Regression and practical advice

Toby Dylan Hocking

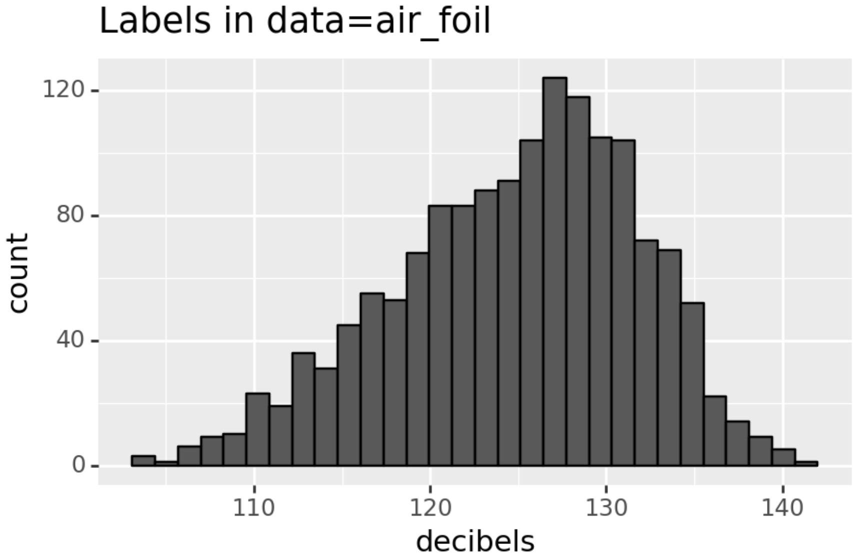
Supervised machine learning

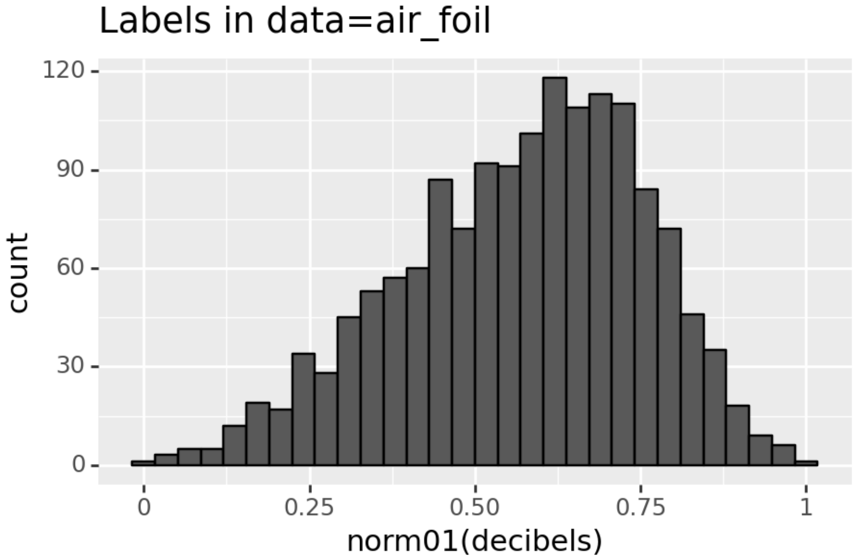
- Goal is to learn a function $f(\mathbf{x}) = y$ where $\mathbf{x} \in \mathbb{R}^p$ is an input/feature vector and y is an output/label.
- This week we will study linear models and neural networks for regression, meaning labels represented by $y \in \mathbb{R}$ is a real number.
- air foil self-noise data: $\mathbf{x} =$ Frequency (Hertz), Angle of attack (degrees), Chord length (meters), Free-stream velocity (meters per second), $y \in \mathbb{R}$ Scaled sound pressure level, in decibels.
- forest fires data: $\mathbf{x} =$meteorological and other data, $y \in \mathbb{R}_+$ burned area.
- some practical advice for getting gradient descent learning to work better (scaling, log transform, feature transform)

air foil self-noise data

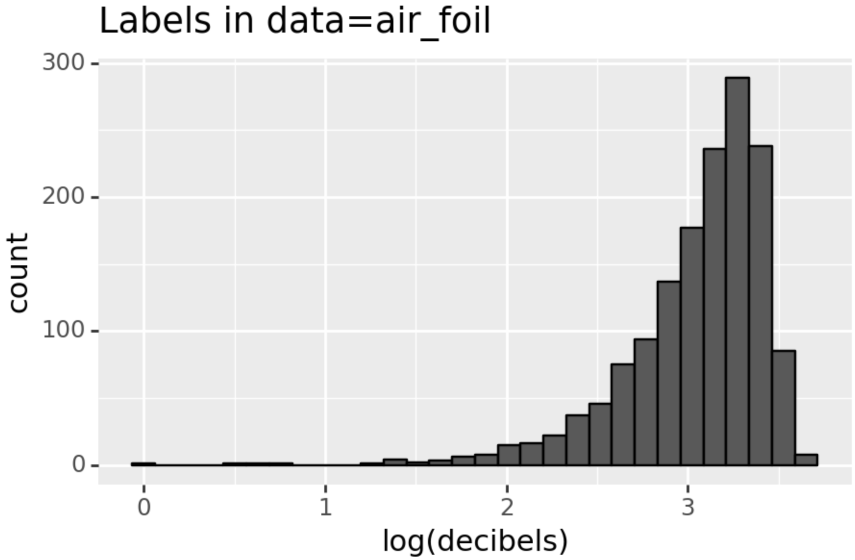
```
##        Hertz  degrees  ...    meters  decibels
## 0        800      0.0  ...  0.002663   126.201
## 1       1000      0.0  ...  0.002663   125.201
## 2       1250      0.0  ...  0.002663   125.951
## 3       1600      0.0  ...  0.002663   127.591
## 4       2000      0.0  ...  0.002663   127.461
## ...      ...      ...  ...       ...       ...
## 1498    2500     15.6  ...  0.052849   110.264
## 1499    3150     15.6  ...  0.052849   109.254
## 1500    4000     15.6  ...  0.052849   106.604
## 1501    5000     15.6  ...  0.052849   106.224
## 1502    6300     15.6  ...  0.052849   104.204
##
## [1503 rows x 6 columns]
```

Need to scale label vector, to avoid numerical instability in gradient descent.


Labels in data=air_foil
count
120
80
40
0
110
120
130
140
decibels



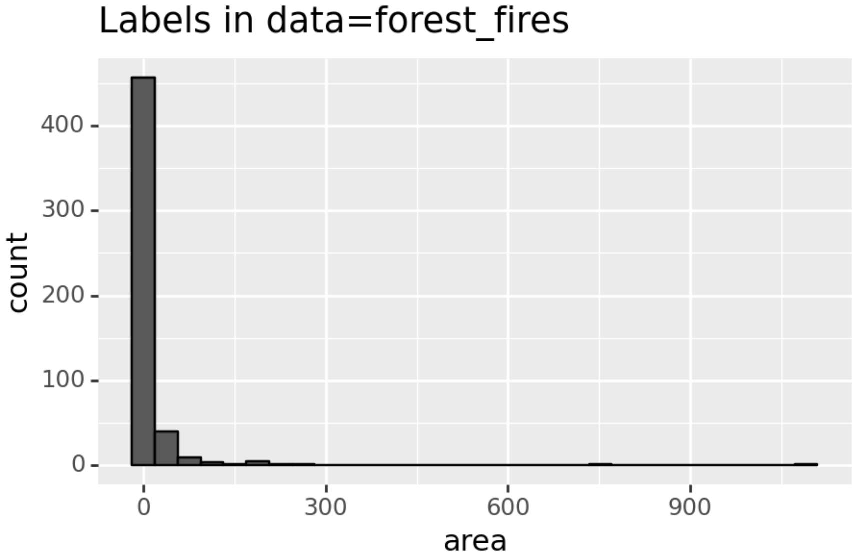
Labels in data=air_foil
120
90
60
30
0
count
0
0.25
0.50
0.75
1
norm01(decibels)

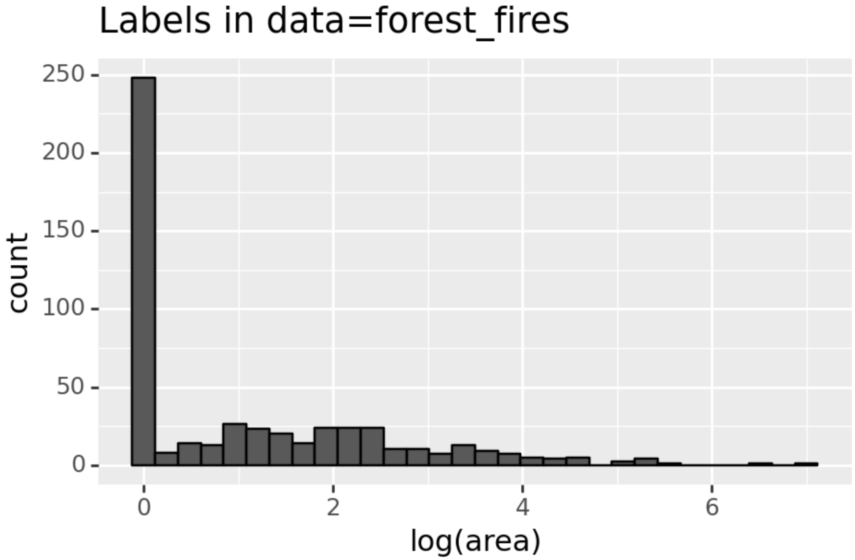

Labels in data=air_foil
count
log(decibels)
0
100
200
300
0
1
2
3

forest fires data

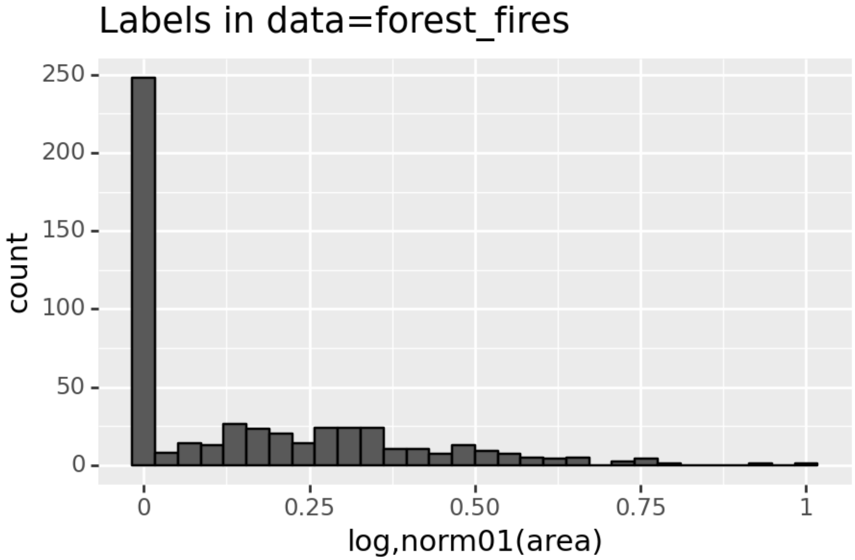
```
##       X  Y month  ... wind  rain   area
## 0     7  5   mar  ...  6.7   0.0   0.00
## 1     7  4   oct  ...  0.9   0.0   0.00
## 2     7  4   oct  ...  1.3   0.0   0.00
## 3     8  6   mar  ...  4.0   0.2   0.00
## 4     8  6   mar  ...  1.8   0.0   0.00
## ..   .. ..   ...  ...  ...   ...    ...
## 512  4  3   aug  ...  2.7   0.0   6.44
## 513  2  4   aug  ...  5.8   0.0  54.29
## 514  7  4   aug  ...  6.7   0.0  11.16
## 515  1  4   aug  ...  4.0   0.0   0.00
## 516  6  3   nov  ...  4.5   0.0   0.00
##
## [517 rows x 13 columns]
```

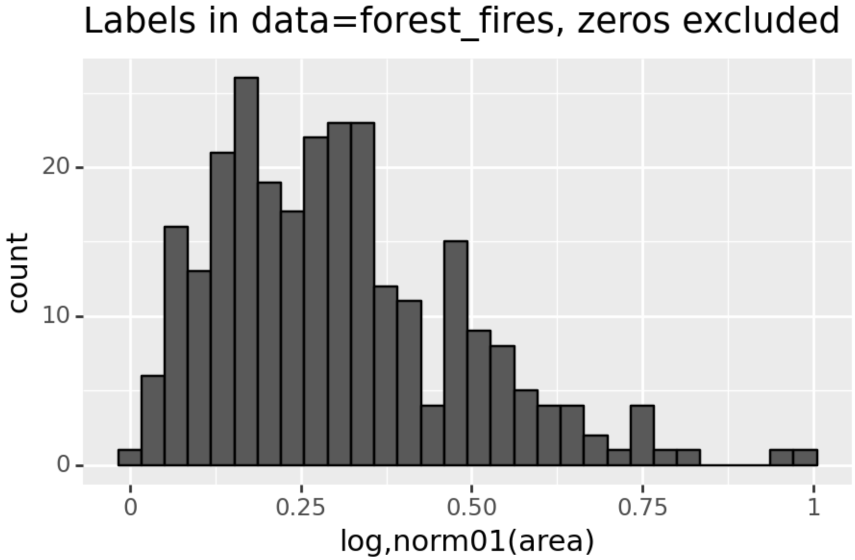
For categorical variables like month, need to ignore, or re-encode (ordinal or one-hot encoding).


Labels in data=forest_fires
count
400
300
200
100
0
0
300
600
900
area



Labels in data=forest_fires
count
250
200
150
100
50
0
0
2
4
6
log(area)


Labels in data=forest_fires
count
250
200
150
100
50
0
0
0.25
0.50
0.75
1
log,norm01(area)


Labels in data=forest_fires, zeros excluded
count
20
10
0
0
0.25
0.50
0.75
1
log,norm01(area)

Enforcing non-negative predictions

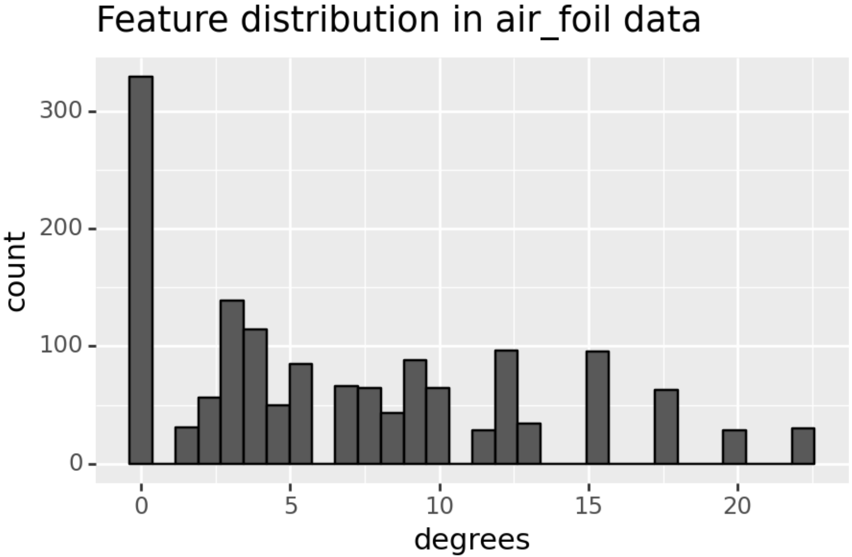
Assume a label $y > 0$. How to make sure that we get a positive prediction from our neural network?

Neural network predicts $f(x)$, a real number (maybe negative).

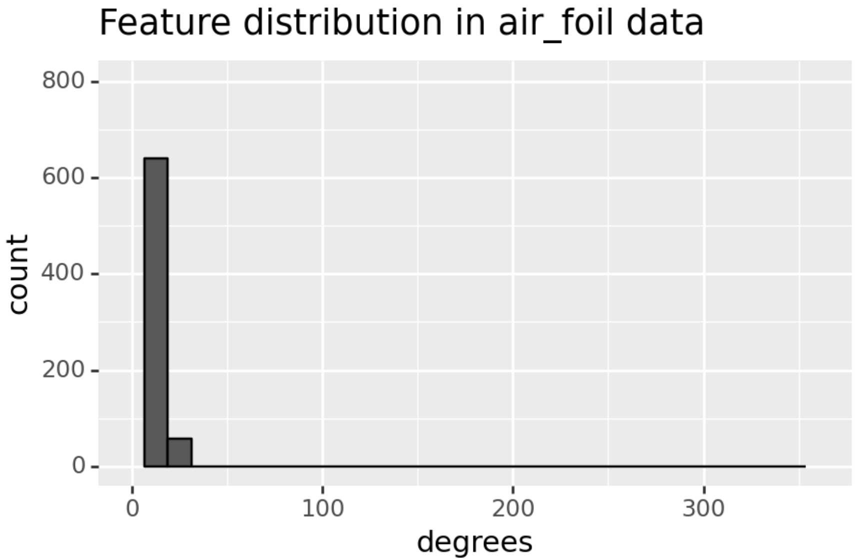
Log-normal loss: for a given label y, loss is $(\log[\exp f(x)] - \log[y])^2$, which is defined for any positive predictions $\exp f(x) > 0$.

How to handle $y = 0$? Binary classification then regression.

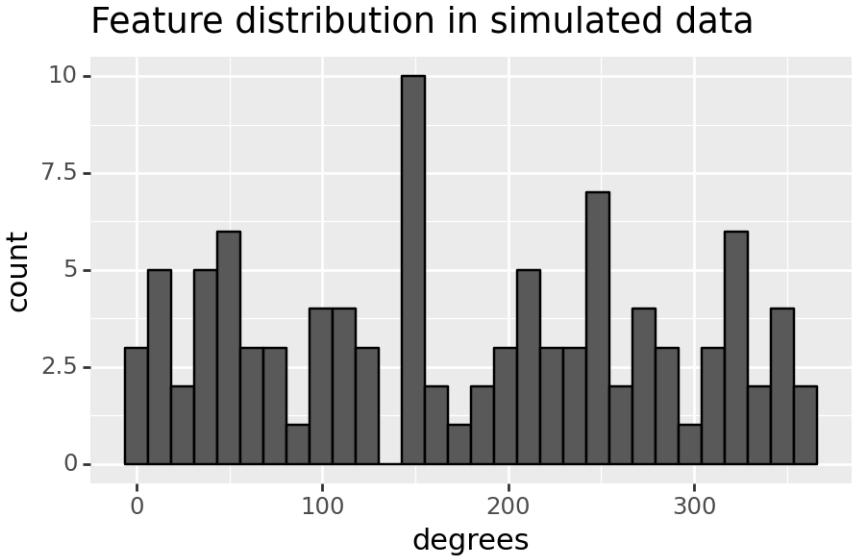
Real data feature distribution



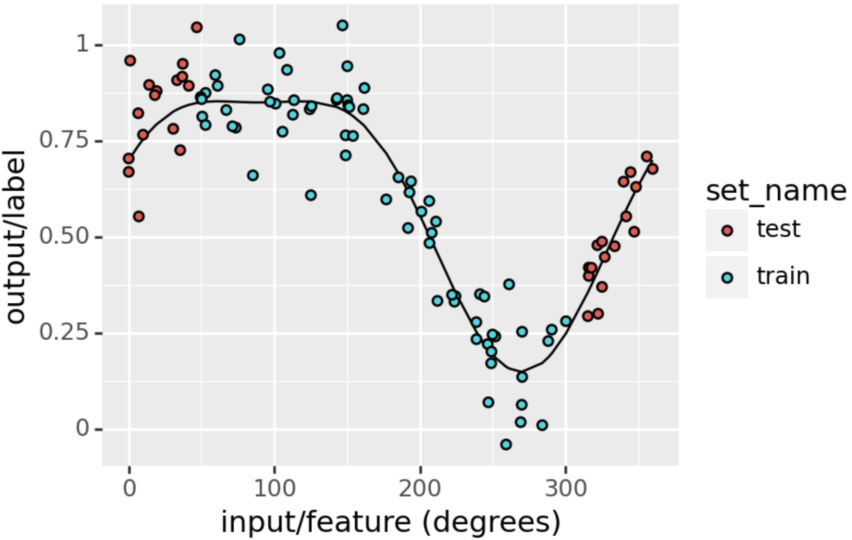
Real data feature distribution



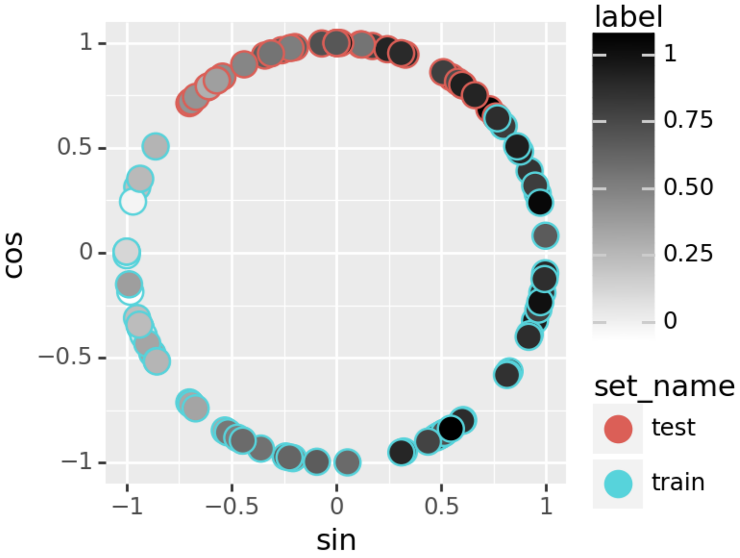
Simulated data feature distribution



Pattern in simulated data has continuity over 0/360 edge



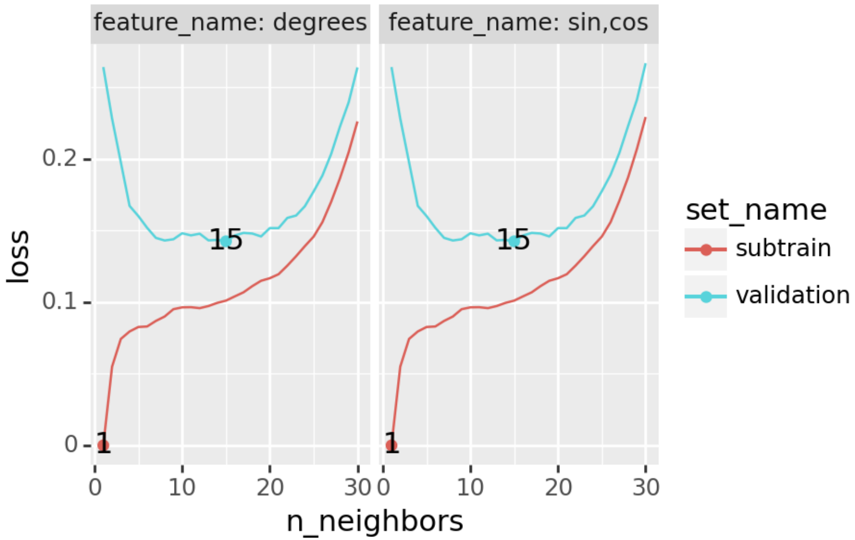
Non-linear basis expansion



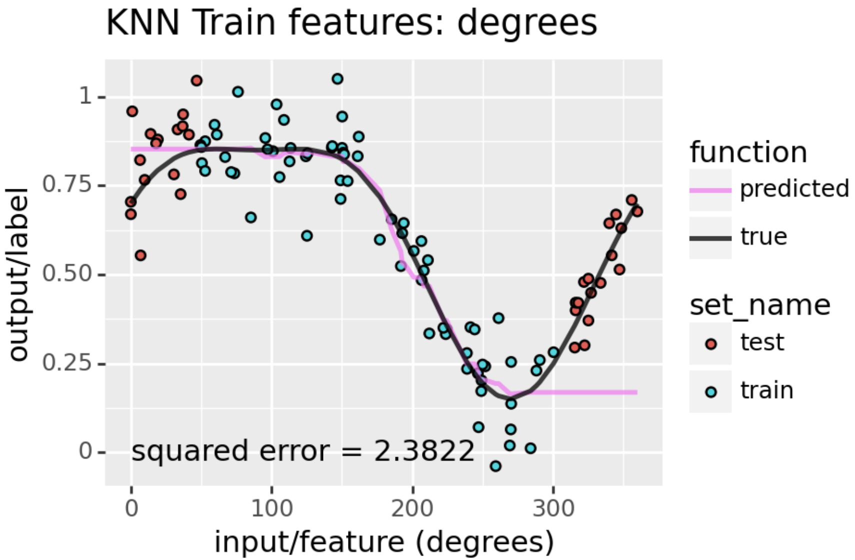
Nearest neighbors, baseline/code

- In binary classification, we predict the most frequent class among the K nearest neighbors (K=N is the featureless baseline).
- In regression we predict mean label of K nearest neighbors (instead of most frequent label / mode).
- That is the only difference between KNeighborsRegressor and KNeighborsClassifier in sklearn.neighbors.

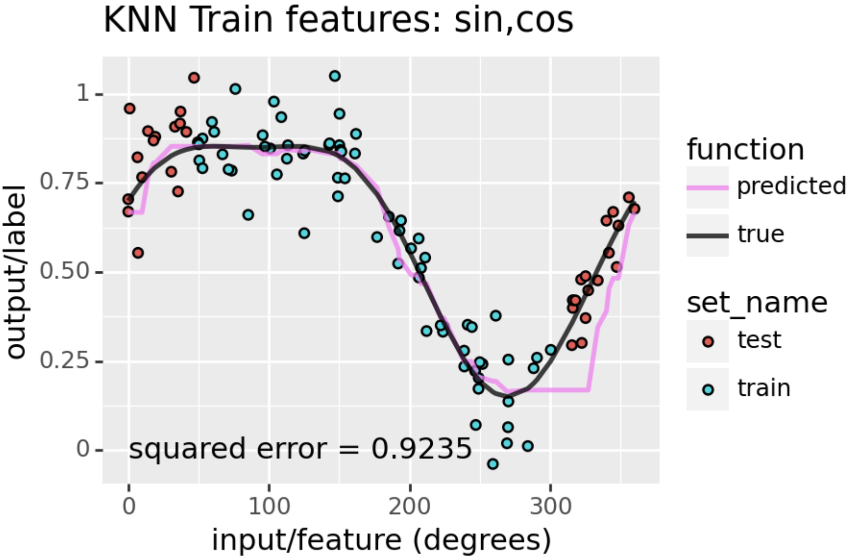
Train nearest neighbor regression



Learned function not continuous over 0/360

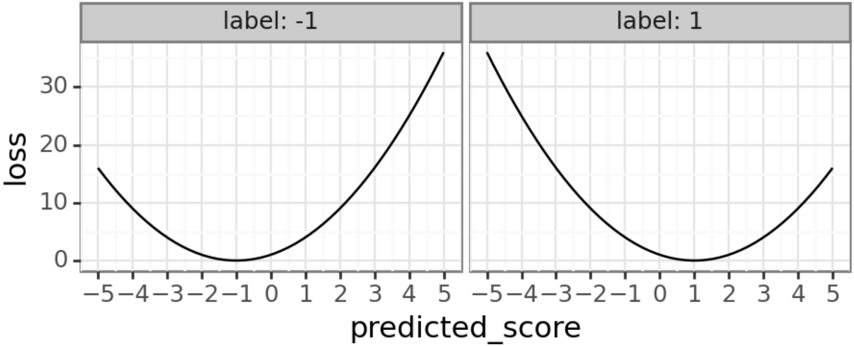


sin/cos features enforce continuity

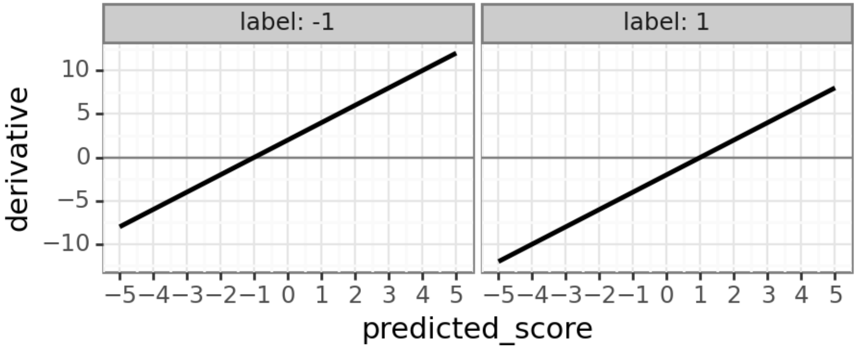


How are the neural network weights learned?

- Typically we use some version of gradient descent.
- This algorithm requires definition of a differentiable loss function to minimize on the train set.
- For regression problems ($y \in \mathbb{R}$) we use the square loss, $\ell[f(\mathbf{x}), y) = [f(\mathbf{x}) - y]^2$.

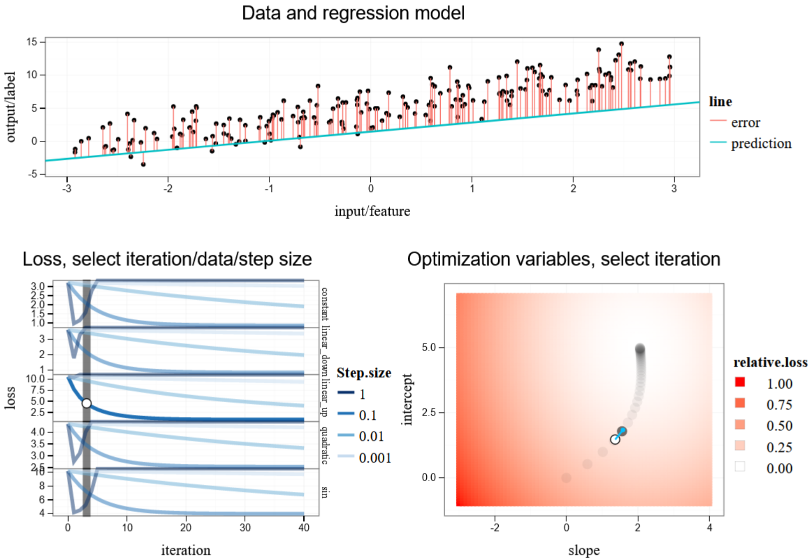


Visualization of square loss gradient/derivative



Interactive visualization of gradient descent for regression

http://ml.nau.edu/viz/2022-02-02-gradient-descent-regression/



TODO slides

- Comparison with ReLU act in last layer to force non-negative predictions (including zeros).

Possible exam questions

- Say $x = [5, -3, 10]$, $w = [2, 3, 1]$, $y = 6$, and we are doing regression (square loss). Compute loss L, gradient $\nabla_w L$, and new weights after one step of gradient descent with learning rate / step size $= 0.1$.