Dall-E

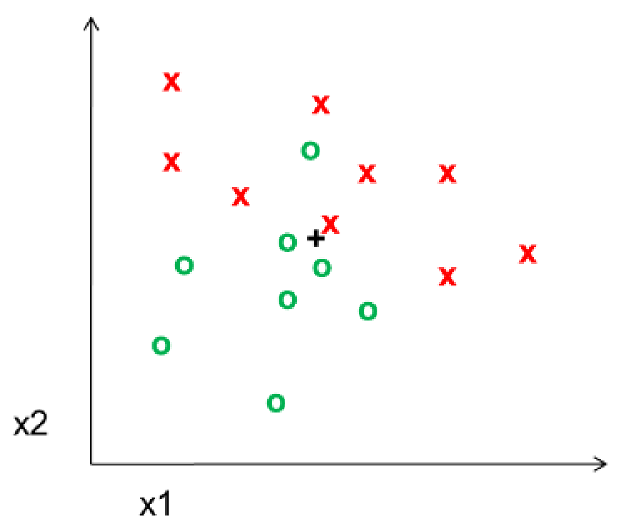
Linear Models: Linear and Logistic Regression

Applied Machine Learning
Derek Hoiem

Brief review

1. Which of these tend to be decreased as the number of training examples increases?
   a) Training error
   b) Test error
   c) Generalization error

8. Classify (‘o’ or ‘x’) the ‘+’ with 1-NN and 3-NN



1-NN:
3-NN:

Brief Review

1. What assumption does the Naive Bayes model make if there are two features x1 and x2?
   (a) $P(y|x1, x2) = P(y|x1)P(y|x2)$
   (b) $P(x1, x2|y) = P(x1|y)P(x2|y)$
   (c) Neither of these are true
   (d) These are equivalent and both true

2. X1 and x2 are binary features, and y is a binary label:

P(x1=1 | y=0) = 2/3
P(x2=1 | y=0) = 1/3

P(x1=1 | y=1) = ¼
P(x2=1 | y=1) = ½

P(y=1) = 0.5

Under Naïve Bayes assumption, what is P(y=1 | x1=1, x2=1)?
Under Naïve Bayes assumption, is it possible to calculate P(y=1 | x1=0, x2=0)?
Without the Naïve Bayes assumption, is it possible to calculate P(y=1 | x1=1, x2=1)?

Today’s Lecture

- Linear models: linear logistic regression and linear regression
  - What are the models
  - How to train
  - How to evaluate
  - When to use them

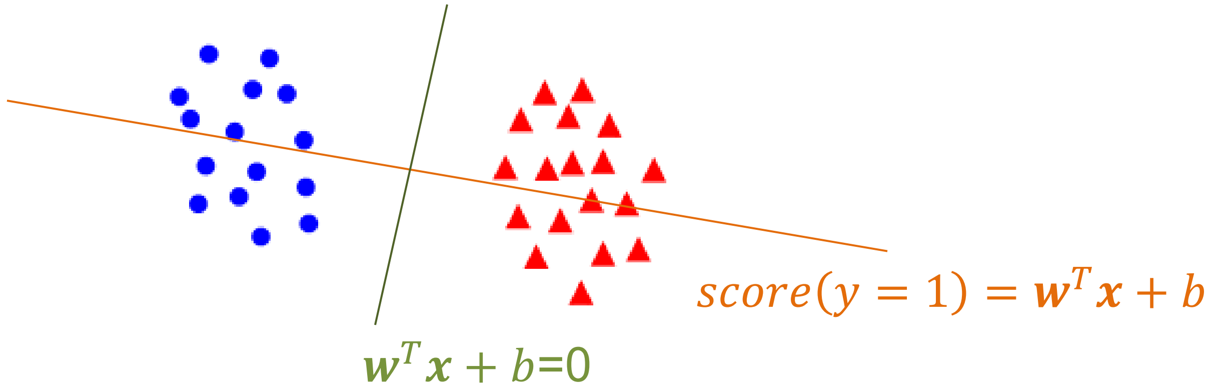
- Regularization
  - How it affects weights
  - How to solve for regularization parameters

Linear Models

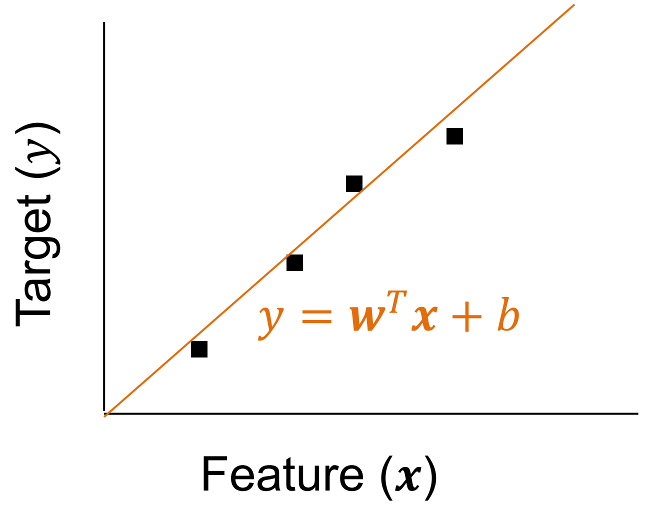
- A model is **linear** in x if it is based on a weighted sum of the values of x (optionally, plus a constant)

$$\boldsymbol{w}^T\boldsymbol{x} + b = \left[\sum_i w_i x_i\right] + b$$

- A **linear classifier** projects the features onto a score that indicates whether the label is positive or negative (i.e., one class or the other). We often show the boundary where that score is equal to zero.
- A **linear regressor** finds a linear model that approximates the prediction value for each set of features.

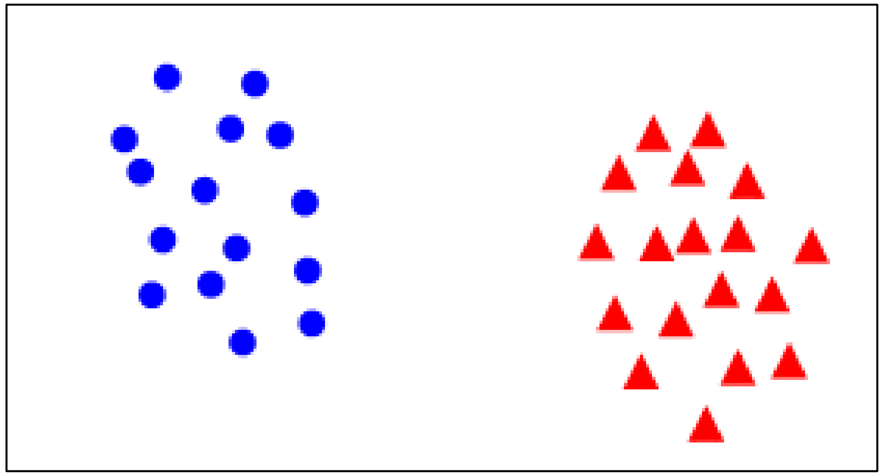


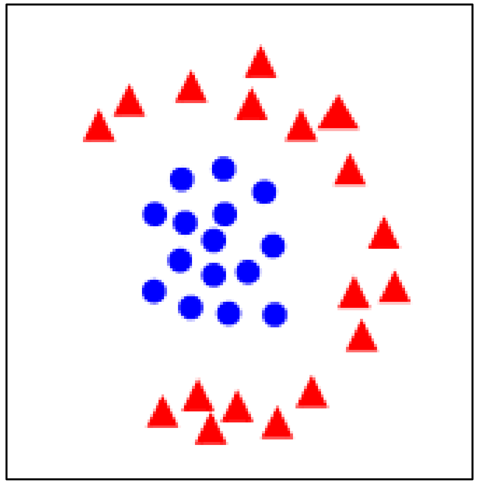
$score(y=1) = \boldsymbol{w}^T\boldsymbol{x} + b$
$\boldsymbol{w}^T\boldsymbol{x} + b$=0



Linear Classifiers and Linear Separability

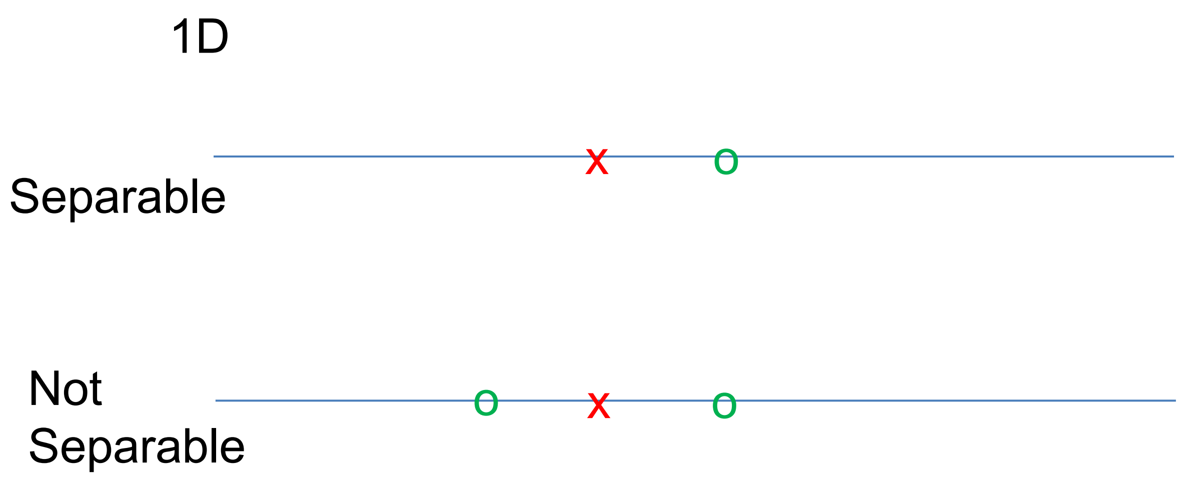
- **Linear classifier**: $y = 1 \text{ if } \boldsymbol{w}^T \boldsymbol{x} + b > 0$
- **Linearly separable**: a line (or hyperplane) in feature space can split the two labels
- Which of these are linearly separable?

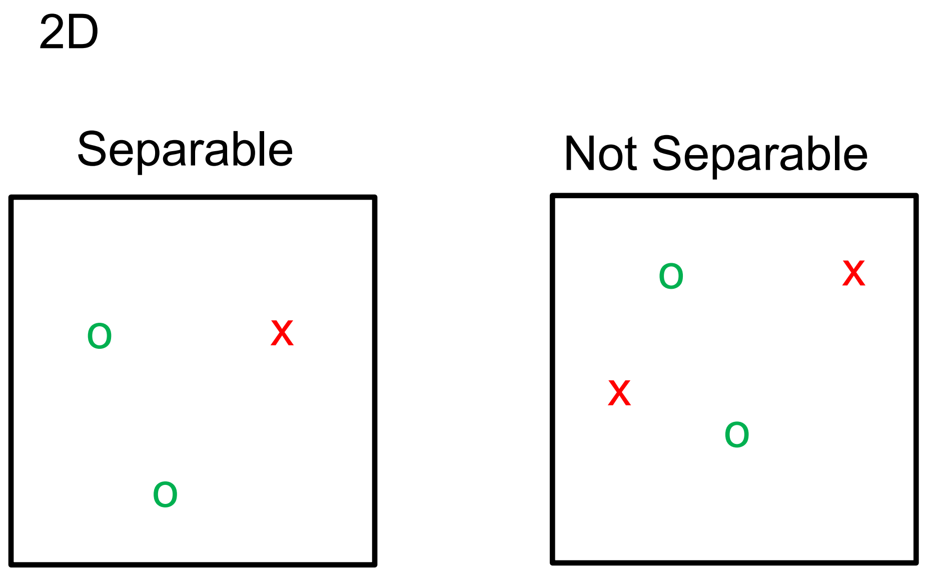




Linear Classifiers and Linear Separability

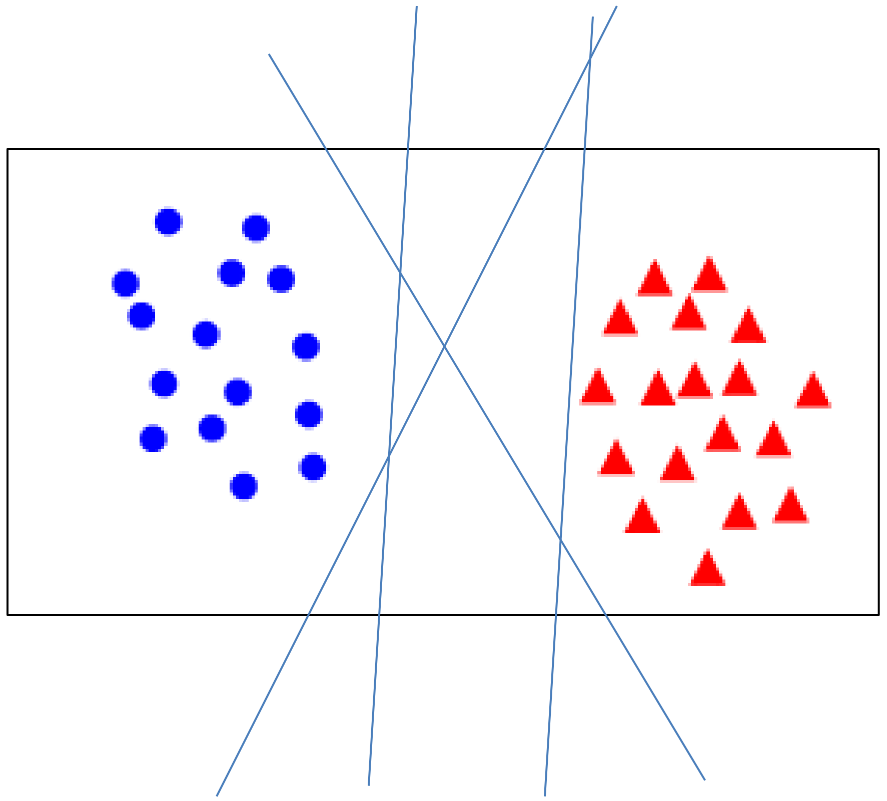
- In high dimensions, a lot more things are linearly separable
- If you have D dimensions, you can separate D+1 points with any arbitrary labeling

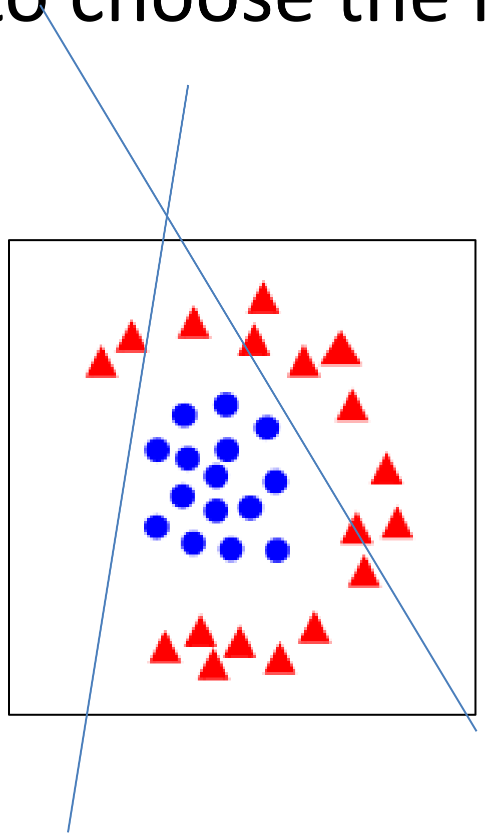




Linear Classifiers and Linear Separability

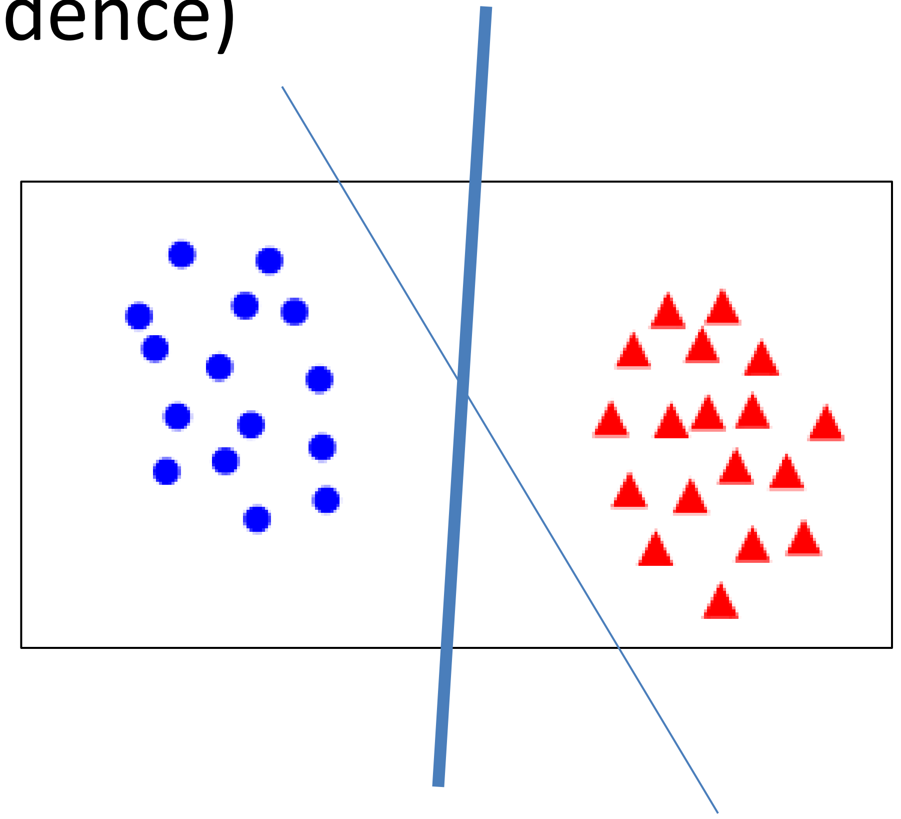
- But how do you choose which line is best?
- Different classifiers use different objectives to choose the line





Linear Classifiers and Linear Separability

- Different classifiers use different objectives to choose the line
- Common principles are that you want training samples on the correct side of the line (low classification error) by some margin (high confidence)



Thick line is better classification function than thin line because all the examples have a good margin

(Linear) Logistic Regression Model

maximize $P(y|x)$

where $P(y=1|x) = 1 / 1+\exp(-[w^T x + b])$ $\leftarrow$ "Logistic function"

$w^T x + b \approx \log \frac{P(y=1|x)}{P(y=0|x)}$ $\leftarrow$ "Logit"

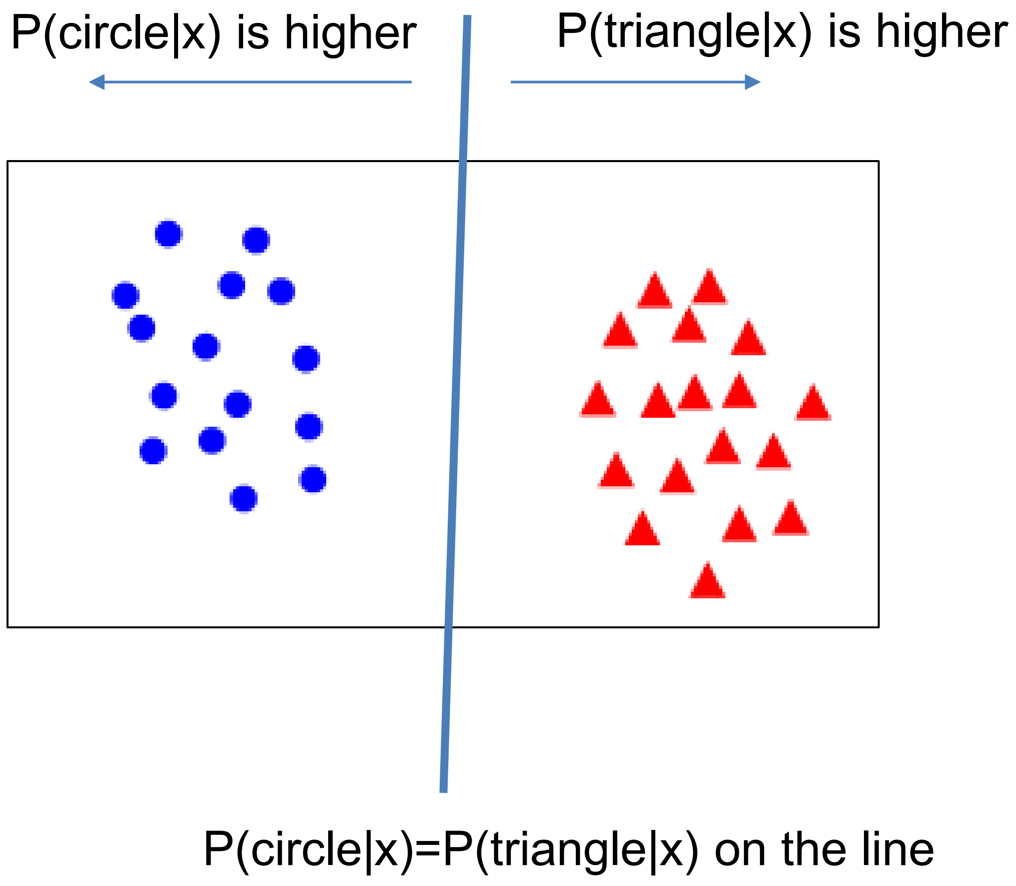
* To simplify notation, I may omit the "b", which can be avoided by adding a "1" to each feature vector

LLR vs. Naïve Bayes

- For many probability functions (exponential family, which includes binomial and Gaussian), Naïve Bayes predictor is also linear in x, but the parameters are independently estimated per feature
- The **logistic regression model is typically more expressive than Naïve Bayes**
- Logistic regression directly fits a discriminative function (i.e. f(x)→y) rather than trying to model P(x|y)

Linear Logistic Regression

- The further you are from the line, the more confident in a label



Linear Logistic Regression algorithm

- Training

$$w^* = \underset{w}{\operatorname{argmin}} - \sum_n \log P(y = y_n | x_n; w) + r(w)$$

regularization

log probability of all labels given features, assuming that each example is i.i.d.

- Prediction

$y = 1$ if $w^T x > 0$

$$P(y = 1|x) = \frac{1}{1+\exp(-w^T x)} = \frac{\exp(w^T x)}{\exp(w^T x)+1}$$

Binary (two-class) case

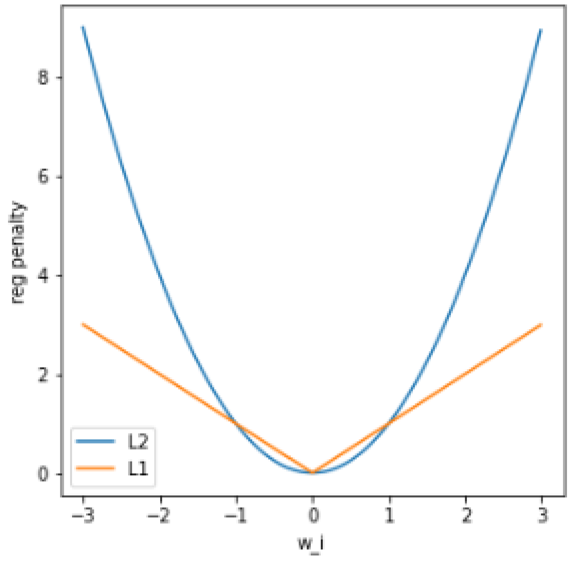
$$P(y = k|x) = \frac{\exp(w_k^T x)}{\sum_j \exp(w_j^T x)}$$

Multiclass case (one w per class)

Training Logistic Regression

$$w^* = \underset{w}{\operatorname{argmin}} - \sum_n \log P_w(y = y_n | x_n) + r(w)$$

- L2 regularization: $r(w) = \lambda \|w\|_2^2 = \lambda \sum_i w_i^2$
- L1 regularization: $r(w) = \lambda \|w\|_1 = \lambda \sum_i |w_i|$



L2 wants no really big weights
L1 does not want a lot of little weights

L1 leads to a sparse weight vector (many zeros) – why?

L1 regularization can be used to select features!

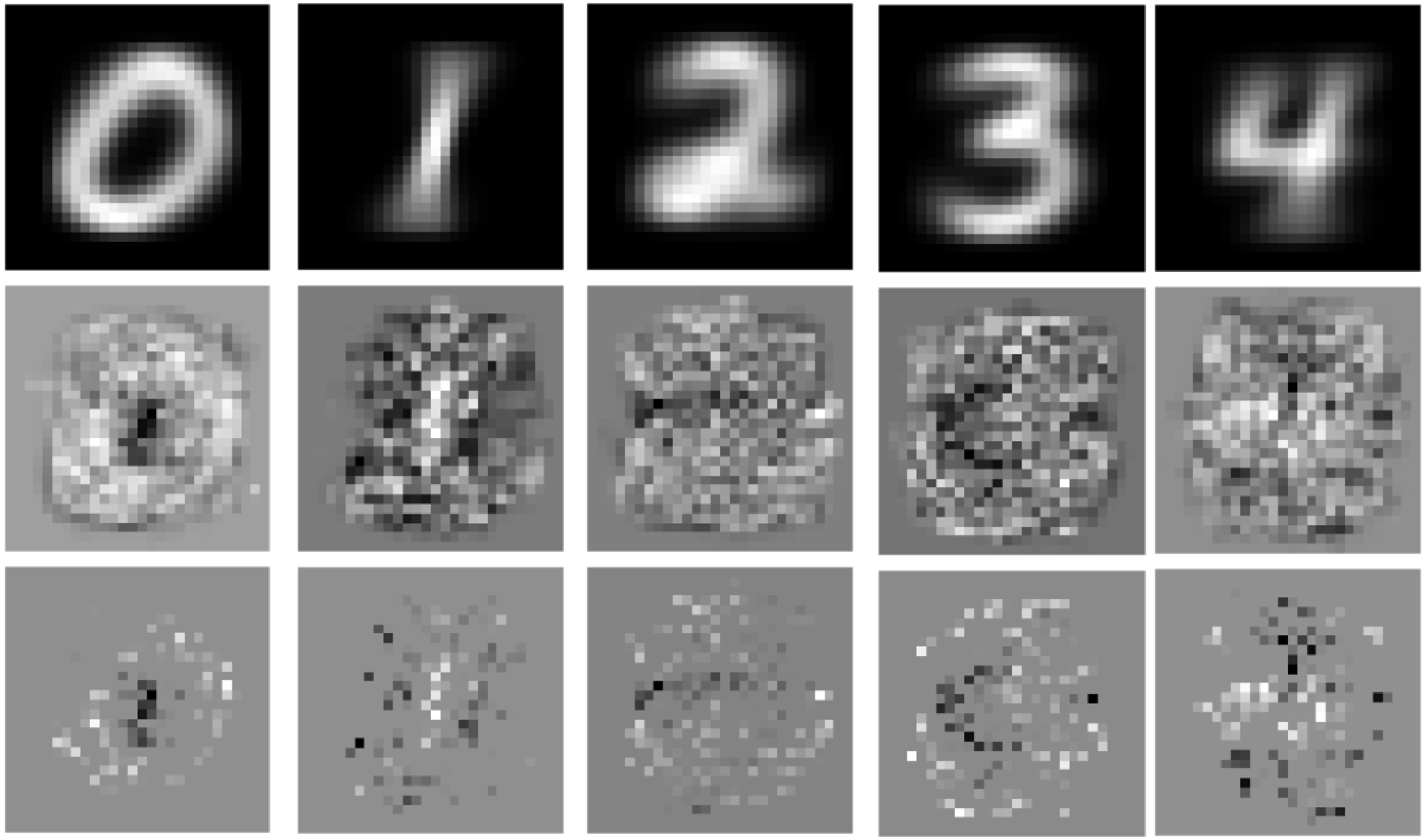
When is regularization absolutely essential?

There are many optimizers for L2 and L1 logistic regression. You will want to use a library.

Inspecting weights for digits

```
model_lr2 = LogisticRegression(max_iter=500, penalty='l2',C=1,verbose=2).fit(x_train[train_indices['m']],y_train[train_indices['m']])
model_lr1 = LogisticRegression(max_iter=500, penalty='l1',C=1,verbose=2,solver='saga').fit(x_train[train_indices['m']],y_train[train_indices['m']])
```

```
for k in np.arange(10):
  print(model_lr1.classes_[k])
  display_mnist(x_train[y_train==k].mean(axis=0))
  display_mnist(model_lr2.coef_[k])
  display_mnist(model_lr1.coef_[k])
```



How to perform model selection for hyperparameters

“Hyperparameters” are part of the objective function, not something that training data can fit.

E.g., the regularization weight λ is a hyperparameter

How to perform model selection for hyperparameters

1. For λ in {1/8, ¼, ½, 1, 2, 4, 8}:
   a. Train model with λ using training set
   b. Measure and record performance on validation set
2. Choose λ with best validation performance
3. (Optional) re-train on train + val sets – don’t need to do this for HW
4. Test final model on the test set

Tips

- In many cases, you want to vary parameters by factors, e.g. times 2 or times 5 for each candidate
- You can start search broad and then narrow, e.g. if ¼ and ½ are the best two, then try 3/8
- If you are searching over many parameters simultaneously, then it’s better to randomly sample candidate parameter values

Other forms of hyper-parameter selection

- Cross-validation
    1. Split training set into 5 parts
    2. Train on 4/5 and train on remaining fifth
    3. Repeat five times, using a different fifth for validation each time
    4. Choose hyperparameters based on average validation performance

    – Useful when you have limited training data
- Leave-one-out cross-validation (LOOCV) is an extreme where for N data points, you have N splits (one held out for validation each time)

Logistic Regression Summary

- Key Assumptions
  - The log odds ratio $\log \frac{P(y=k|x)}{P(y \neq k|x)}$ can be expressed as a linear combination of features
- Model Parameters
  - One coefficient per feature (plus one for bias or class prior)
- Designs
  - L1 or L2 or elastic (both L1 and L2) regularization weight
- When to Use / Strengths
  - Many features, some of which could be irrelevant or redundant
  - Provides a good estimate of label likelihood
- When Not to Use / Weaknesses
  - Features are low-dimensional (linear function not likely to be expressive enough)

Pause (2 min) and think/stretch

- Suppose I have two features, x1 and x2, that are each predictive of y.
  - $x2 = x1 + \epsilon$, where epsilon is a small amount of Gaussian noise

Suppose we train a logistic regressor, giving weights for each feature.

Q: Will the weights of x1 and x2 be similar:

a. under L2-regularized logistic regression?
b. under L1-regularized logistic regression?

$$w^* = \underset{w}{\operatorname{argmin}} -\sum_n \log P_w(y = y_n | x_n) + r(w)$$

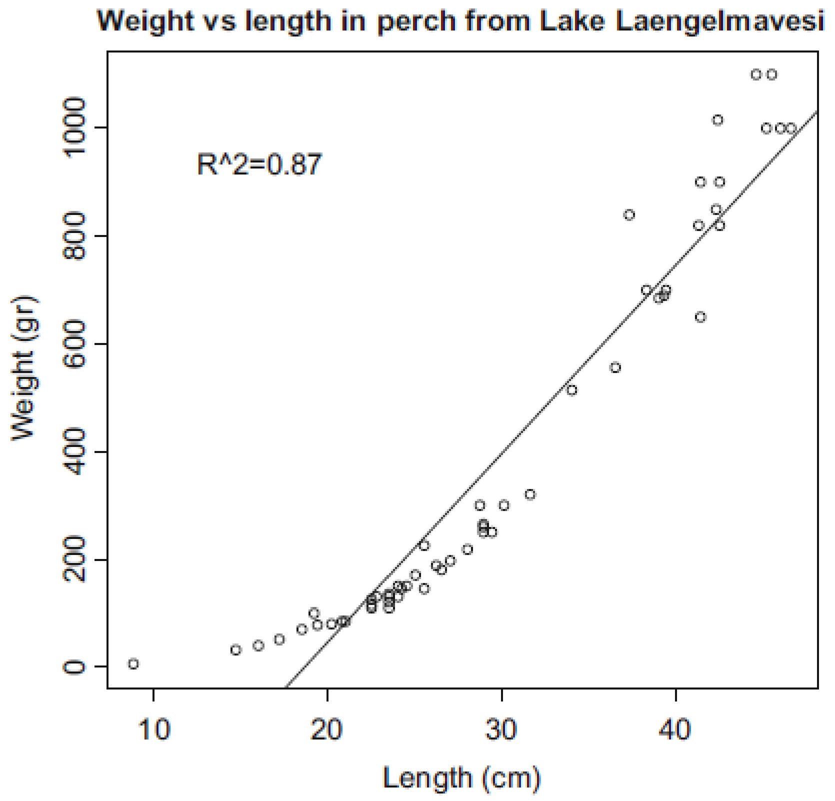
L2 regularization: $r(w) = \lambda \|w\|_2^2 = \lambda \sum_i w_i^2$

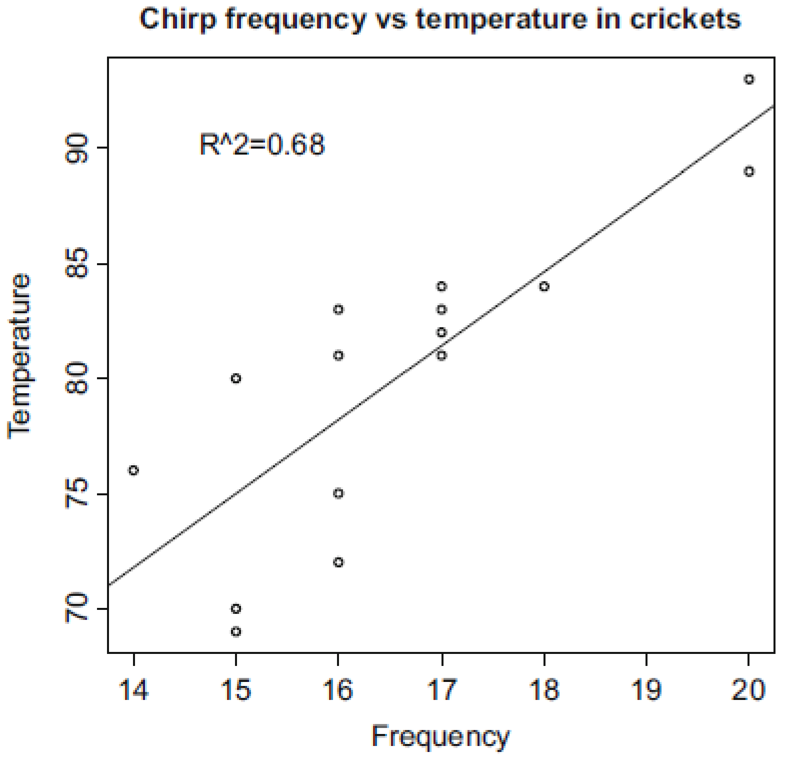
L1 regularization: $r(w) = \lambda \|w\|_1 = \lambda \sum_i |w_i|$

Examples from Forsyth AML

Linear regression

- Fit linear coefficients to features to predict a continuous variable





R^2: $1 - \sum_i (f(X_i) - y_i)^2 \,/\, \sum_i (y_i - \bar{\boldsymbol{y}})^2$

- R^2 close to zero indicates very weak relationship
- R^2 close to 1 indicates y very linearly predictable from **x**

Linear Logistic Regression algorithm

- Training

$$w^* = \underset{w}{\operatorname{argmin}} \textstyle\sum_n (w^T x - y_n)^2 + r(w)$$

- Prediction

$$y = w^T x$$

Linear regression

Model/Prediction

$$Ax+b=y \rightarrow A[x;1]=y$$

If y is 1 dim: $a^T[x;1]=y$

Solve with least squares:

Training (least squares)

$$a^* = \underset{\underline{a}}{\operatorname{argmin}} \sum_{n}^{N} (a_0 x_{n0} + \ldots + a_m x_{nm} - y_n)^2$$

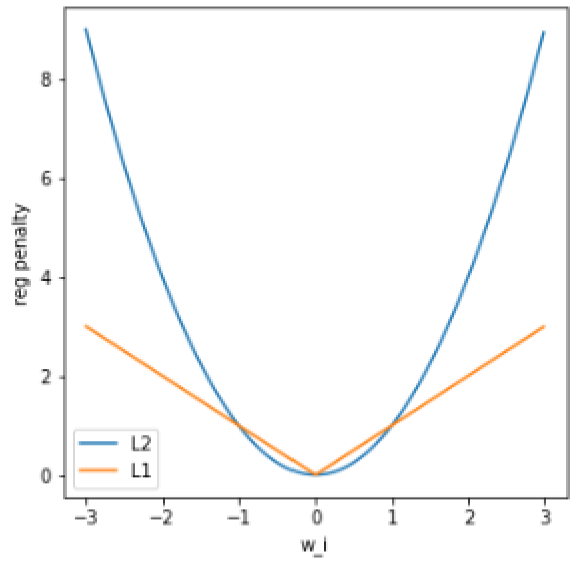
$$= \underset{\underline{a}}{\operatorname{argmin}} (\underset{\substack{\uparrow \\ n\times m}}{X} \underset{\substack{\uparrow \\ m\times 1}}{a} - \underset{\substack{\uparrow \\ n\times 1}}{y})^T (Xa-y)$$

$$\Rightarrow a^* = X^{\dagger} y \qquad X^{\dagger} = (X^T X)^{-1} X^T y$$

Training Linear Regression

$$w^* = \underset{w}{\operatorname{argmin}} \sum_n (w^T x - y_n)^2 + r(w)$$

- L2 regularization: $r(w) = \lambda \|w\|_2^2 = \lambda \sum_i w_i^2$
- L1 regularization: $r(w) = \lambda \|w\|_1 = \lambda \sum_i |w_i|$



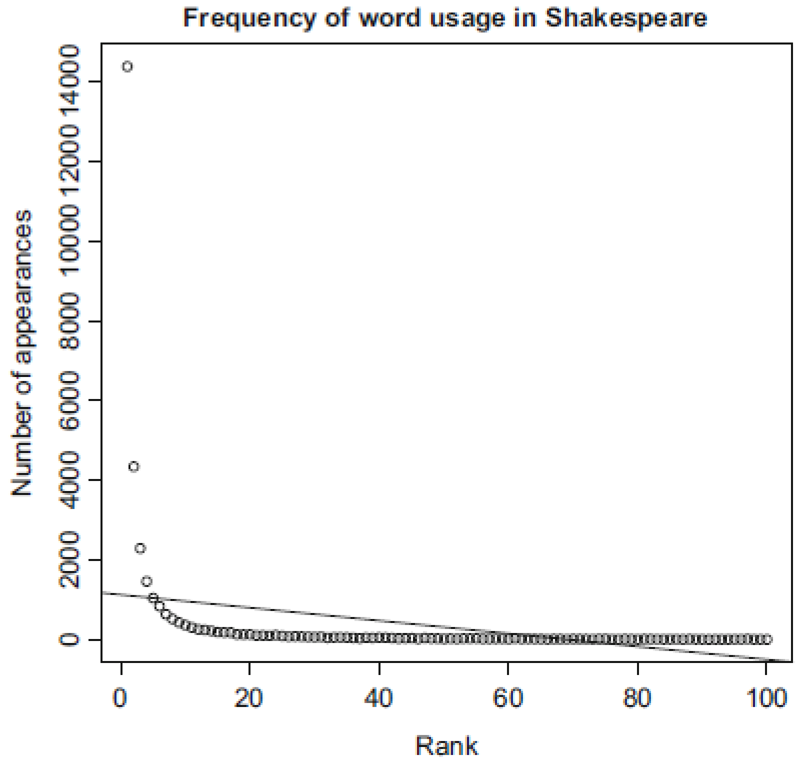
L2 wants no really big weights
L1 does not want a lot of little weights

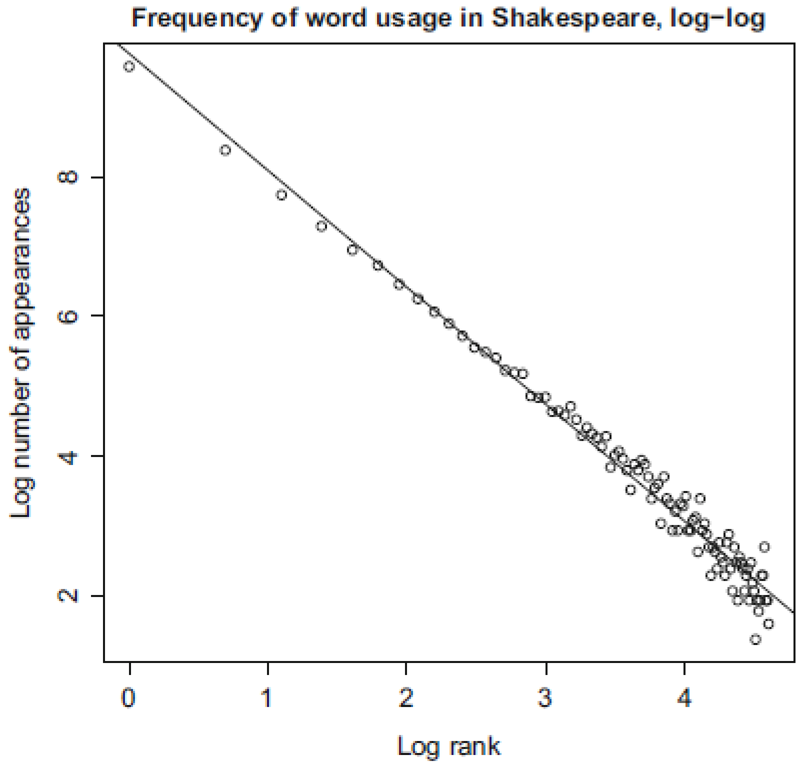
L1 regularization can be used to select features

L2 linear regression is least squares and not hard to implement, but you can use a library.

Transforming variables

- Sometimes you need to transform variables before fitting a linear model
  - Helpful to plot histograms or distributions of individual features to figure out what transformations might apply

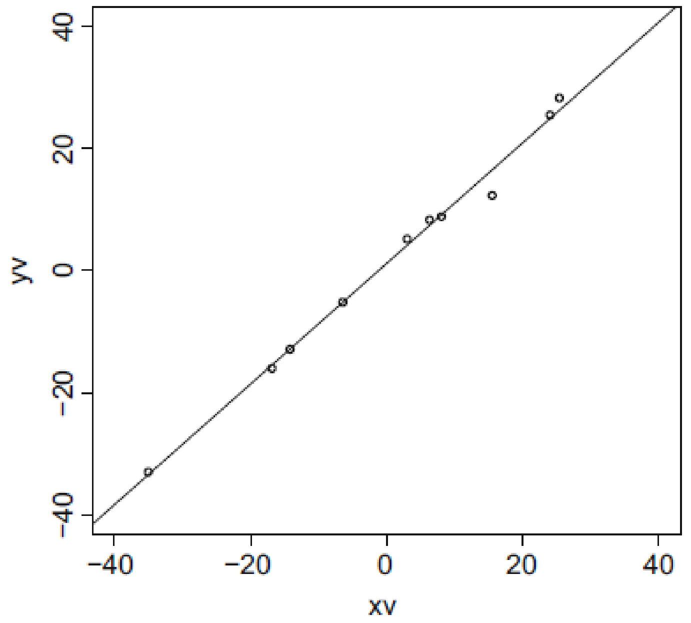


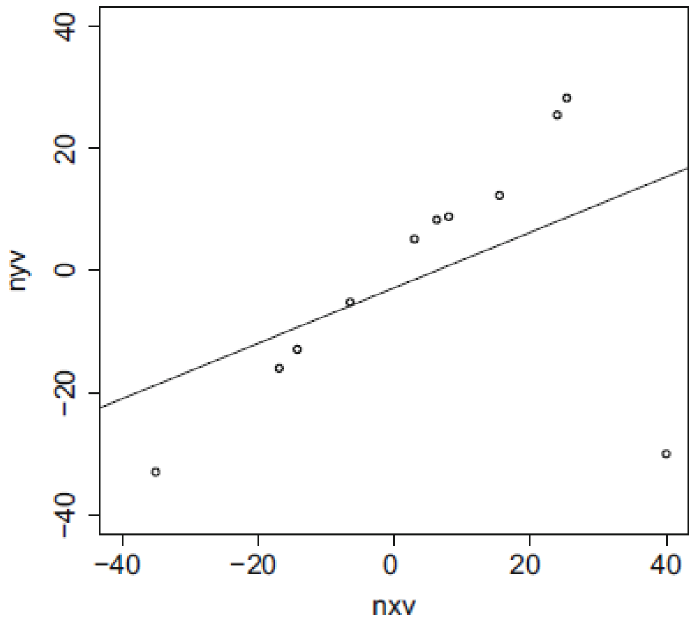


Example from Forsyth AML

Linear regression can be sensitive to outliers

- There are robust forms that estimate a weight on each sample, e.g. m-estimation
- Minimizing the sum of absolute error does not have this problem, but is a harder optimization

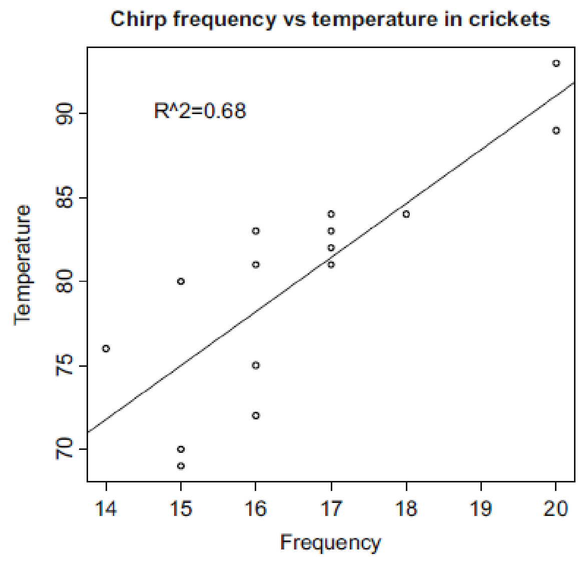




Example from Forsyth AML

Linear regression vs KNN vs Naïve Bayes

- KNN can fit non-linear functions
- Only linear regression can extrapolate
- Linear regression is higher bias, lower variance
- Linear regression is more useful to explain a relationship
- Linear regression is more powerful (should always fit training data better, but not necessarily fit test data better) than Gaussian Naïve Bayes



Linear Regression Summary

- Key Assumptions
  - y can be predicted by a linear combination of features
- Model Parameters
  - One coefficient per feature (plus one for y-intercept)
- Designs
  - L1 or L2 or elastic (both L1 and L2) regularization weight
  - Different objective functions (e.g. squared error, absolute error, m-estimation)
- When to Use
  - Want to extrapolate
  - Want to visualize or quantify correlations/relationships
  - Have many features
- When Not to Use
  - Relationships are very non-linear (requires transformation or feature learning first)

When are these used?

- Logistic regression is the default classification decoder (e.g. it is the last layer of neural network classifiers)

- Linear regression is used to explain data or predict continuous variables in a wide range of applications

Recap

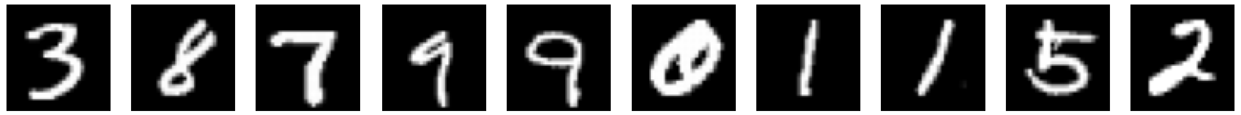
- Nearest neighbor is widely used
  - Super-powers: can instantly learn new classes and predict from one or many examples
- Naïve Bayes represents a common assumption as part of density estimation, more typical as part of an approach rather than the final predictor
  - Super-powers: Fast estimation from lots of data; not terrible estimation from limited data
- Logistic Regression is widely used
  - Super-powers: Effective prediction from high-dimensional features
- Linear Regression is widely used
  - Super-powers: Can extrapolate, explain relationships, and predict continuous values from many variables
- Almost all algorithms involve nearest neighbor, logistic regression, or linear regression
  - The main learning challenge is typically **feature learning**

HW 1 review

MNIST Digit Classification

Predict label (0-9) from pixel intensities (784x1 vector)

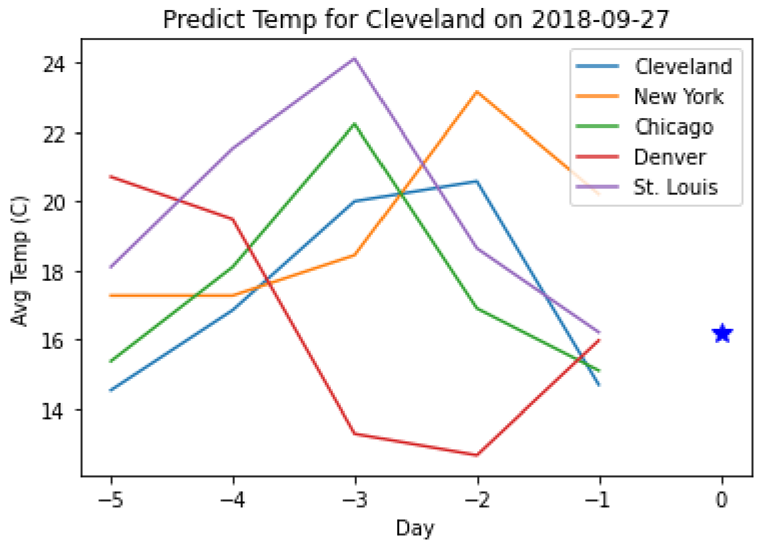
1. Implement and test KNN, Naïve Bayes, Linear Logistic Regression
2. Plot Error vs Training Size
3. Select best parameter using validation



Temperature Regression

Predict Cleveland's next day temperature from recent temperatures of US cities

1. Implement and test KNN, Naive Bayes (NB), and Linear Regression (LR)
2. Identify most important features with L1 linear regression, and re-train/evaluate with most important features



Stretch Goals

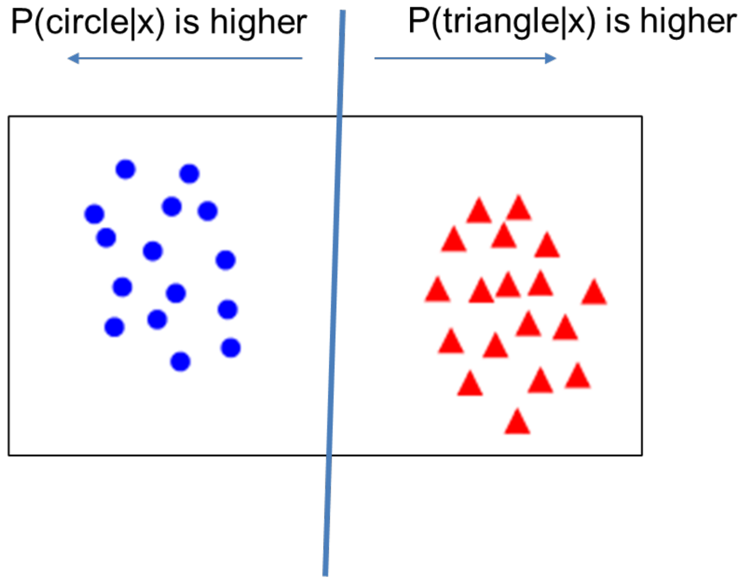
1. Improve MNIST Classification
2. Improve Temperature Regression
3. Generate a train/test classification dataset in which Naive Bayes outperforms 1-NN and Logistic Regression

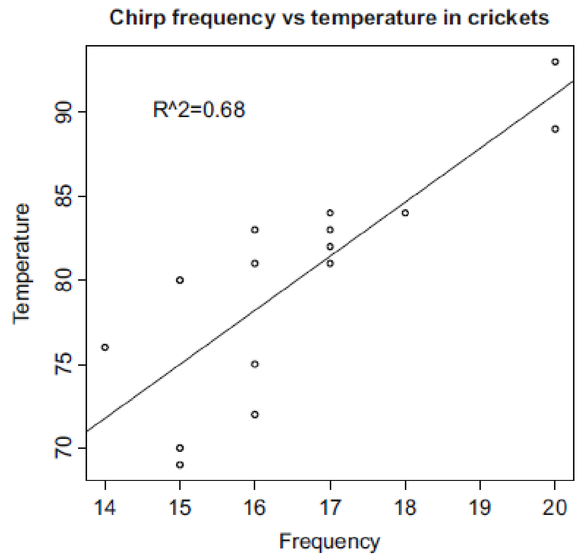
Completing HWs

- Read assignment and tips
- Code by adding to starter code notebook (which mainly has data loading and visualization functions)
- Complete report, including expected points
- Submit the report, notebook pdf/html, and zipped code
  - Mainly grader will look at report first, notebook pdf for clarification, and zipped code rarely
  - Notebook does not need to include all outputs

Things to remember

- Linear logistic regression is a classification technique that aims to split features between two classes with a linear model
  - Predict categorical values with confidence
- Linear regression fits a linear model to a set of feature points to predict a continuous value
  - Explain relationships
  - Predict values
  - Extrapolate observations
- Nearest neighbor and linear models are the final predictors of most ML algorithms – the complexity lies in finding features that work well with NN or linear models





Next week

- Trees
- Ensembles