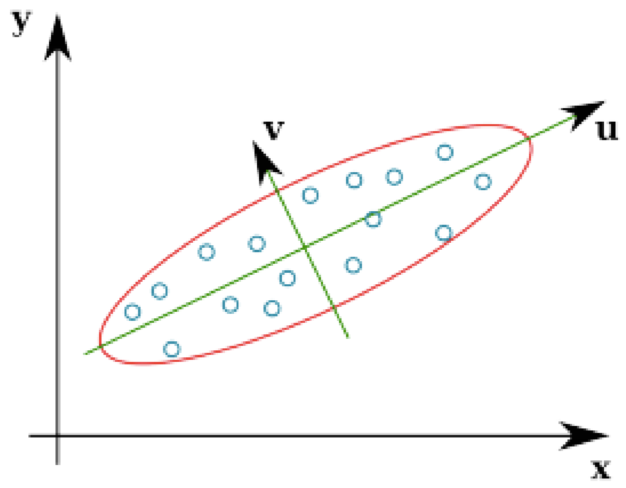

Dall-E

Linear Regression and Regularization

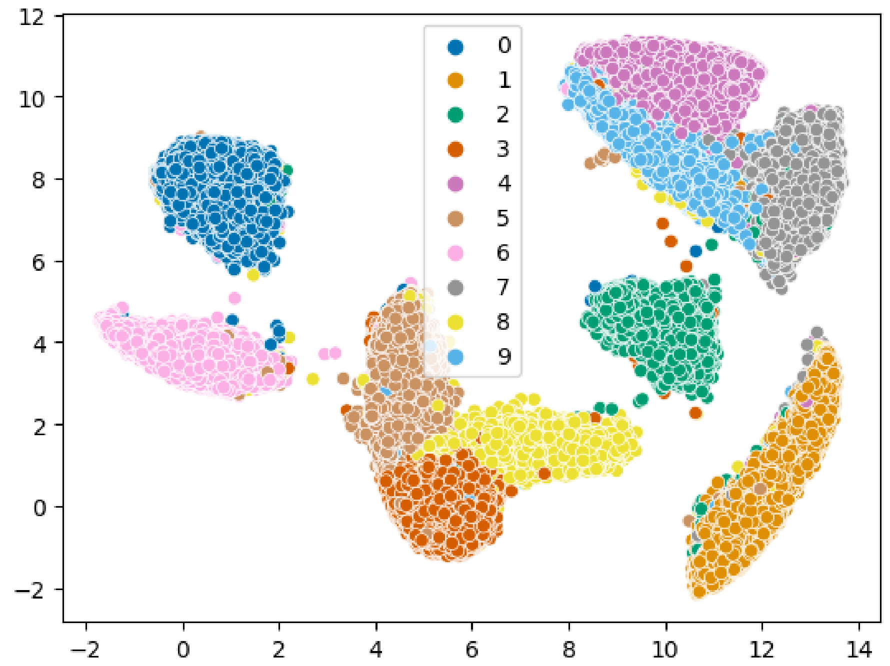
Applied Machine Learning
Derek Hoiem

Last class

- PCA reduces dimensions by linear projection
  - Preserves variance to reproduce data as well as possible, according to mean squared error
  - May not preserve local connectivity structure or discriminative information



- Other methods try to preserve relationships between points
  - MDS: preserve pairwise distances
  - IsoMap: MDS but using a graph-based distance
  - t-SNE: preserve a probabilistic distribution of neighbors for each point (also focusing on closest points)
  - UMAP: incorporates k-nn structure, spectral embedding, and more to achieve good embeddings relatively quickly



Today’s Lecture

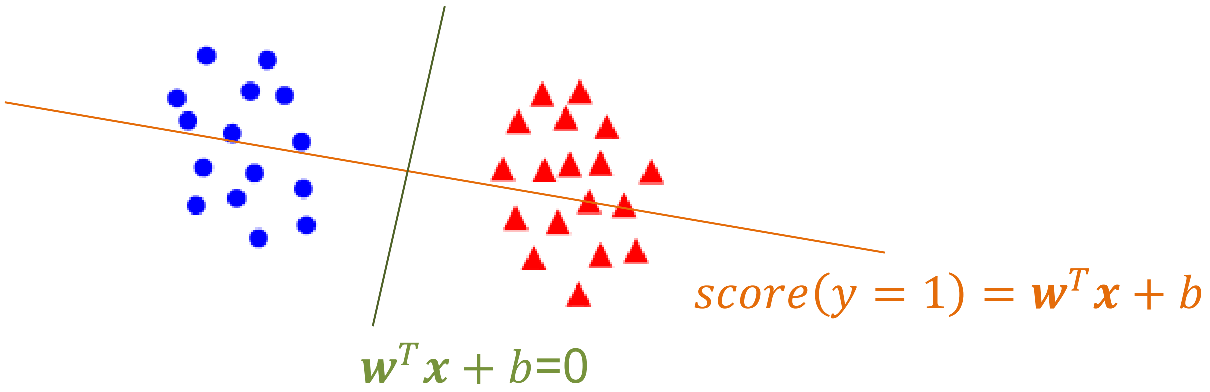
- Linear regression
- Regularization
  - What is it
  - How it affects models
  - How to select regularization parameters

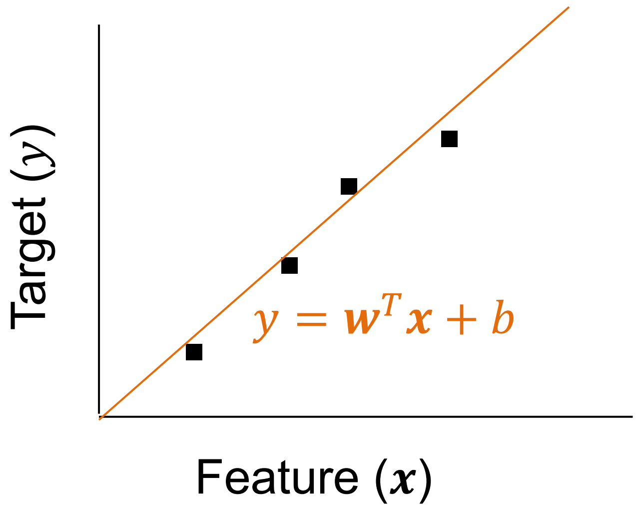
Linear Models

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

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

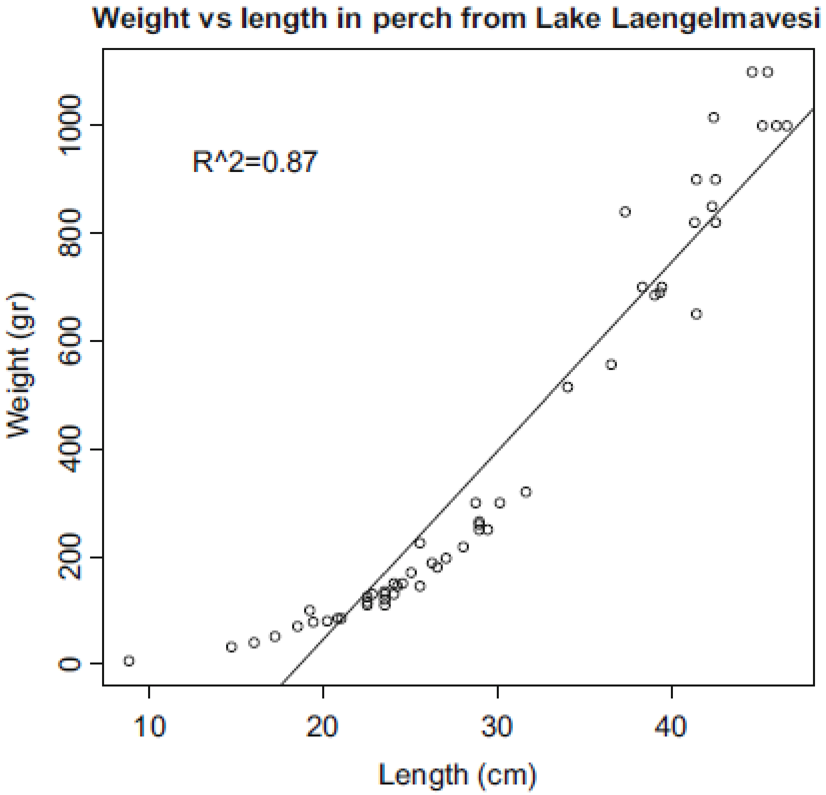
- A **linear classifier** projects the features onto a score that indicates whether the label is positive or negative (i.e., one class or the other). We often show the boundary where that score is equal to zero.
- A **linear regressor** finds a linear model that approximates the prediction value for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

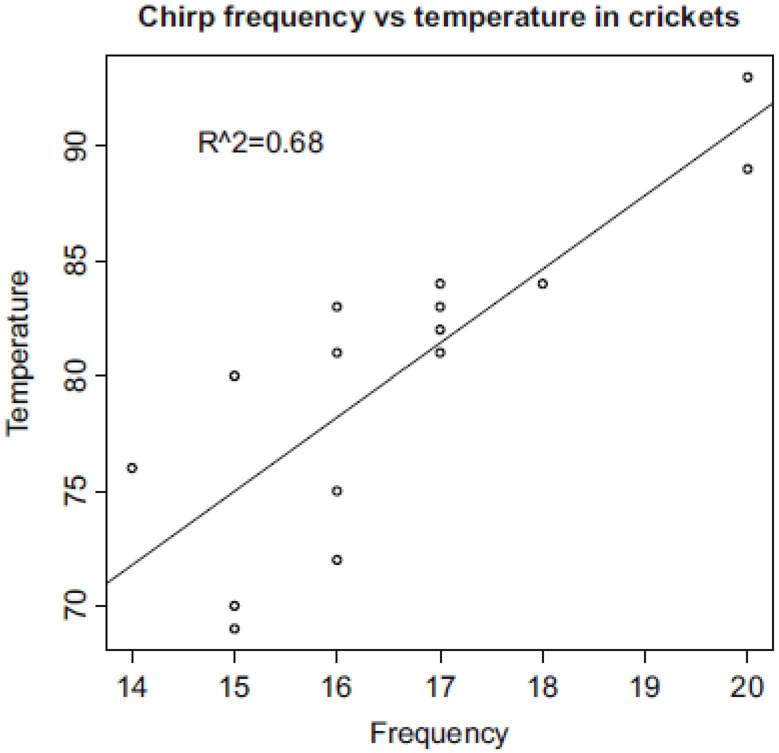




Linear regression

- 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 Regression algorithm

- Training: find $\boldsymbol{w}$ that minimizes sum square difference between target and prediction

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

Loss due to missing the target

Loss due to parameter complexity

- Prediction

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

Linear regression

Model/Prediction $\boldsymbol{w}^T[\boldsymbol{x}_n; 1] = y_n^{\boldsymbol{w}}$

1. Predicted y is linear function of x with a constant “bias” term
2. If predicting multiple values, use a separate $\boldsymbol{w}$ for each

Training (least squares)

$$\boldsymbol{w}^* = \mathrm{argmin}_{\boldsymbol{w}} \sum_{n=0}^{N} (\boldsymbol{w}^T[\boldsymbol{x}_n; 1] - y_n)^2$$

3. Minimize the squared error between predicted and true value

$$\boldsymbol{w}^* = \mathrm{argmin}_{\boldsymbol{w}} (\underset{N \times m}{X}\underset{m \times 1}{\boldsymbol{w}} - \underset{1 \times 1}{\boldsymbol{y}})^T (X\boldsymbol{w} - \boldsymbol{y})$$

4. Stack features/targets into matrix/vector

$$X\boldsymbol{w} \approx \boldsymbol{y}$$

5. We want $X\boldsymbol{w}$ to be close to $\boldsymbol{y}$ in a least squared sense

$$\boldsymbol{w} = X^{\dagger}\boldsymbol{y} = (X^T X)^{-1} X^T \boldsymbol{y}$$

6. Solution pseudoinverse of X times $\boldsymbol{y}$

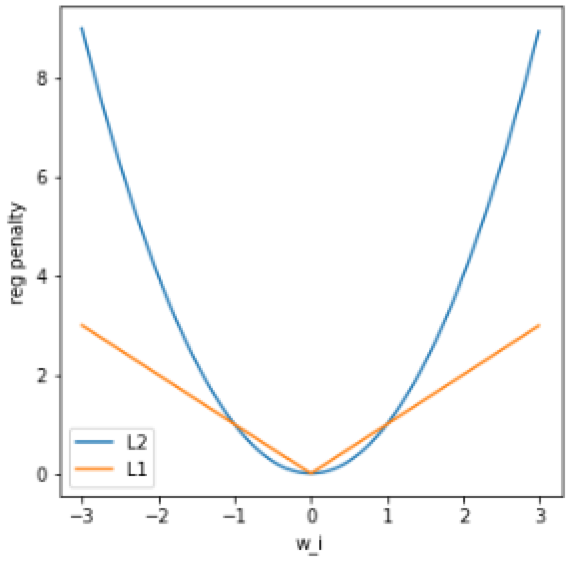
Q1

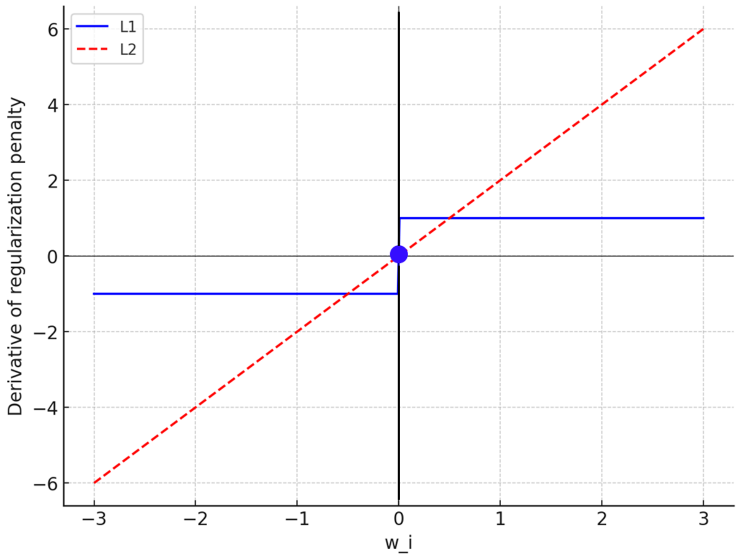
https://tinyurl.com/441-fa24-L6

Training Linear Regression

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

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





L2 strongly penalizes big weights

L1 penalizes increasing the magnitude of big and small weights the same (derivative is 1 for negative weights, -1 for positive weights, 0 for 0)

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.

Why regularize?

- Biases each weight toward 0: helps prevent noisy features from having too much influence
  - Bias: each weight toward 0
  - Reduced variance: less difference with different samples

- For L1, only most independently useful features have weight

How to select hyperparameters

"Hyperparameters" are part of the objective function or model design, not something that training data can fit.

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

How to select 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
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

Linear Regression Walkthrough

Diabetes Dataset from sklearn

Number of Instances: 442
Target: measure of disease progression one year after baseline
Number of Attributes: 10
Attribute Information:

- age: age in years
- sex
- bmi :body mass index
- bp: average blood pressure
- s1: tc, total serum cholesterol
- s2: ldl, low-density lipoproteins
- s3: hdl, high-density lipoproteins
- s4: tch, total cholesterol / HDL
- s5: ltg, possibly log of serum triglycerides level
- s6: glu, blood sugar level

Normalization: subtract mean, divide by std, divide by sqrt(n_samples)

Goals: regress target value, explain causes of progression

Regression paper (Efron et al. 2003)

Q2 – Q3

https://tinyurl.com/441-fa24-L6

Hyper-parameter selection with multiple variables

- Basic methods
  - Grid search: try all possible combinations
    - Becomes infeasible for more than two parameters
  - Line search: optimize each parameter by itself sequentially
    - Does not account for dependencies between parameters
  - Randomized search: randomly generate parameters within a range
    - Often best strategy for 3+ parameters

[Article by Prof. Arindam Banerjee]()

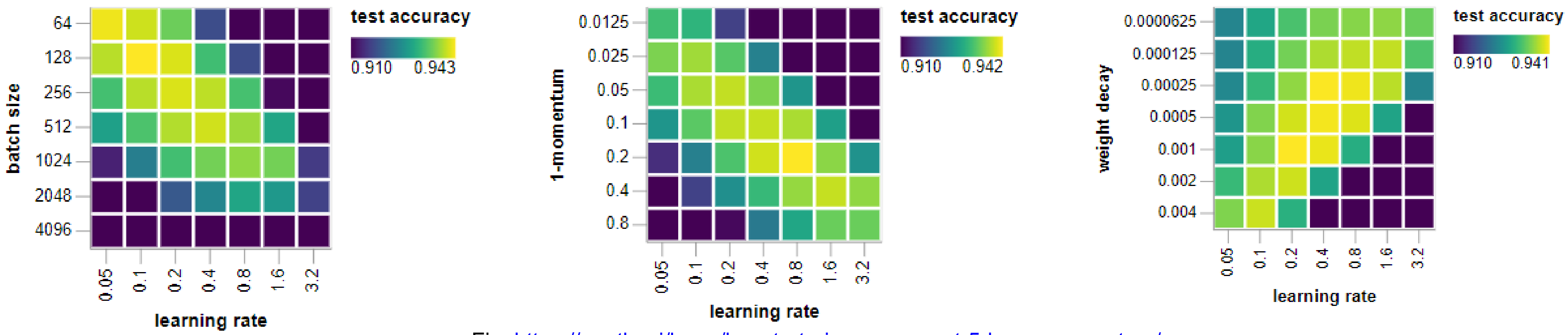


Fig: https://myrtle.ai/learn/how-to-train-your-resnet-5-hyperparameters/

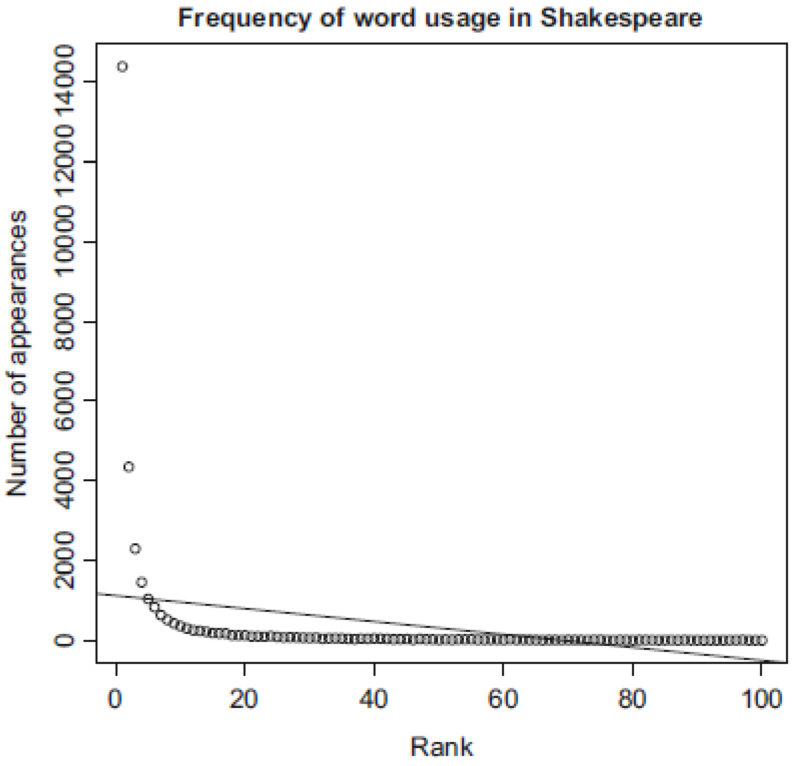
Other forms of hyper-parameter selection

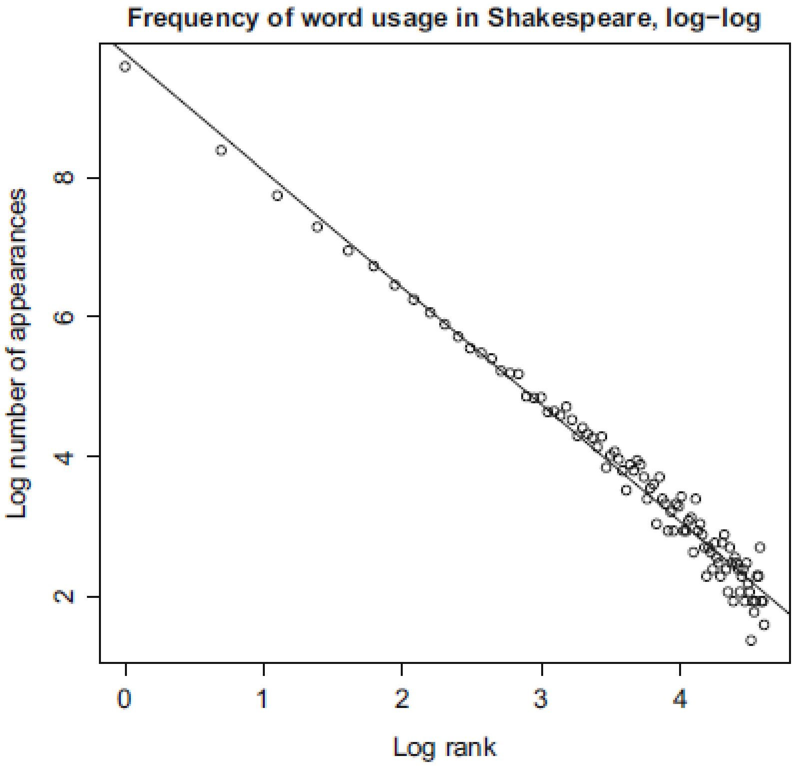
- Cross-validation
  1. Split training set into 5 parts (for 5-fold cross-validation)
  2. Train on 4/5 and train on remaining fifth
  3. Repeat five times, using a different fifth for validation each time
  4. Choose hyperparameters based on average validation performance
  – Useful when you have limited 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)

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

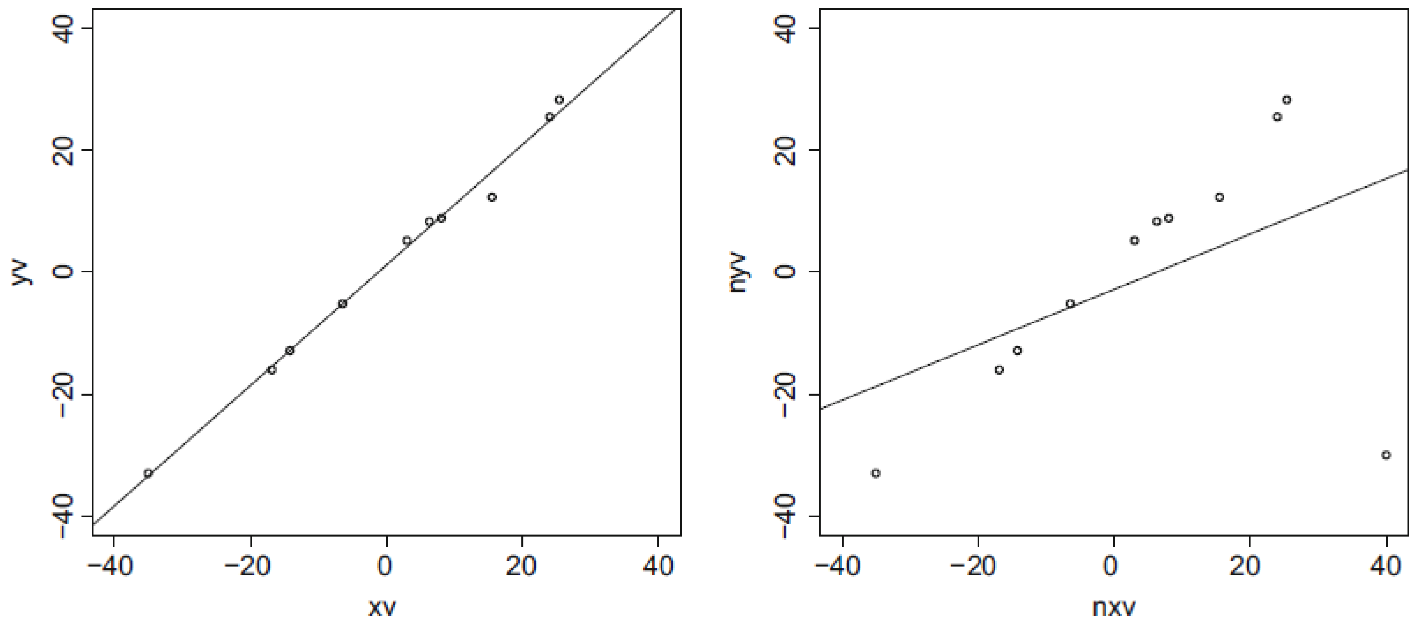




Example from Forsyth AML

Linear regression can be sensitive to outliers

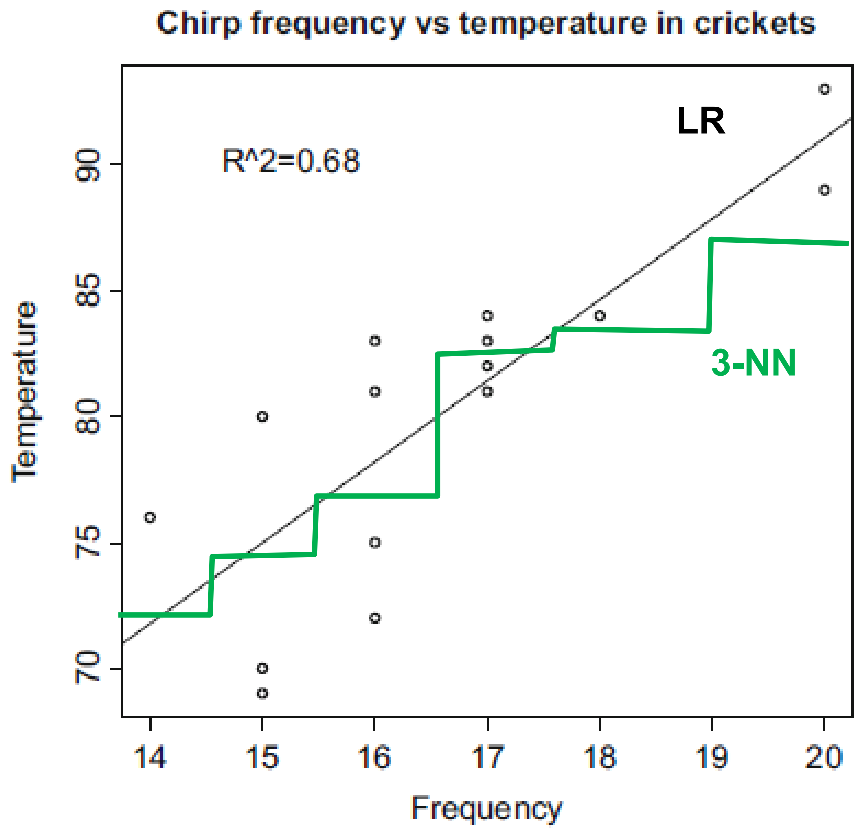
- There are robust forms that estimate a weight on each sample, e.g. m-estimation
- Minimizing the sum of absolute error does not have this problem, but is a harder optimization



Example from Forsyth AML

Linear regression vs KNN

- KNN can fit non-linear functions
- Only linear regression can extrapolate
- Linear regression is more useful to explain a relationship



Linear Regression Summary

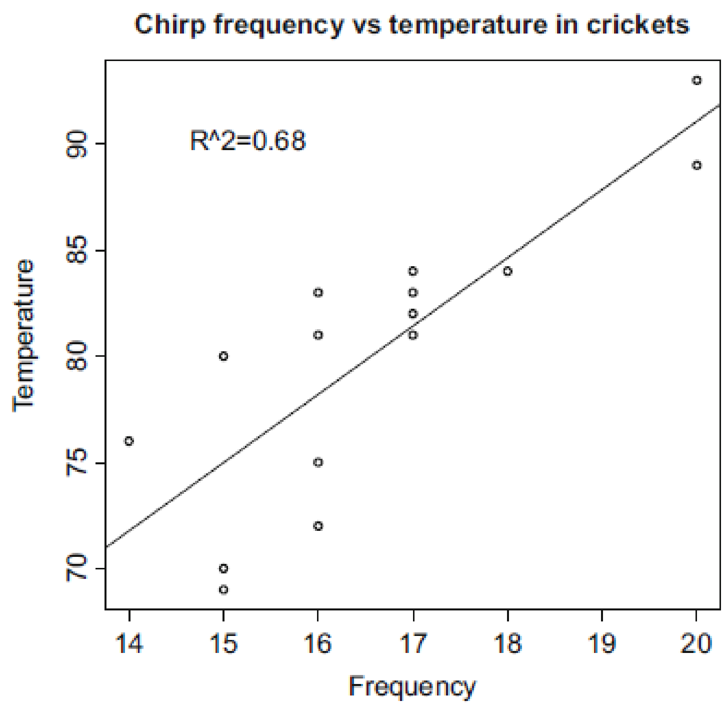
Linear regression is used to explain data or predict continuous variables in a wide range of applications

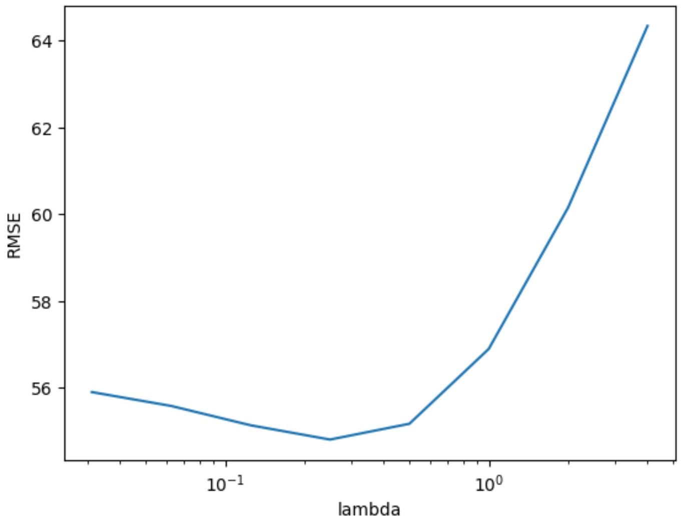
- Key Assumptions
  - y can be predicted by a linear combination of features
- Model Parameters
  - One coefficient per feature (plus one for y-intercept)
- Designs
  - L1 or L2 or elastic (both L1 and L2) regularization weight
  - Different objective functions (e.g. squared error, absolute error, m-estimation)
- 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)

Things to remember

- Linear regression fits a linear model to a set of feature points to predict a continuous value
  - Explain relationships
  - Predict values
  - Extrapolate observations

- Regularization prevents overfitting by restricting the magnitude of feature weights
  - L1: prefers to assign a lot of weight to the most useful features
  - L2: prefers to assign smaller weight to everything





Next week

- HW 1 due on Monday
- Linear classifiers
- Naïve Bayes Classifier