Ya gotta make it obvious

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

A bit of background…

- I'm retooling as a machine learning droid after an entire career in parsing and programming language implementation
  (you might know me as "the ANTLR guy")
- Since I'm no longer subject to the paper count treadmill, I'm exploring less traditional scholarly work, such as creating state-of-the-art visualizations and animations to explain machine learning concepts and models: https://explained.ai/
- My mission is to explain complex topics deeply, but in the simplest possible way, and to build useful libraries to help the research and education community
- *My niche is what industry can't justify and academy doesn't value*

UNIVERSITY OF SAN FRANCISCO

What's wrong w/traditional academic output?

- The goal of academic research papers and instruction is to transmit ideas, but…
- Peer review and our egos often work against this goal because they are the enemy of simplicity and clarity
- We fear that simplicity implies small research contribution and we also want to impress our audience with our brilliance
- Therefore, we "math it up" or simply don't bother spending time to make our techniques or results clear
- We should value simple and clear expositions most of all, for both large and small contributions, even if it's lots of extra work
- Try to illuminate not impress!

What's wrong with articles and lectures?

- It's frustrating trying to learn new concepts in machine learning
- There is some excellent content on the net, but most of it is the same superficial mostly-correct article rehashed 10^5 times
- There are lots of pain points that don't seem to bother people, or perhaps they think it's the best we can do; e.g., debugging and visualization aids can be primitive
- In contrast, academic papers are correct but usually too dense to include examples and don't allow dynamic content like animations and video
- What we need: correct, deep, and obvious (as possible)

Advice on achieving "correct, deep, and obvious"

1. Seek a deep understanding of concepts
   - Ask why or how something works; be able to reproduce the technique
   - W/o depth, we can't get to the essence, to zero in on the critical idea
   - W/o depth, we can't strip away any mysticism (I'm talkin to you neural nets!)
   - Can you describe the essence in one key phrase or sentence? 🔑
2. Target the human visual cortex in your artifacts
3. Rederive everything and baby step the audience to the solution; don't start with the final math solution, for example
4. You can't build what you can't imagine; don't limit yourself by what's possible or what you know how to build; engage your audience!
5. Sometimes you are right and everyone else is wrong

Some examples following this advice

Explaining gradient boosting (regression)

- Start by isolating the essence of boosting; boil it down!
- Next, design a visualization that clearly shows the key bits
- Boosting is just the addition of an initial "drive" f_0 plus a series of imprecise "putts" Δ_i

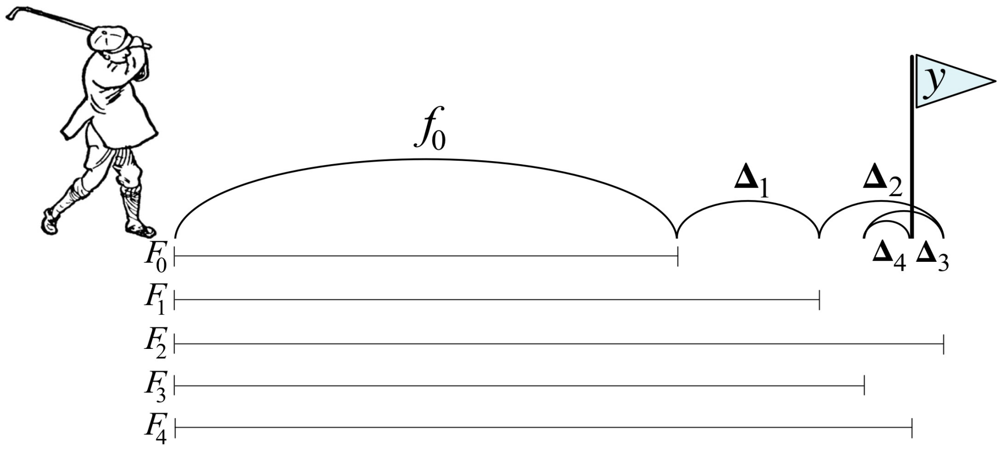


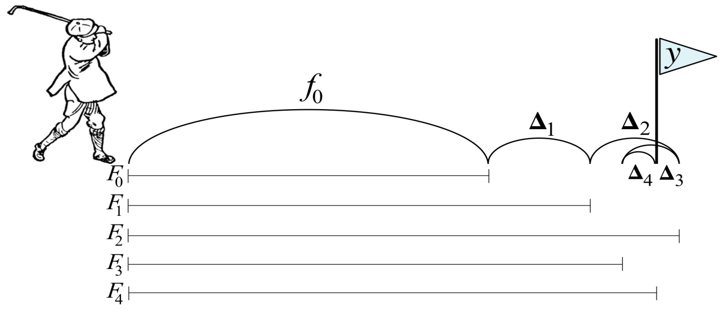
Diagram from https://explained.ai/gradient-boosting/index.html
but golfer clipart from http://etc.usf.edu/clipart/

UNIVERSITY OF SAN FRANCISCO

Explaining gradient boosting (regression)

- **Then**, with this intuition, show the concise equation that embodies the process:

$$\begin{aligned}\hat{y} &= f_0(\mathbf{x}) + \Delta_1(\mathbf{x}) + \Delta_2(\mathbf{x}) + \ldots + \Delta_M(\mathbf{x}) \\ &= f_0(\mathbf{x}) + \sum_{m=1}^{M} \Delta_m(\mathbf{x}) \\ &= F_M(\mathbf{x})\end{aligned}$$

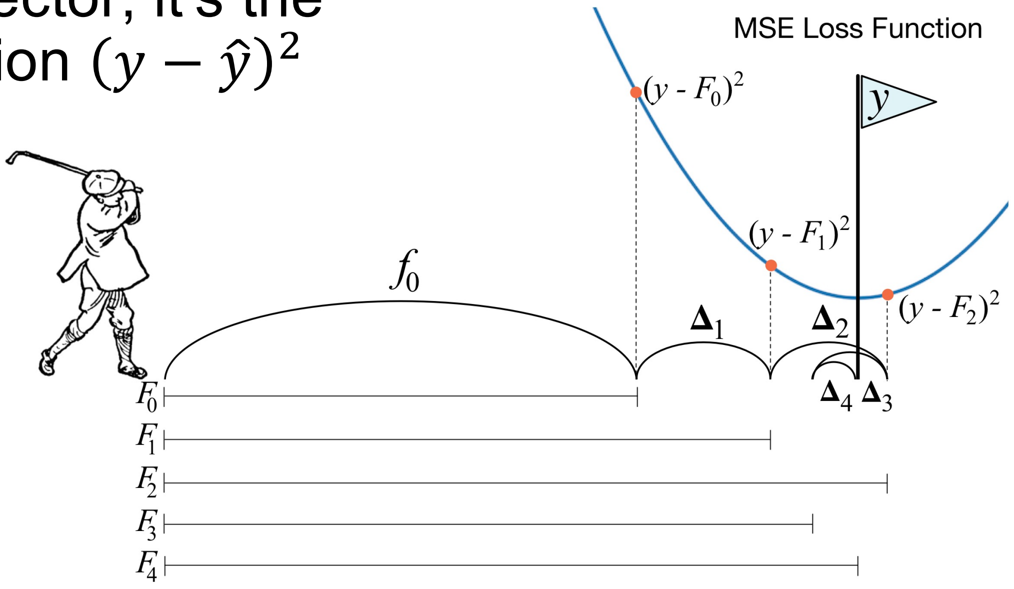


Don't start with the math!
The math is the condensation of an inventor's intuition
The inventor didn't start with the answer; why should our audience have to?

UNIVERSITY OF SAN FRANCISCO

While we're at it…
GBMs: Gradent descent in prediction space

- Clarifying a common fuzzy detail
- Difference $y - \hat{y}$ is not just a vector; it's the gradient of the MSE loss function $(y - \hat{y})^2$
- Chasing the difference vector performs MSE gradient descent in prediction space



See https://explained.ai/gradient-boosting/descent.html#sec:3.2 for more details

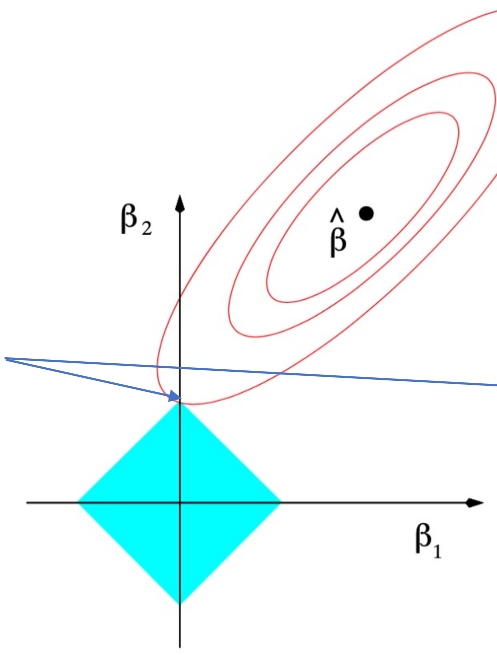
UNIVERSITY OF SAN FRANCISCO

Just because it's visual doesn't mean it's effective

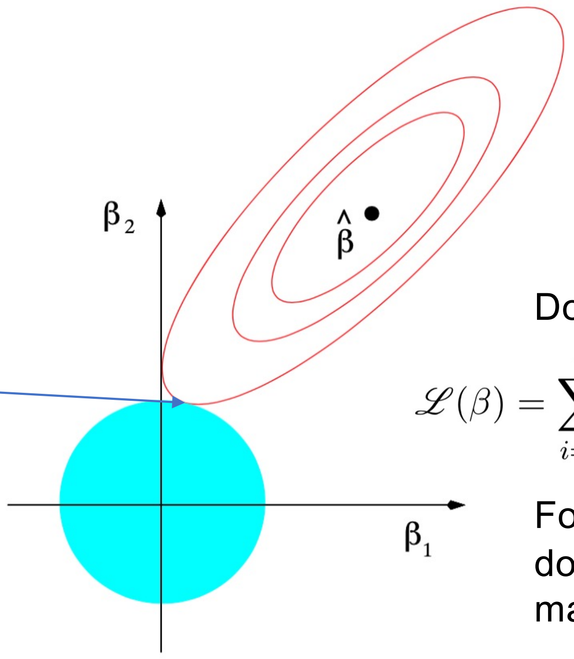
Viz meant to help explain L1 vs L2 regularization and why L1 encourages zero coefficients but L2 does not

(ESL is a great book written by brilliant statisticians, but these visualizations never helped me)

Why here?



L1 regularization (lasso)



L2 regularization (ridge)

Don't show this math first:

$$\mathscr{L}(\beta) = \sum_{i=1}^{n}(y^{(i)} - (\mathbf{x}'^{(i)} \cdot \beta))^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

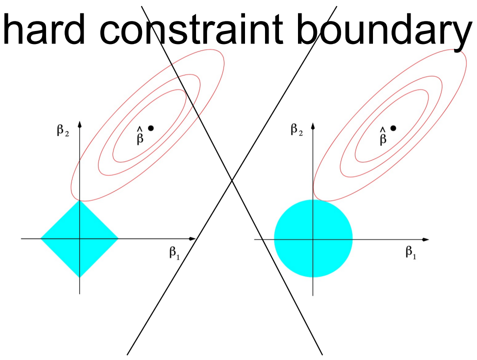
For one thing, the pictures don't correspond to that math!!!

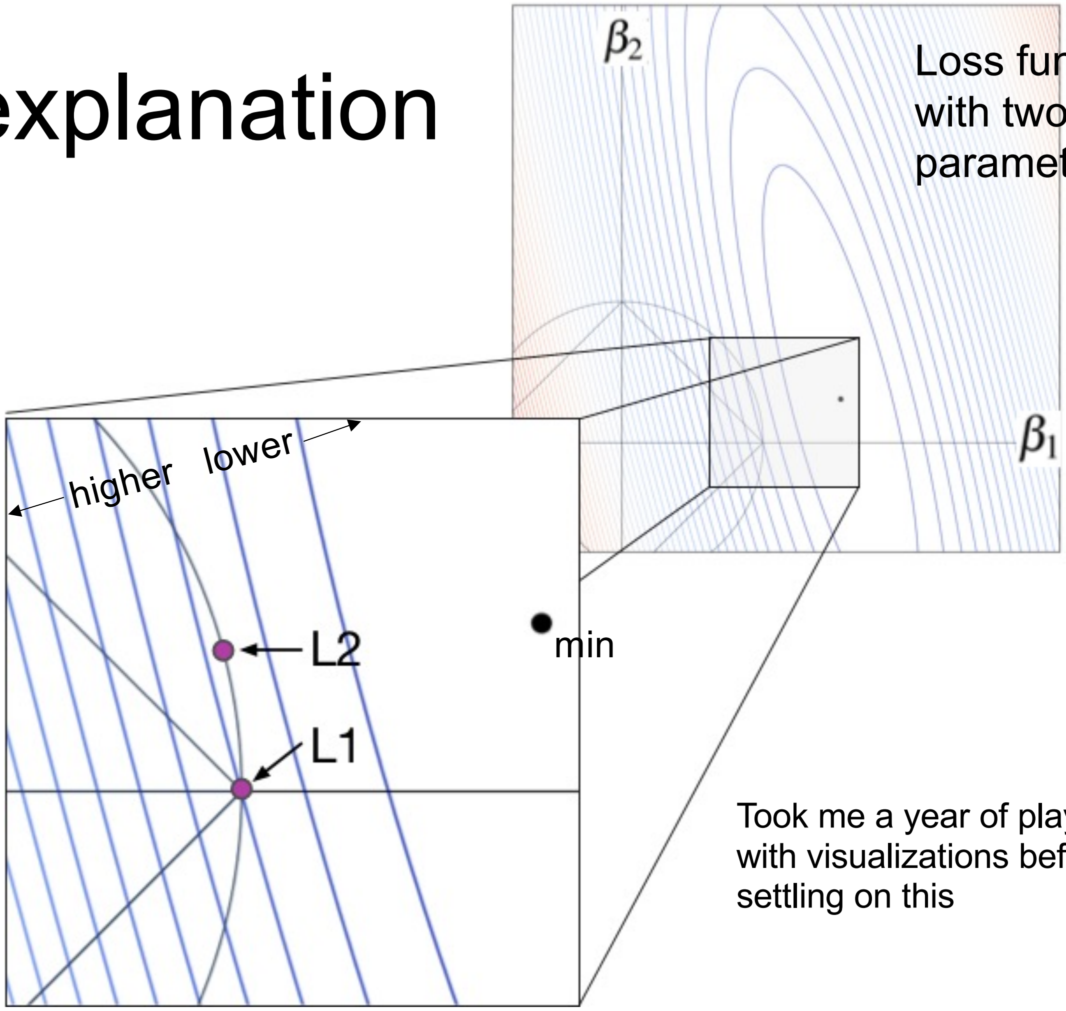
Fig 3.11, Page 71 Elements of Statistical Learning

UNIVERSITY OF SAN FRANCISCO

Better L1 vs L2 explanation

- Simple visual proof: Any movement of (β_1, β_2) away from purple L1 dot at $\beta_2 = 0$ increases the loss
- Left of any contour line is higher loss
- To shift away from 0, we'd need to rotate loss function contour lines to be more parallel with L1 hard constraint boundary





Loss function with two parameters

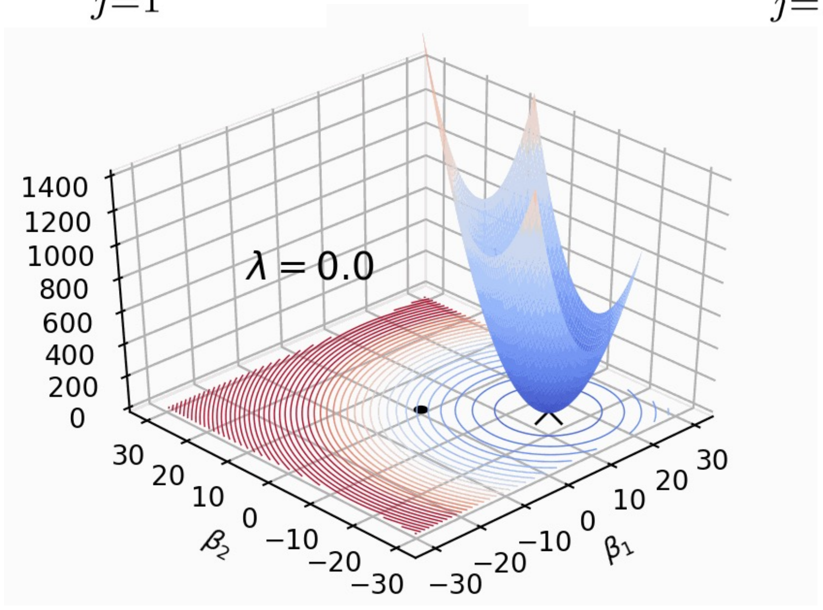
Took me a year of playing with visualizations before settling on this

From https://explained.ai/regularization/index.html

Sometimes animations and 3D help

- We use **hard constraint** regions conceptually but **soft constraints** to implement
- Increasing λ increases penalty at any (β_1, β_2) and shifts loss point towards the origin
- 3D animations are painful to build but illuminating

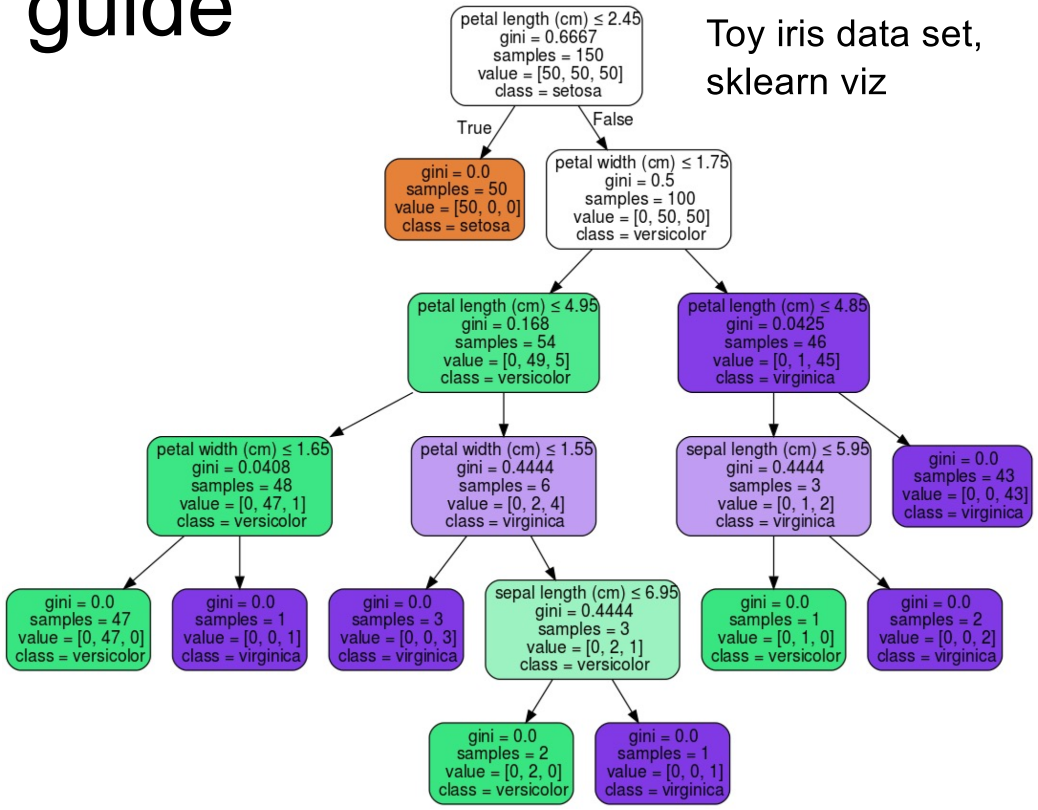
$\mathscr{L}(\beta)$ s.t. $\sum_{j=1}^{p} \beta_j^2 \le t$ becomes $\mathscr{L}(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$



UNIVERSITY OF SAN FRANCISCO

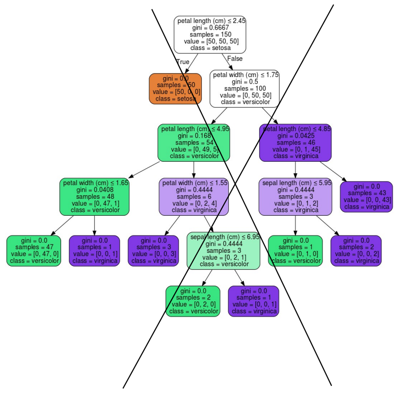
Let your imagination and deep understanding be your guide

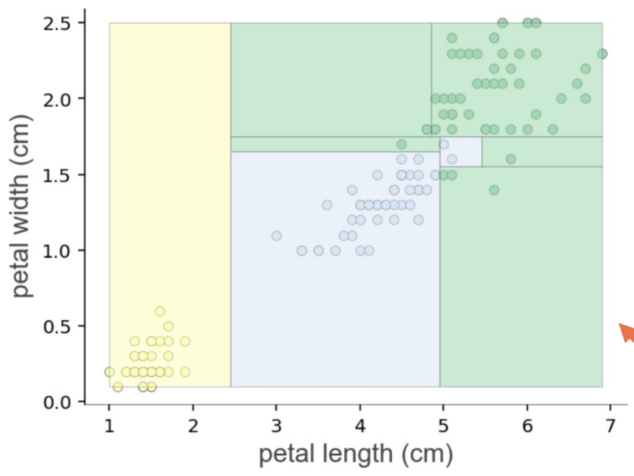
- Decision trees tesselate feature space into regions of feature space according to purity of target y; each node partitions a single var at a specific value
- Consider sklearn's viz, which shows tree structure, but that's not what we really care about
- I have to think too hard here; decisionmaking is not obvious
- Why were those split vars/values chosen? What do colors mean?



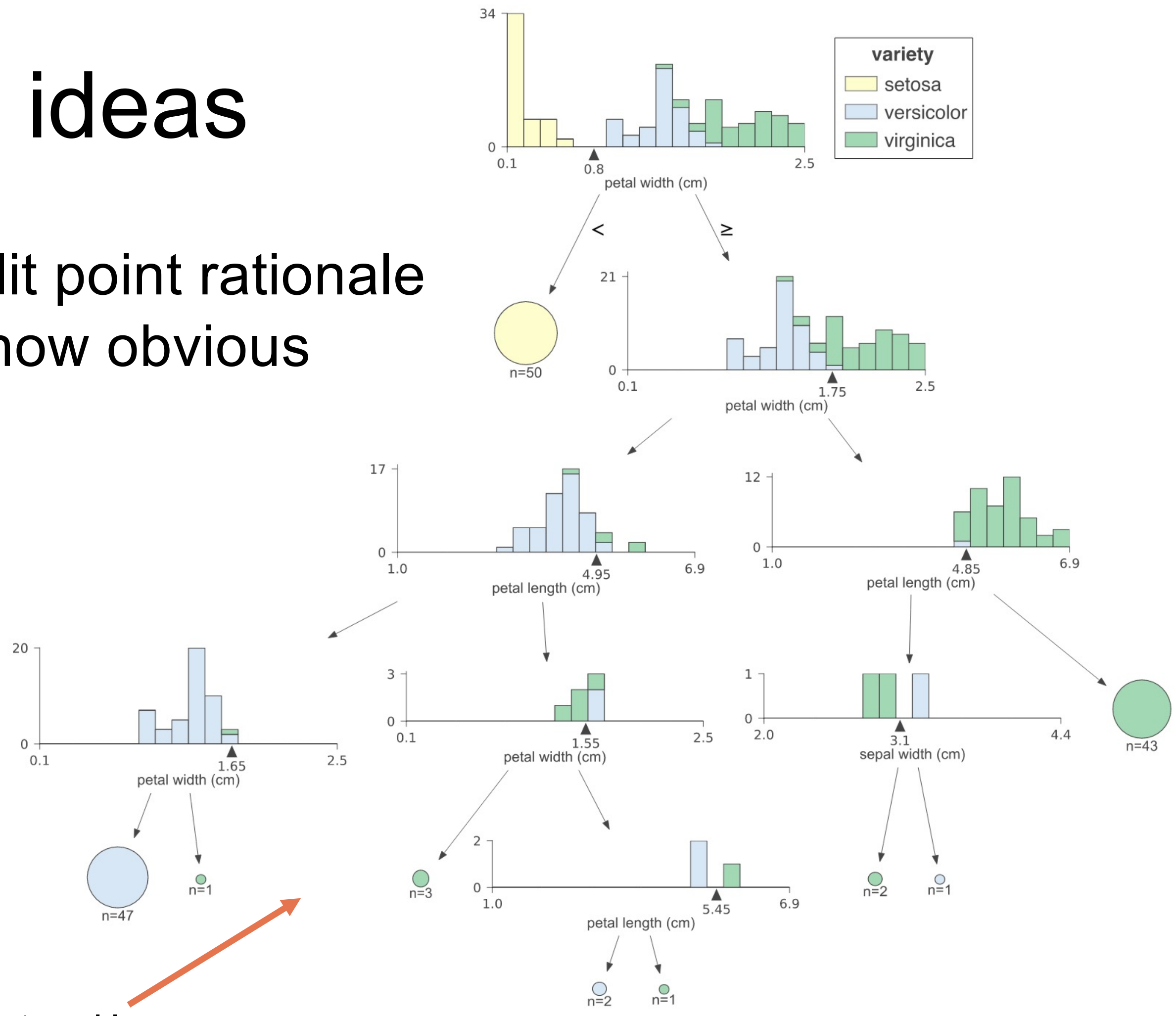
Toy iris data set, sklearn viz

Illustrate the key ideas





Split point rationale is now obvious



Effect How

UNIVERSITY OF SAN FRANCISCO

See https://explained.ai/decision-tree-viz/index.html

Debugging matrix algebra is not obvious

We often get less than helpful exception messages:

```
h_ = torch.tanh(Whh_ @ (r*h) + Uxh_ @ X.T + bh_)
```

C++?

```
RuntimeError: size mismatch, m1: [764 x 256], m2: [764 x 200] at
/tmp/pip-req-build-as628lz5/aten/src/TH/generic/THTensorMath.cpp:41
```

It's not clear why/where expression failed; I still have to think hard

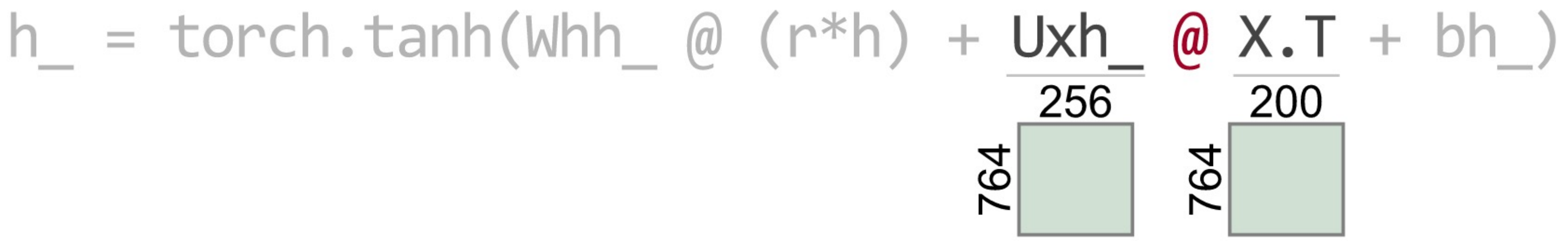
We need operator and operands; TensorSensor generates:

```
Cause: @ on tensor operand Uxh_ w/shape [764, 256]
and operand X.T w/shape [764, 200]
```

See https://explained.ai/tensor-sensor/index.html

UNIVERSITY OF SAN FRANCISCO

Better yet: make it obvious with a viz



```
Cause: @ on tensor operand Uxh_ w/shape [764, 256]
and operand X.T w/shape [764, 200]
```

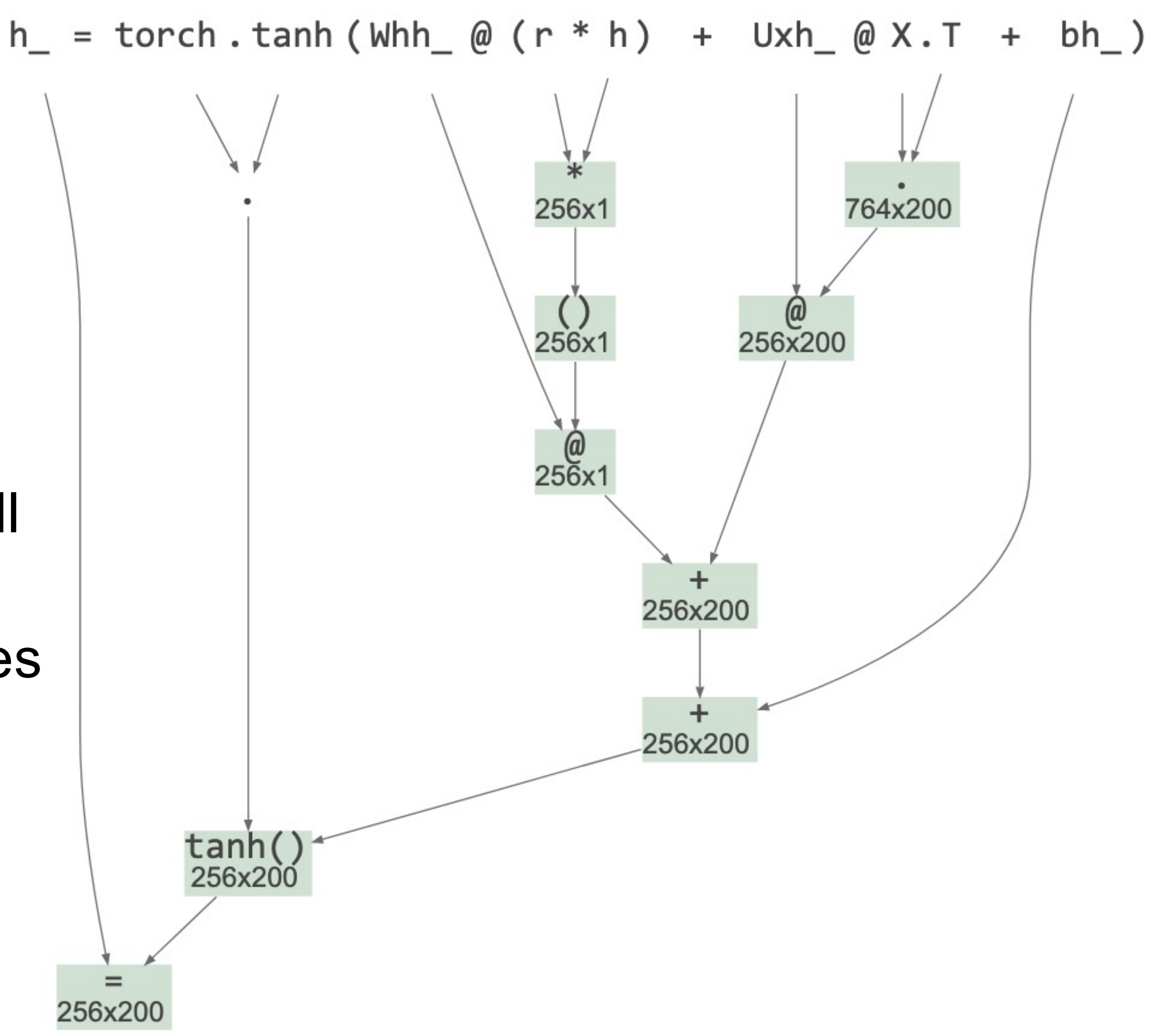
(Notice the details like highlighting, red operator color, and de-highlighting)

See https://explained.ai/tensor-sensor/index.html

UNIVERSITY OF SAN FRANCISCO

While we're at it… Aid for reading matrix code

- What do we care about when reading complex matrix expressions?
- We need the shape of all subexpressions
- Abstract syntax tree gives us an obvious viz with matrix dimensions



RNNs / neural nets are cloaked in mysticism

- I don't think the neural net analogy is helpful anymore
- The tangle of connections between neurons obscures rather than clarifies
- What exactly does that diagram portray?

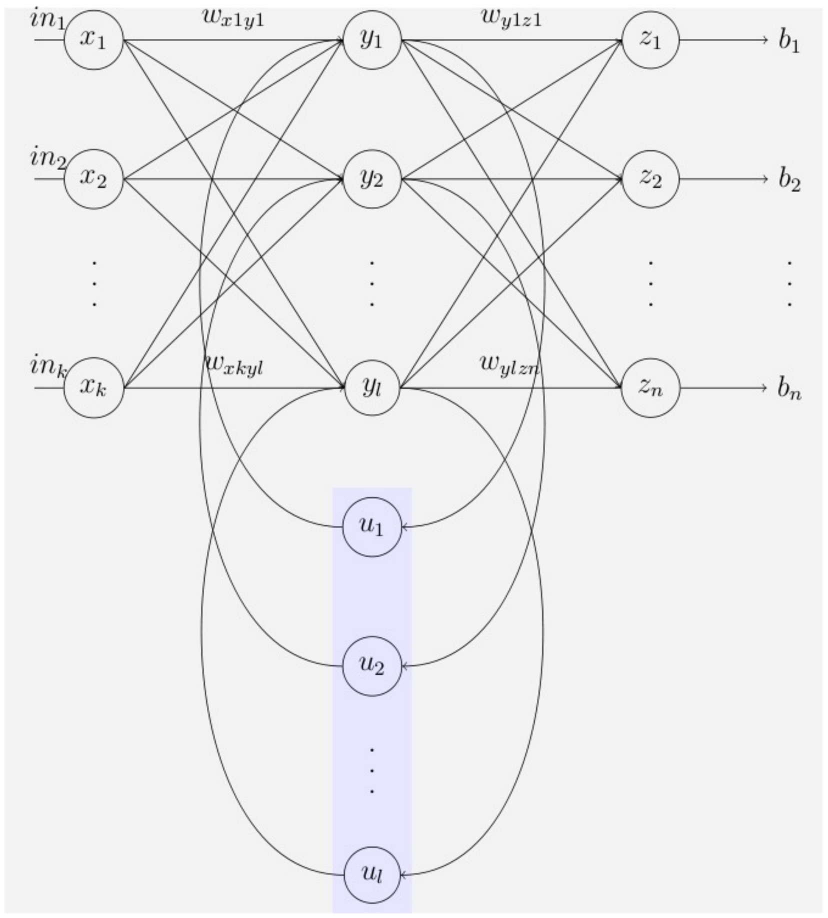


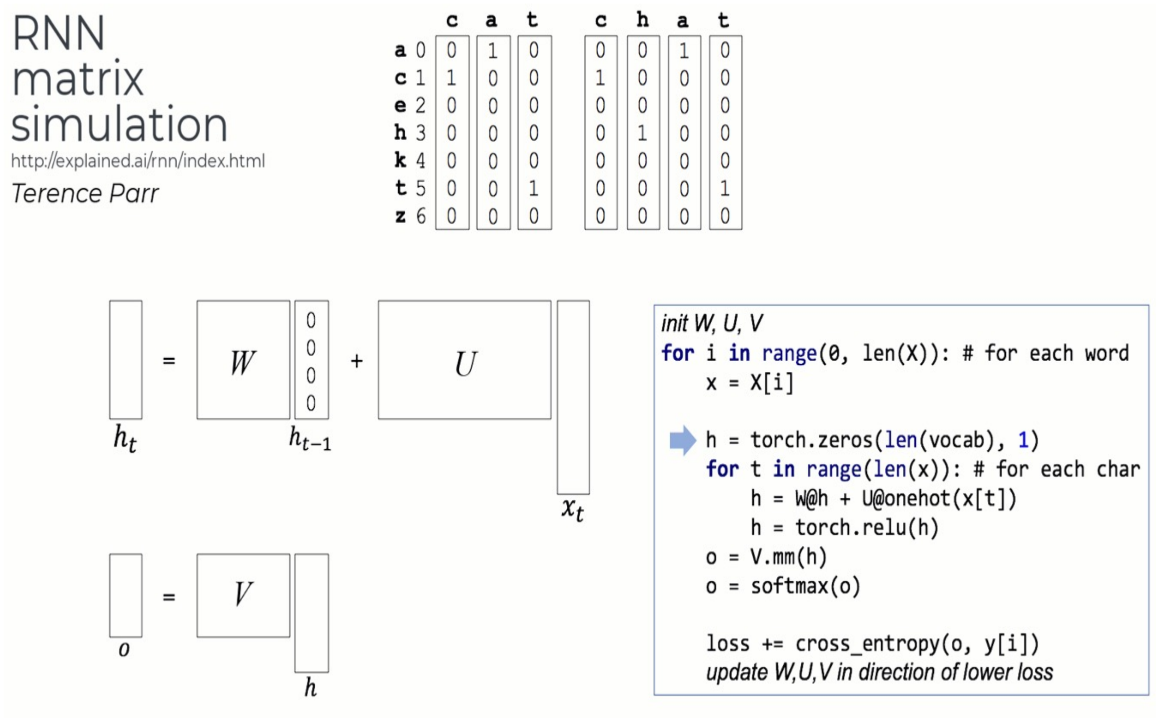
Image credit Fyedernoggersnodden https://en.wikipedia.org/wiki/Recurrent_neural_network

UNIVERSITY OF SAN FRANCISCO

Strip away the mysticism

- What exactly is an RNN doing?
- It's just encoding symbols as vectors then aggregating them in a position-dependent way like a hash function for vectors; we use matrix transforms W, U and solve a classification problem to learn the parameters
- Animate the matrix algebra to make the implementation as concrete as possible
- (This animation was painful to build but is very helpful)

Classify cat words to language


RNN matrix simulation
http://explained.ai/rnn/index.html
Terence Parr
h_t = W h_{t-1} + U x_t
o = V h

		c	a	t	c	h	a	t
a	0	0	1	0	0	0	1	0
c	1	1	0	0	1	0	0	0
e	2	0	0	0	0	0	0	0
h	3	0	0	0	0	1	0	0
k	4	0	0	0	0	0	0	0
t	5	0	0	1	0	0	0	1
z	6	0	0	0	0	0	0	0

```
init W, U, V
for i in range(0, len(X)): # for each word
    x = X[i]

    h = torch.zeros(len(vocab), 1)
    for t in range(len(x)): # for each char
        h = W@h + U@onehot(x[t])
        h = torch.relu(h)
    o = V.mm(h)
    o = softmax(o)

    loss += cross_entropy(o, y[i])
    update W,U,V in direction of lower loss
```

(Notice weights W, U, V flash when updated)

See https://explained.ai/rnn/index.html

Be skeptical and confident: Sometimes everybody else is wrong and you are right

- Gini/var drop importance for random forests can be wildly wrong

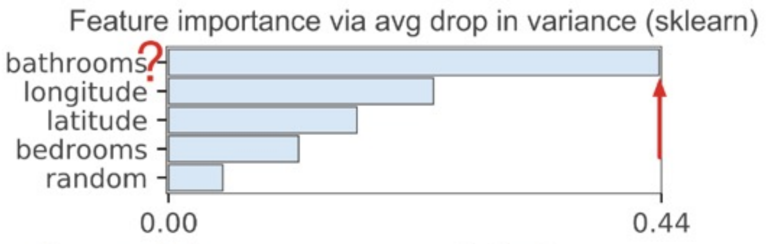


Figure 1(a). `scikit-learn` default importances for Random Forest **regressor** predicting apartment rental price from 4 features + a column of random numbers. Random column is last, as we would expect but the importance of the number of bathrooms for predicting price is highly suspicious.

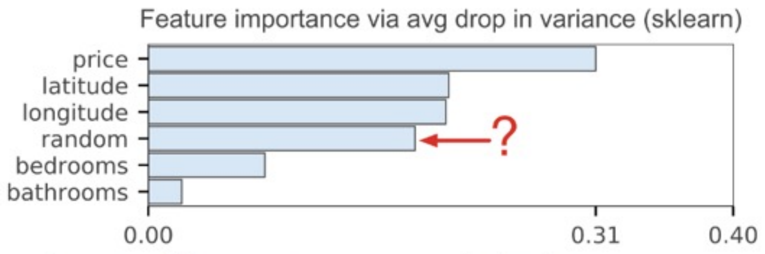
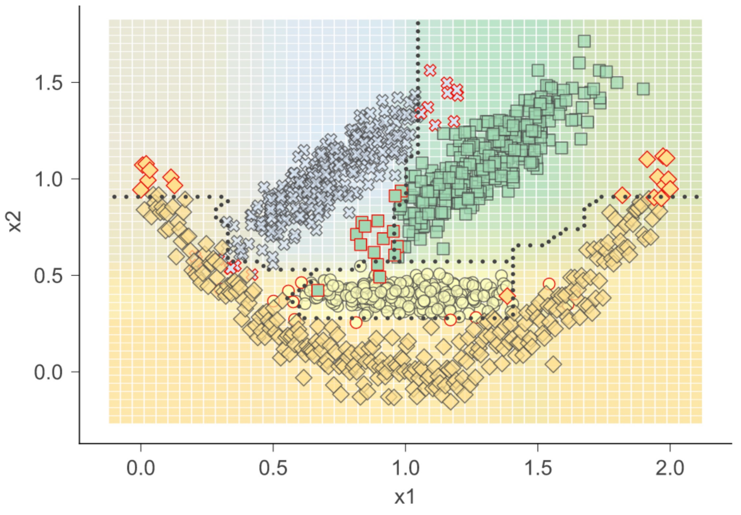


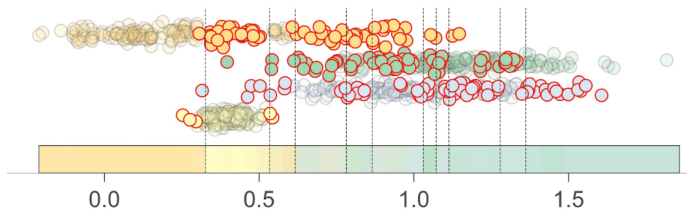
Figure 1(b). `scikit-learn` default importances for Random Forest **classifier** predicting apartment interest level (low, medium, high) using 5 features + a column of random numbers. Highly suspicious that random column is much more important than the number of bedrooms.

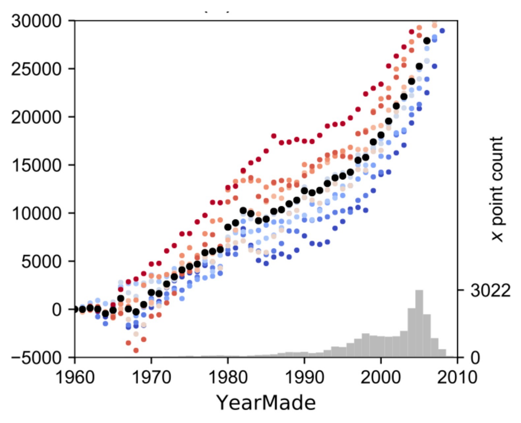
See https://explained.ai/rf-importance/index.html

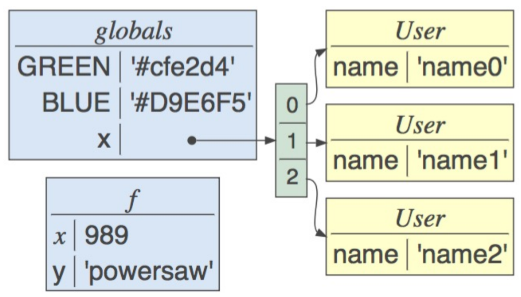
UNIVERSITY OF SAN FRANCISCO

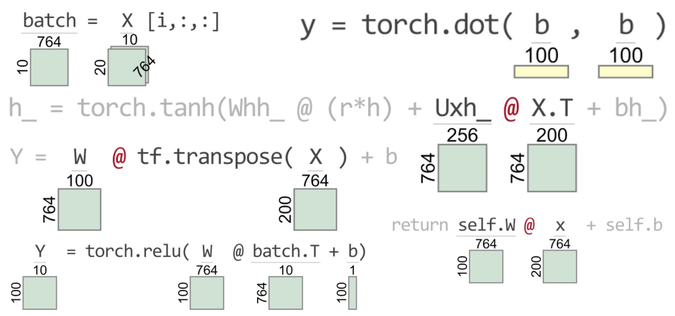
Engage the audience's visual cortex!











batch = X [i,:,:]
y = torch.dot(b , b)
h_ = torch.tanh(Whh_ @ (r*h) + Uxh_ @ X.T + bh_)
Y = W @ tf.transpose(X) + b
return self.W @ x + self.b
Y = torch.relu(W @ batch.T + b)

UNIVERSITY OF SAN FRANCISCO

Some final advice

6. When you feel coding or learning pain, work hard to erase it; crave beauty, excellence, and simplicity
7. Be tenacious; never let the computer win; bash your way to victory; deep understanding and clear visualizations often require pathological determination and a trail of dead code