model?**
  - **A. Yes.** McNemar testing can be used for any testing of any model without any assumptions.
  - **B. Yes.** McNemar testing can be applied pairwise for the three models.
  - **C. No.** McNemar testing cannot be used for more then two models.
  - **D. No.** McNemar testing can only tell if the models are different, we still cannot tell which one is best

Town Hall

Next Time:
Deep Learning