[544] PyTorch Machine Learning

Tyler Caraza-Harter

Outline

Machine Learning, Major Ideas

Deep Learning

PyTorch

- Calculations at DAGs
- Machine Learning as Optimization

Machine Learning, Major Ideas

Categories of Machine Learning:

- **Reinforcement learning**: agent makes series of actions to maximize reword
- **Unsupervised learning**: looking for generate patterns
- **Supervised learning:** train models to predict unknowns (today)

Models are functions that return predictions:

```
def my_model(some_info):
    ...
    return some_prediction
```

→ numeric (1, 2, 3) is "regression"

→ categorical (A, B, C) is "classification"

Example:

```
def weather_forecast(temp_today, temp_yesterday):
    ...
    return temp_tomorrow
```

Machine Learning, Major Ideas

Categories of Machine Learning:

- **Reinforcement learning**: agent makes series of actions to maximize reword
- **Unsupervised learning**: looking for generate patterns
- **Supervised learning:** train models to predict unknowns (today)

Models are functions that return predictions:

```
def my_model(some_info):
    ...
    return some_prediction
```

computation usually involves some calculations (multiply, add) with various numbers (parameters). Training is finding parameters that result in good predictions for known training data

Example:

```
def weather_forecast(temp_today, temp_yesterday):
    ...
    return temp_tomorrow
```

Learning from Data

	x1	x2	y
0	2	8	5
1	9	2	6
2	4	1	0
3	7	9	7
4	2	2	3
5	3	4	3
6	3	5	9
7	7	1	4
8	6	6	3
9	4	3	?
10	1	2	?
11	2	9	?

how can the cases where we DO know y help us predict the cases where we do not?

Learning from Data

	x1	x2	y
0	2	8	5
1	9	2	6
2	4	1	0
3	7	9	7
4	2	2	3
5	3	4	3
6	3	5	9
7	7	1	4
8	6	6	3
9	4	3	?
10	1	2	?
11	2	9	?

train (rows 0–2)

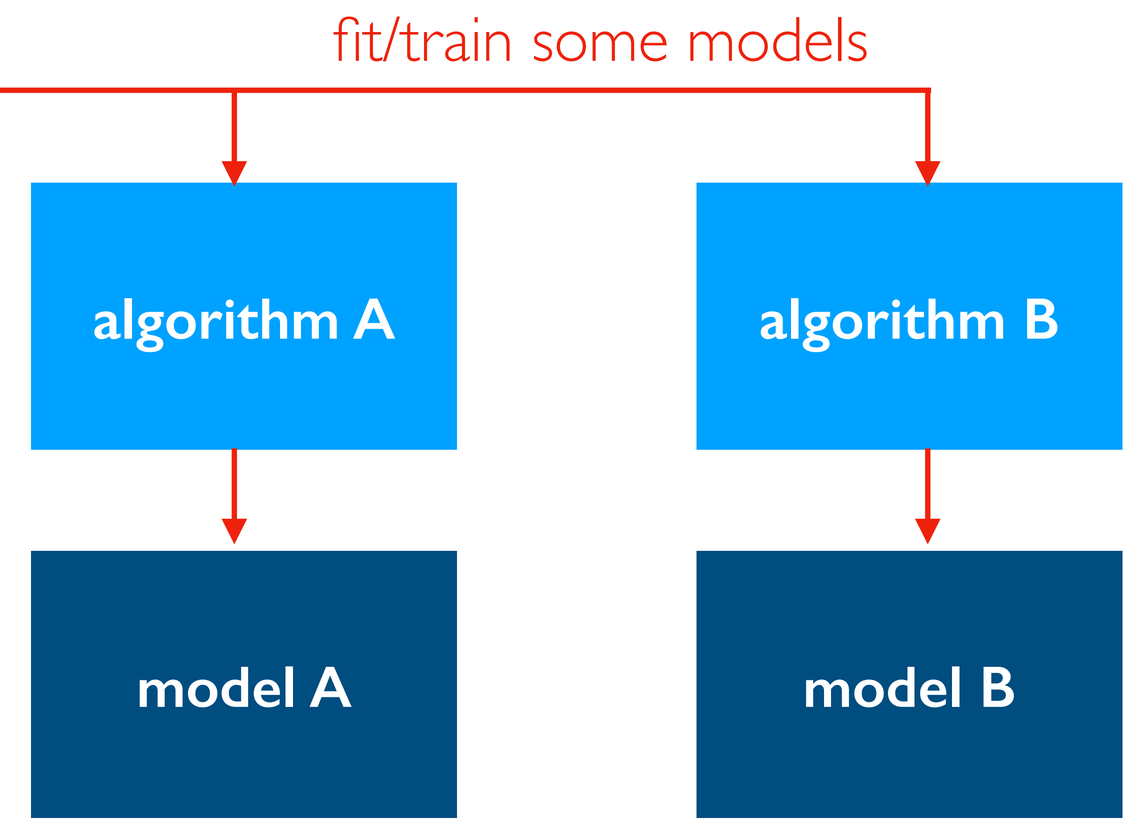
validation (rows 3–5)

test (rows 6–8)

random split

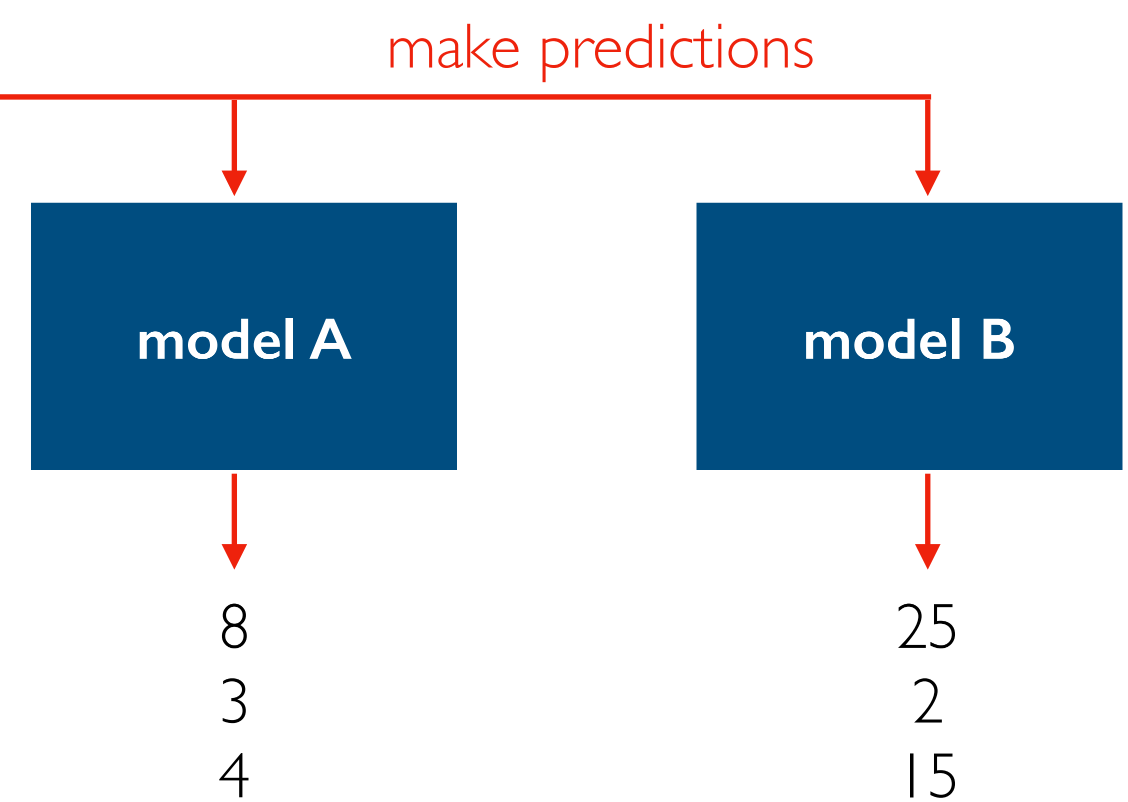
Learning from Data

		x1	x2	y
train	0	2	8	5
train	1	9	2	6
train	2	4	1	0
validation	3	7	9	7
validation	4	2	2	3
validation	5	3	4	3
test	6	3	5	9
test	7	7	1	4
test	8	6	6	3
	9	4	3	?
	10	1	2	?
	11	2	9	?



Learning from Data

		x1	x2	y
	0	2	8	5
train	**1**	9	2	6
	2	4	1	0
	3	7	9	7
validation	**4**	2	2	3
	5	3	4	3
	6	3	5	9
test	**7**	7	1	4
	8	6	6	3
	9	4	3	?
	10	1	2	?
	11	2	9	?



Learning from Data

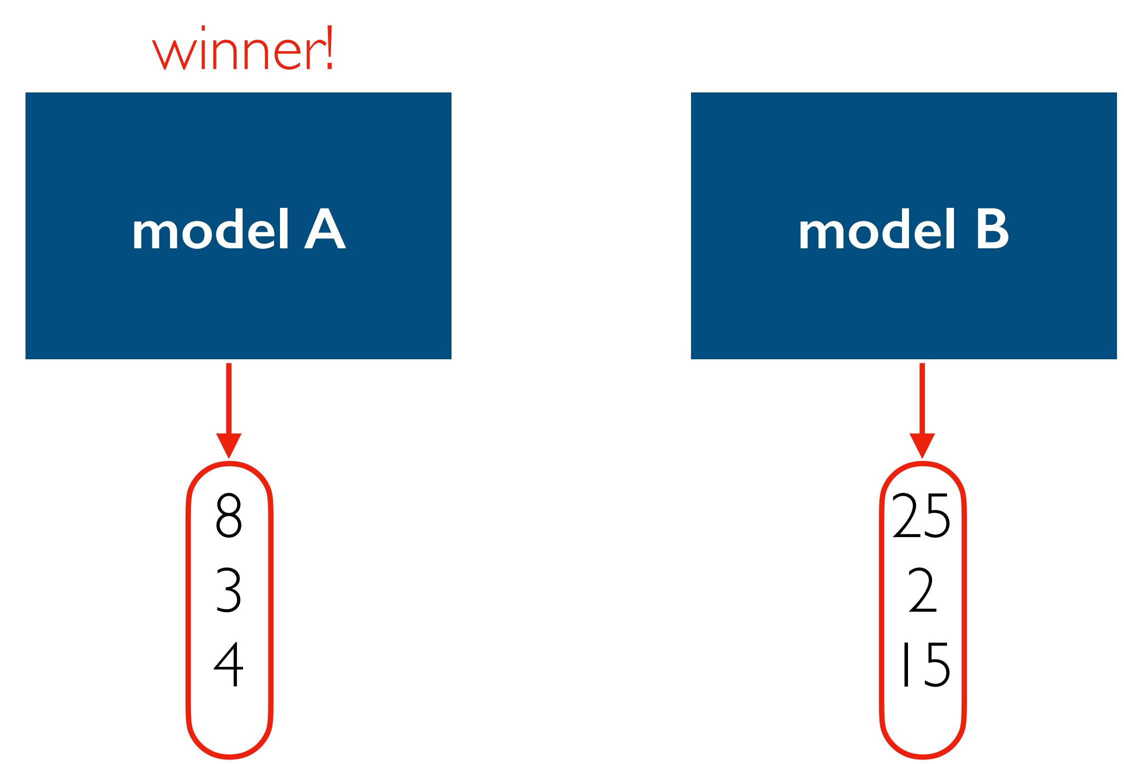
	x1	x2	y
0	2	8	5
1	9	2	6
2	4	1	0
3	7	9	7
4	2	2	3
5	3	4	3
6	3	5	9
7	7	1	4
8	6	6	3
9	4	3	?
10	1	2	?
11	2	9	?

train (rows 0–2)

validation (y of rows 3–5)

test (rows 6–8)

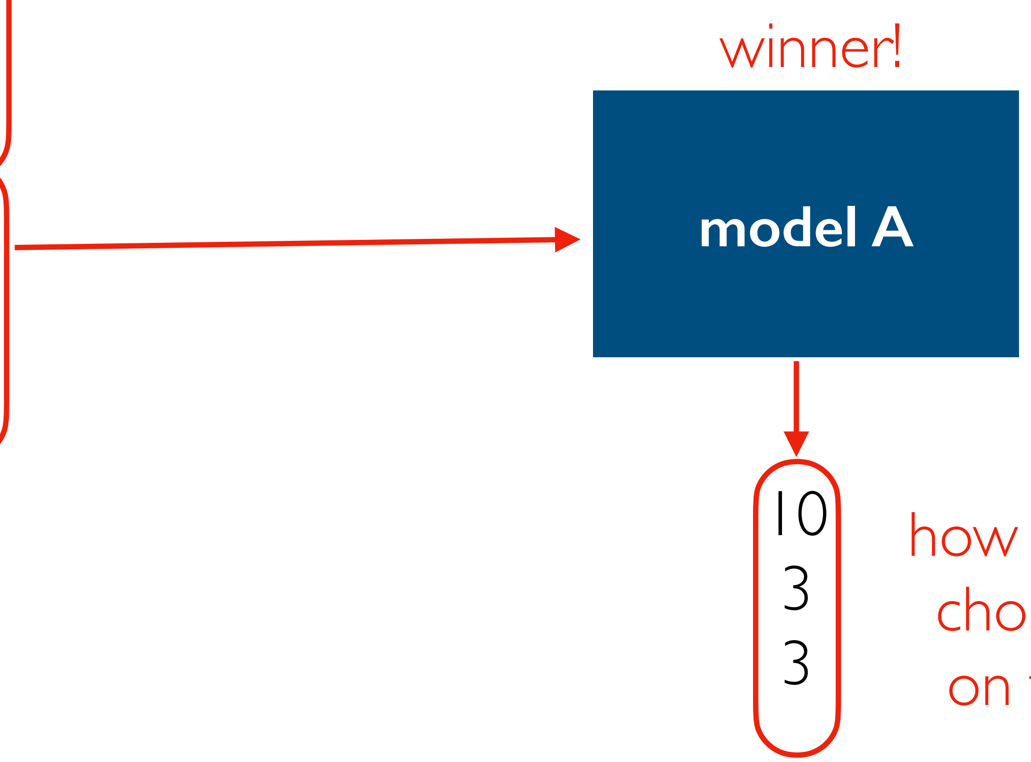
which model predicts better?



Learning from Data

why might the winning model do worse on the test data than the validation data?

		x1	x2	y
train	**0**	2	8	5
	1	9	2	6
	2	4	1	0
validation	**3**	7	9	7
	4	2	2	3
	5	3	4	3
test	**6**	3	5	9
	7	7	1	4
	8	6	6	3
	9	4	3	?
	10	1	2	?
	11	2	9	?

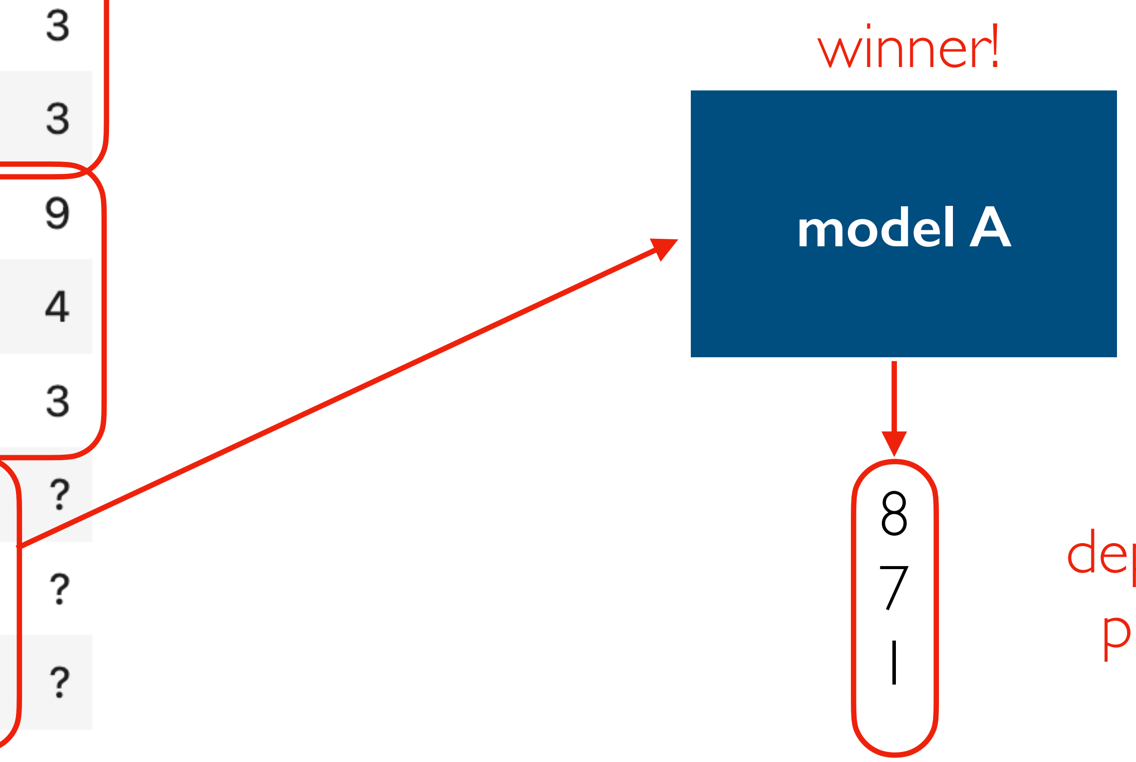


how good does the chosen model do on the test data?

models that do good on train data but bad on validation/test data have "overfitted"

Learning from Data

		x1	x2	y
train	**0**	2	8	5
	1	9	2	6
	2	4	1	0
validation	**3**	7	9	7
	4	2	2	3
	5	3	4	3
test	**6**	3	5	9
	7	7	1	4
	8	6	6	3
	9	4	3	?
	10	1	2	?
	11	2	9	?



deploy the model. Use it for predicting real unkwowns!

Outline

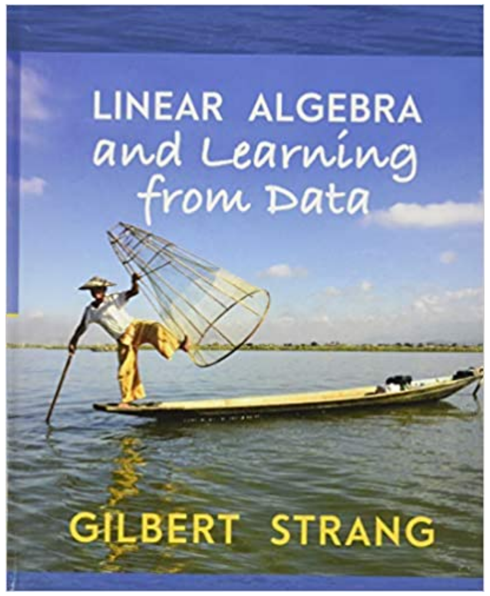
Machine Learning, Major Ideas

Deep Learning

PyTorch

- Calculations at DAGs
- Machine Learning as Optimization

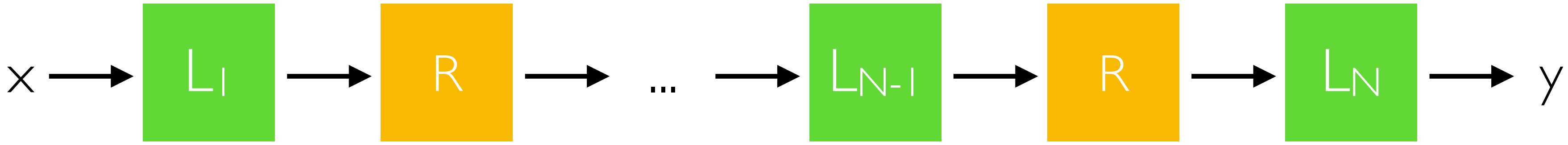
Deep Learning



Know x (maybe a vector of numbers), want to predict y.

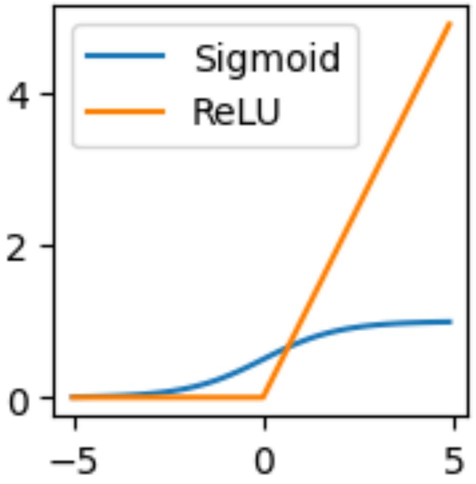
$y = \text{model}(x) = L_N(R(L_{N-1}(R(...(L_1(x))))))$

function nesting = a pipeline



$L_k(x) = Wx + b$

R:



Outline

Machine Learning, Major Ideas

Deep Learning

PyTorch

- Calculations at DAGs
- Machine Learning as Optimization

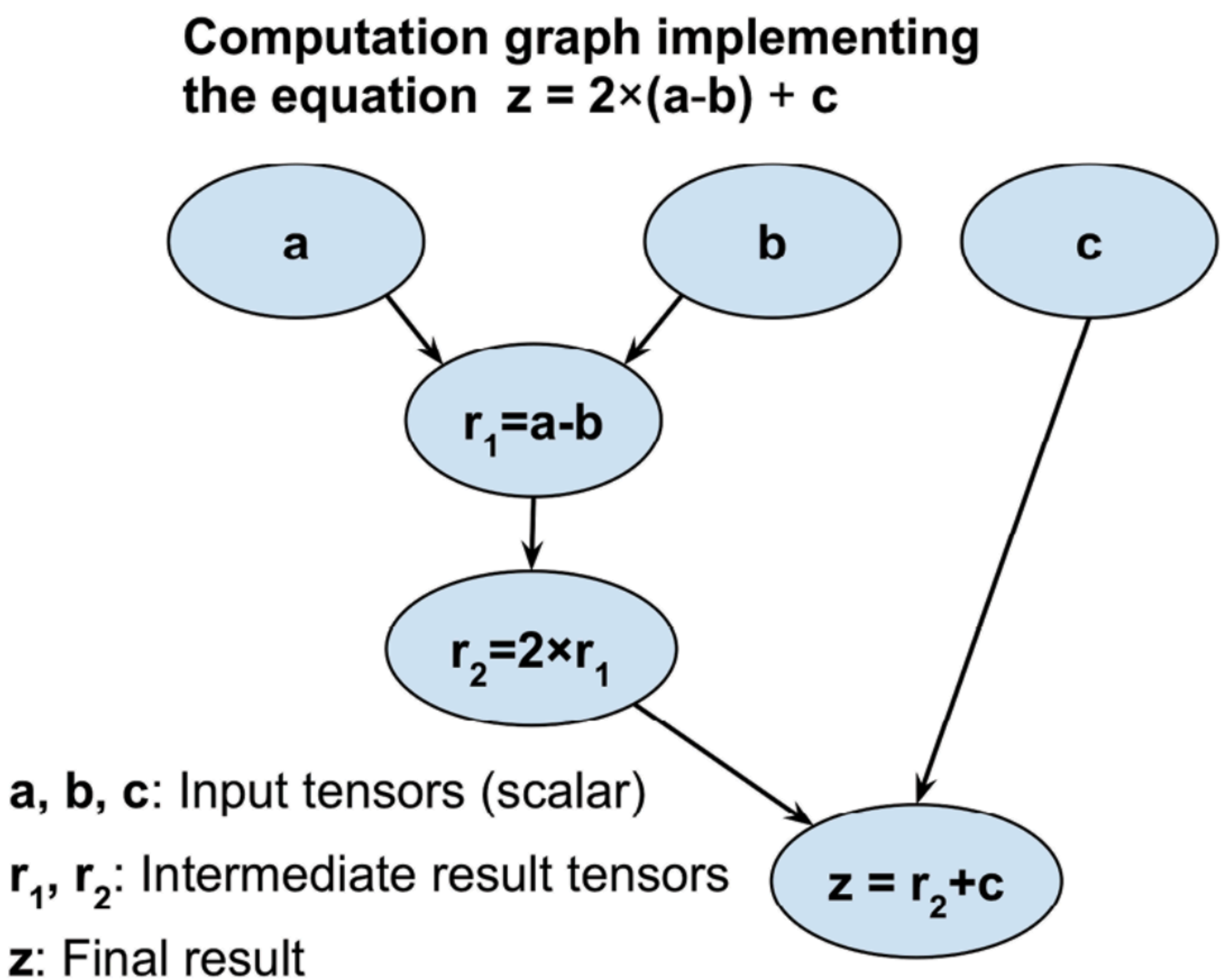
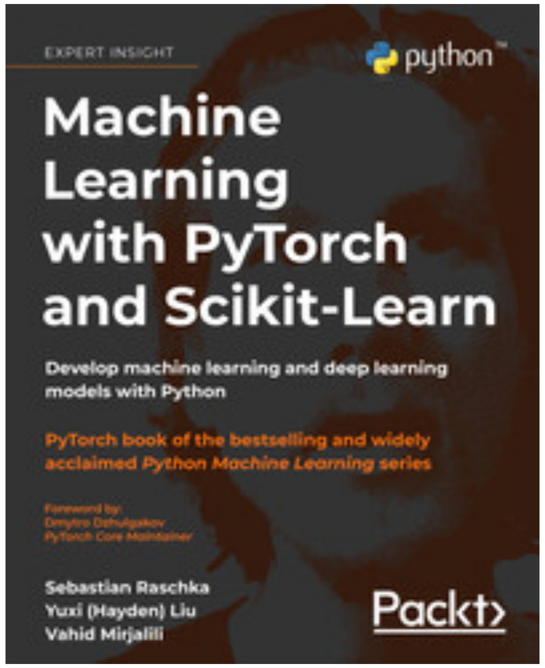
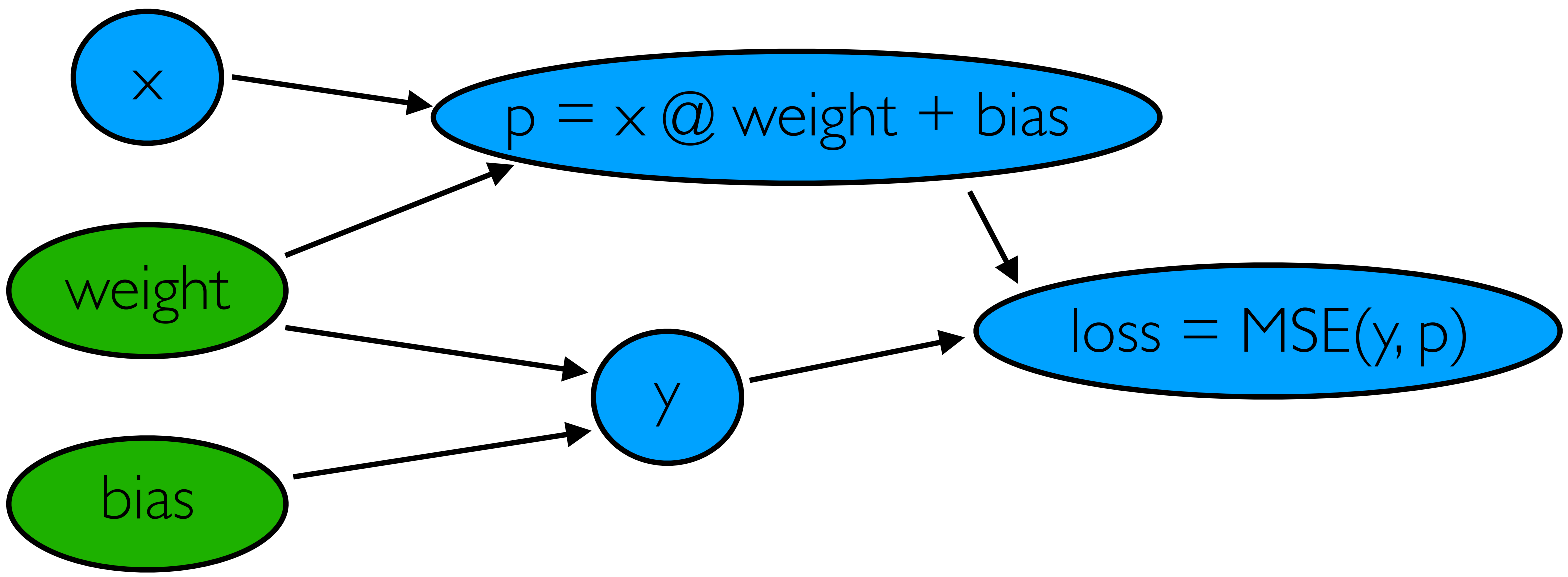


Figure 13.1: How a computation graph works



PyTorch can calculate how small changes in one variable in the DAG impacts another.
Example: if b *increases* by 0.001, z will *decrease* by 0.002. The **gradient** of z with respect to b is -2.

Optimization: if we want z to be large, decreasing b a little (how much?) is probably a good idea.



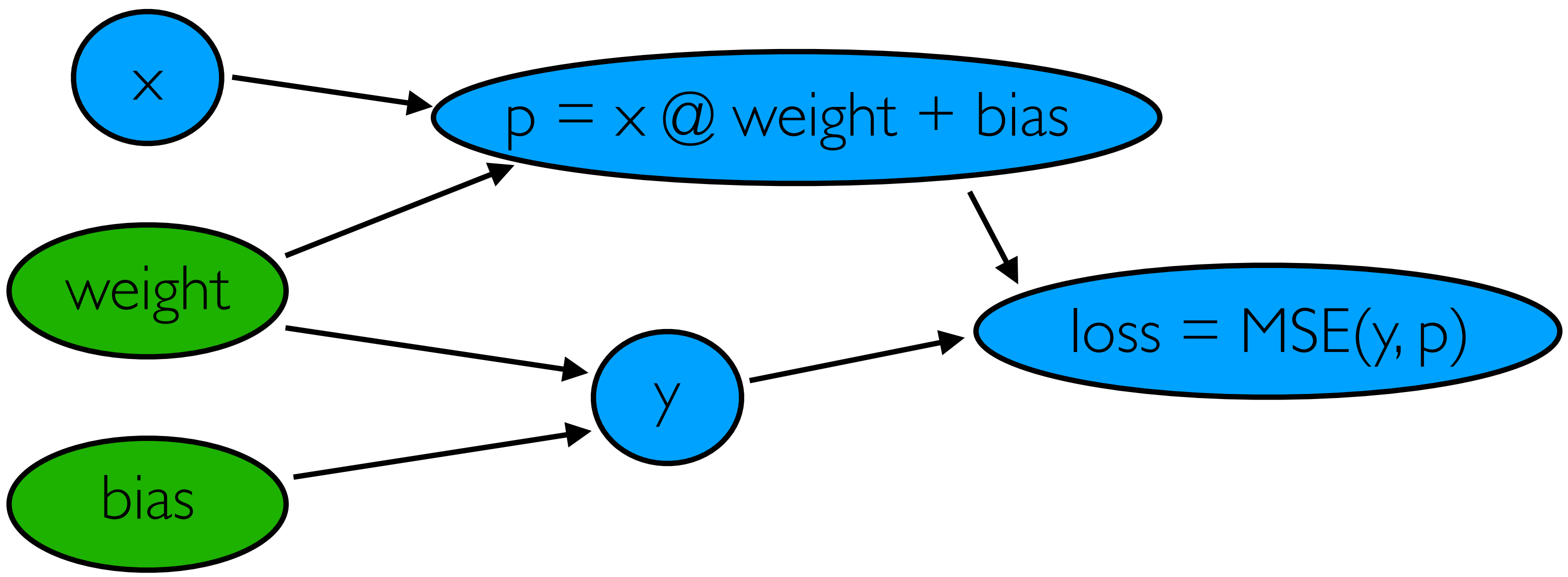

```
# x and y are known (these are matrices/vectors).
# what should weight and bias (parameters) be?

def model(data):
    return weight @ data + bias

p = model(x)
loss = MSE(y, p)
```

MSE (means squared error) measures how different predictions are from real values, so we want small loss (optimization).

If gradient of loss with respect to weight is positive, then decrease weight.



epoch 0

scan

	x1	x2	y
1	2	3	8
4	8	1	7
0	7	8	0
3	3	5	5
2	9	8	5
5	2	9	0

update params

epoch 1

scan

	x1	x2	y
1	2	3	8
4	8	1	7
0	7	8	0
3	3	5	5
2	9	8	5
5	2	9	0

update params

epoch 2

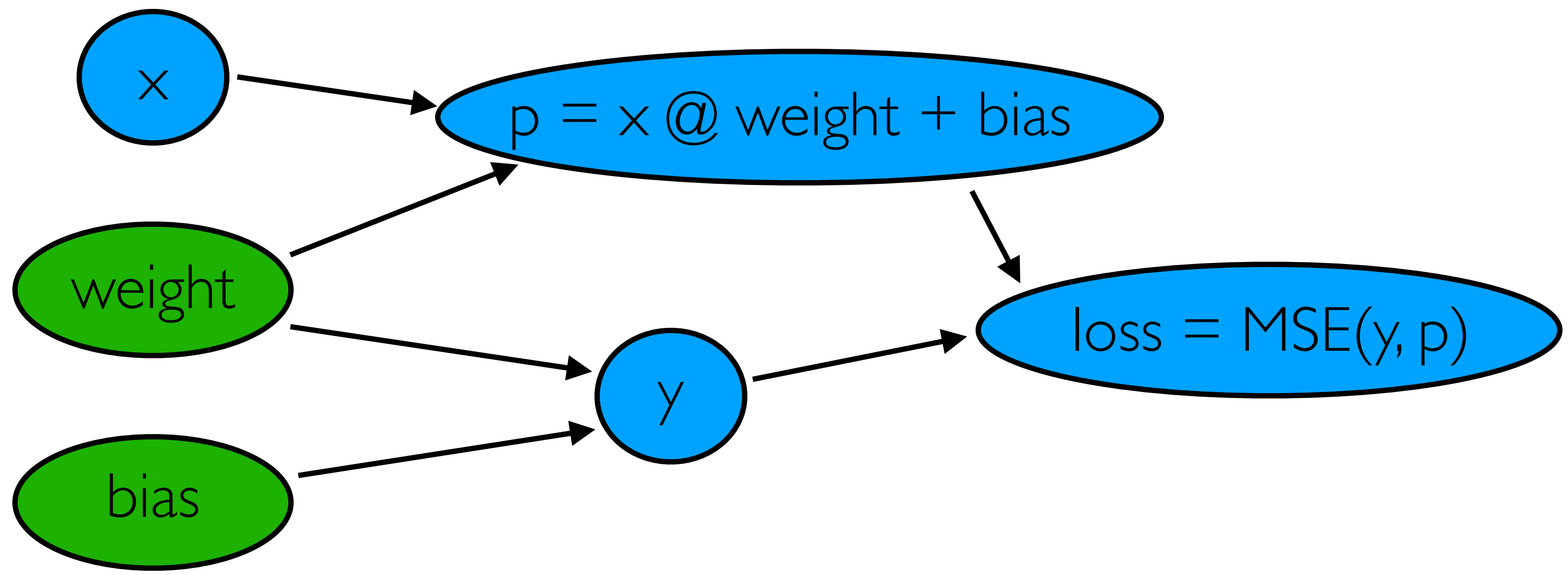
scan

	x1	x2	y
1	2	3	8
4	8	1	7
0	7	8	0
3	3	5	5
2	9	8	5
5	2	9	0

update params

...

gradient descent. slow (consider all data each update), and data might not fit in RAM



epoch 0

scan *shuffle*

	x1	x2	y
1	2	3	8
4	8	1	7
0	7	8	0
3	3	5	5
2	9	8	5
5	2	9	0

update
update
update

epoch 1

scan *shuffle*

	x1	x2	y
3	3	5	5
5	2	9	0
4	8	1	7
2	9	8	5
0	7	8	0
1	2	3	8

update
update
update

epoch 2

scan

	x1	x2	y
2	9	8	5
0	7	8	0
5	2	9	0
4	8	1	7
3	3	5	5
1	2	3	8

update
update
update

...

stochastic gradient descent. shuffle each time, process in small batches that fit in memory

Demos...