Clustering and Prediction

Probability and Statistics for Data Science

CSE594 - Spring 2016

But first,

One final useful statistical technique from Part II

Confidence Intervals

Motivation: p-values tell a nice succinct story but neglect a lot of information.

Estimating a point, approximated as normal (e.g. error or mean)

$$\hat{\mu} = \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i .$$

$$\mathrm{SE}_{\bar{x}} \ = \frac{s}{\sqrt{n}}$$

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}}$$

$$\left[\overline{x} - 1.96 \frac{\sigma}{\sqrt{n}}, \ \overline{x} + 1.96 \frac{\sigma}{\sqrt{n}} \right]$$

find CI% based on standard normal distribution   (i.e. CI% = 95, z = 1.96)

Resampling Techniques Revisited

The bootstrap

- What if we don’t know the distribution?

Resampling Techniques Revisited

The bootstrap

- What if we don’t know the distribution?
- *Resample* many potential distributions based on the observed data and find the range that CI% of the data fall in (e.g. mean).

Resample: for each *i* in *n* observations, put all observations in a hat and draw one (all observations are equally likely).

Clustering and Prediction

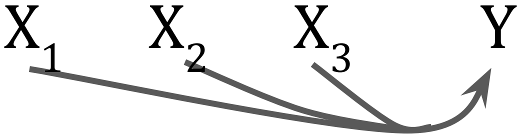
(now back to our regularly scheduled program)

I. Probability Theory

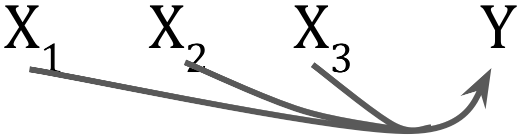
II. Discovery: Quantitative Research Methods

III. Clustering and Prediction

(now back to our regularly scheduled program)

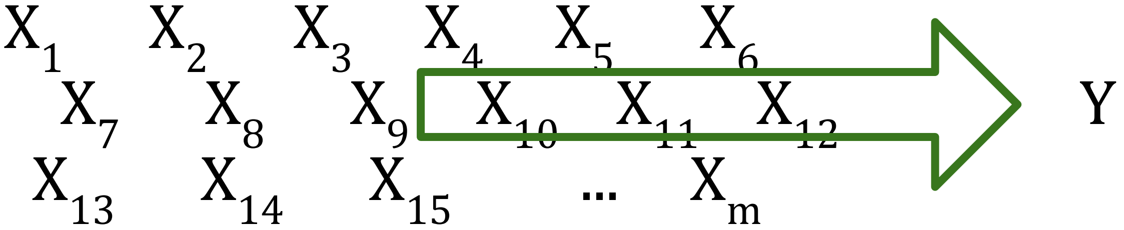


Clustering and Prediction



X_1 X_2 X_3 Y

#Discovery

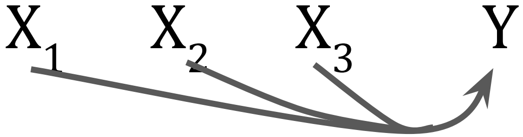


X_1 X_2 X_3 X_4 X_5 X_6

X_7 X_8 X_9 X_{10} X_{11} X_{12} Y

X_{13} X_{14} X_{15} … X_m

Clustering and Prediction

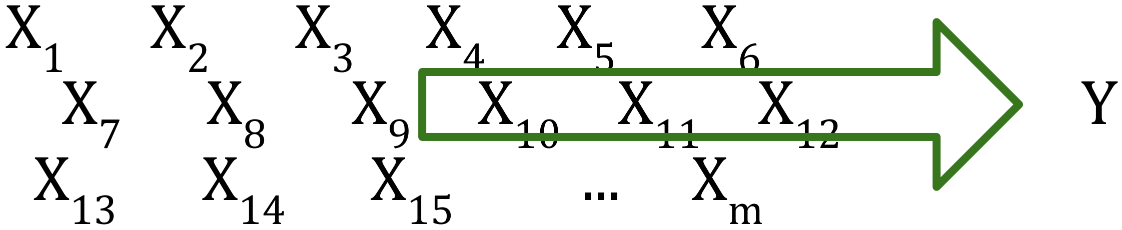


X_1 X_2 X_3 Y

#Discovery

$M <$ ~5 or $m << n$

(much less)



X_1 X_2 X_3 X_4 X_5 X_6

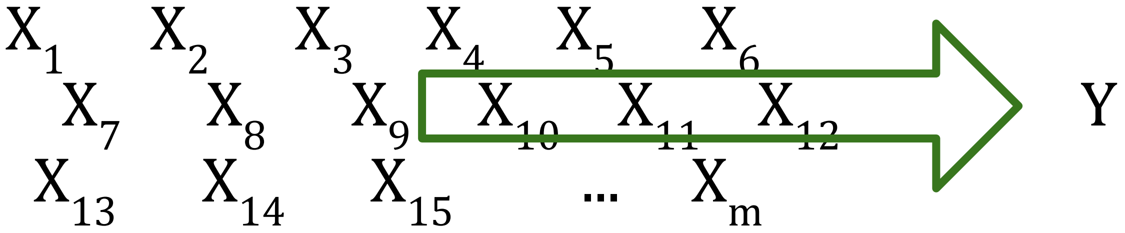
X_7 X_8 X_9 X_{10} X_{11} X_{12}

X_{13} X_{14} X_{15} ... X_m

Y

$M >$ ~100 or m □ n or $m >> n$

Clustering and Prediction

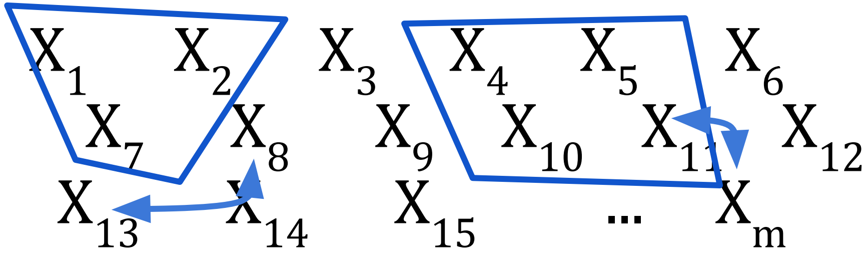


X_1 X_2 X_3 X_4 X_5 X_6

X_7 X_8 X_9 X_{10} X_{11} X_{12} Y

X_{13} X_{14} X_{15} ... X_m

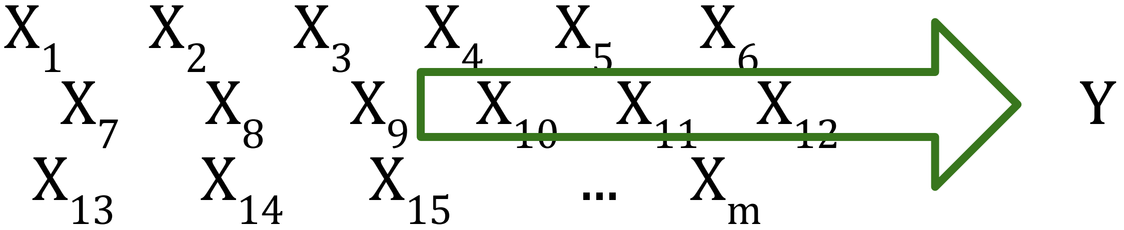
Clustering and Prediction



X_1 X_2 X_3 X_4 X_5 X_6

X_7 X_8 X_9 X_{10} X_{11} X_{12}

X_{13} X_{14} X_{15} ... X_m

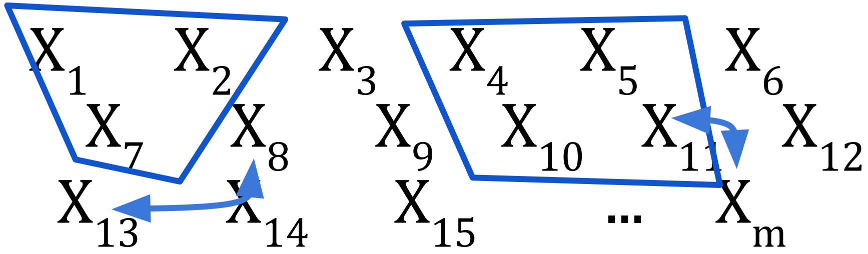


X_1 X_2 X_3 X_4 X_5 X_6

X_7 X_8 X_9 X_{10} X_{11} X_{12} Y

X_{13} X_{14} X_{15} ... X_m

Clustering and Prediction

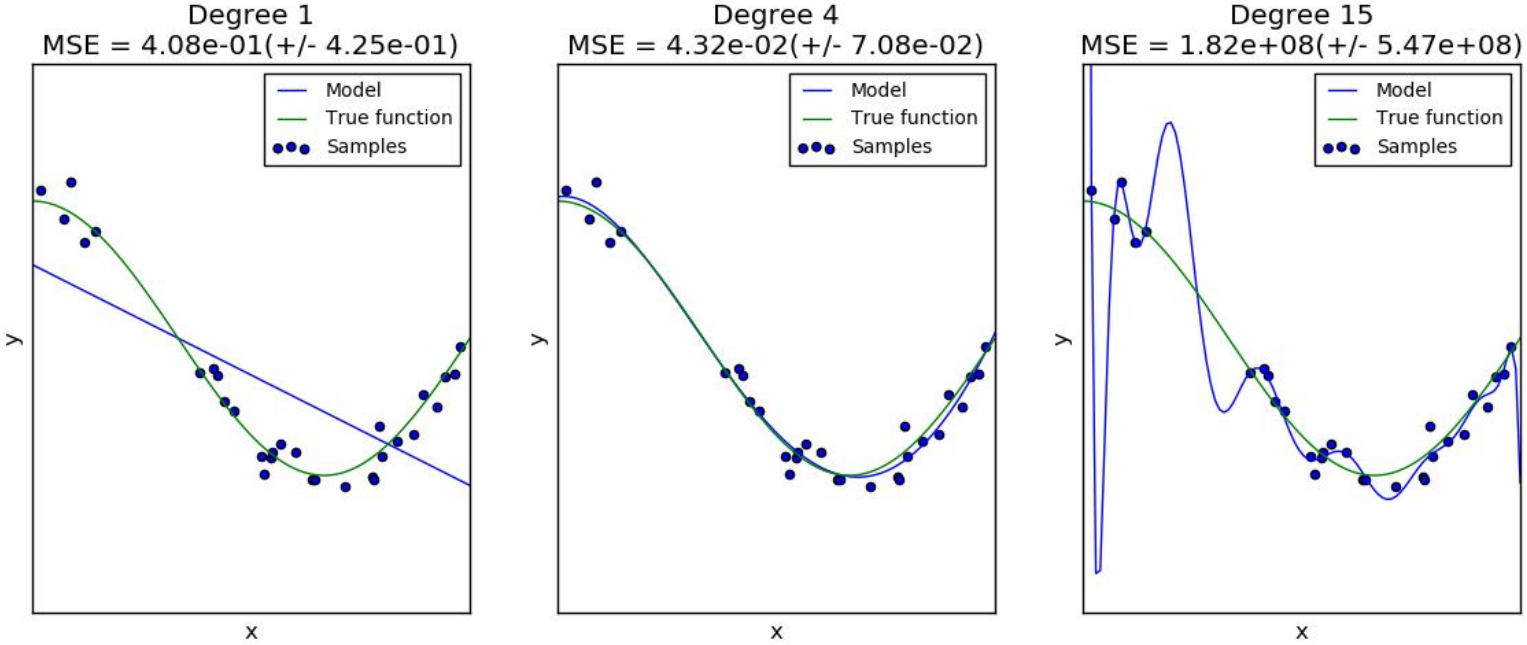


X_1 X_2 X_3 X_4 X_5 X_6

X_7 X_8 X_9 X_{10} X_{11} X_{12}

X_{13} X_{14} X_{15} ... X_m

Overfitting (1-d example)



Underfit

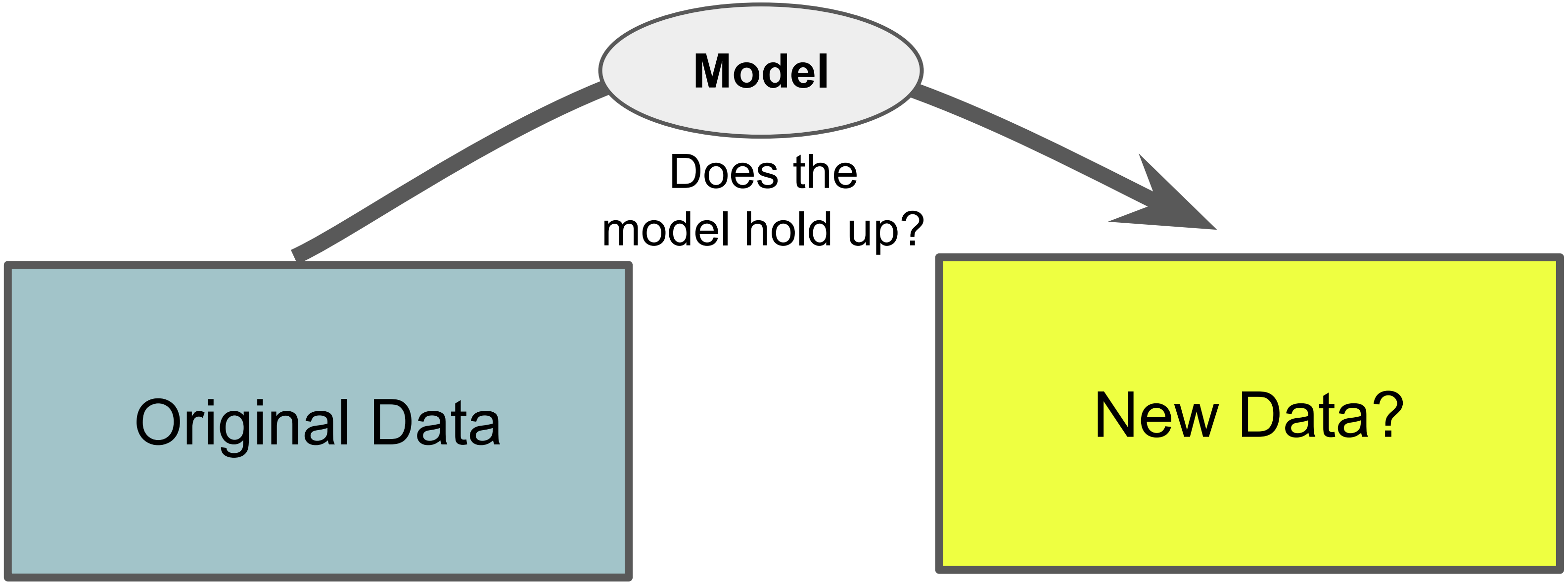
High Bias

Overfit

High Variance

(image credit: Scikit-learn; in practice data are rarely this clear)

Common Goal: Generalize to new data



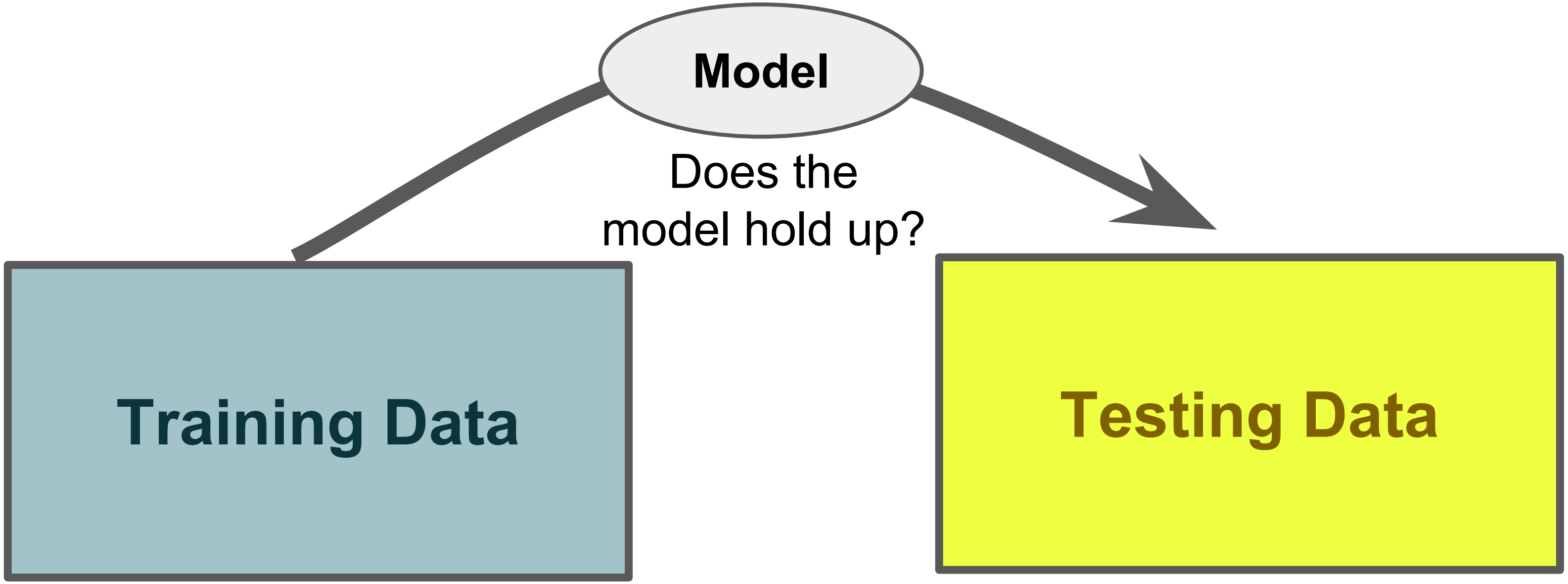
Model

Does the model hold up?

Original Data

New Data?

Common Goal: Generalize to new data



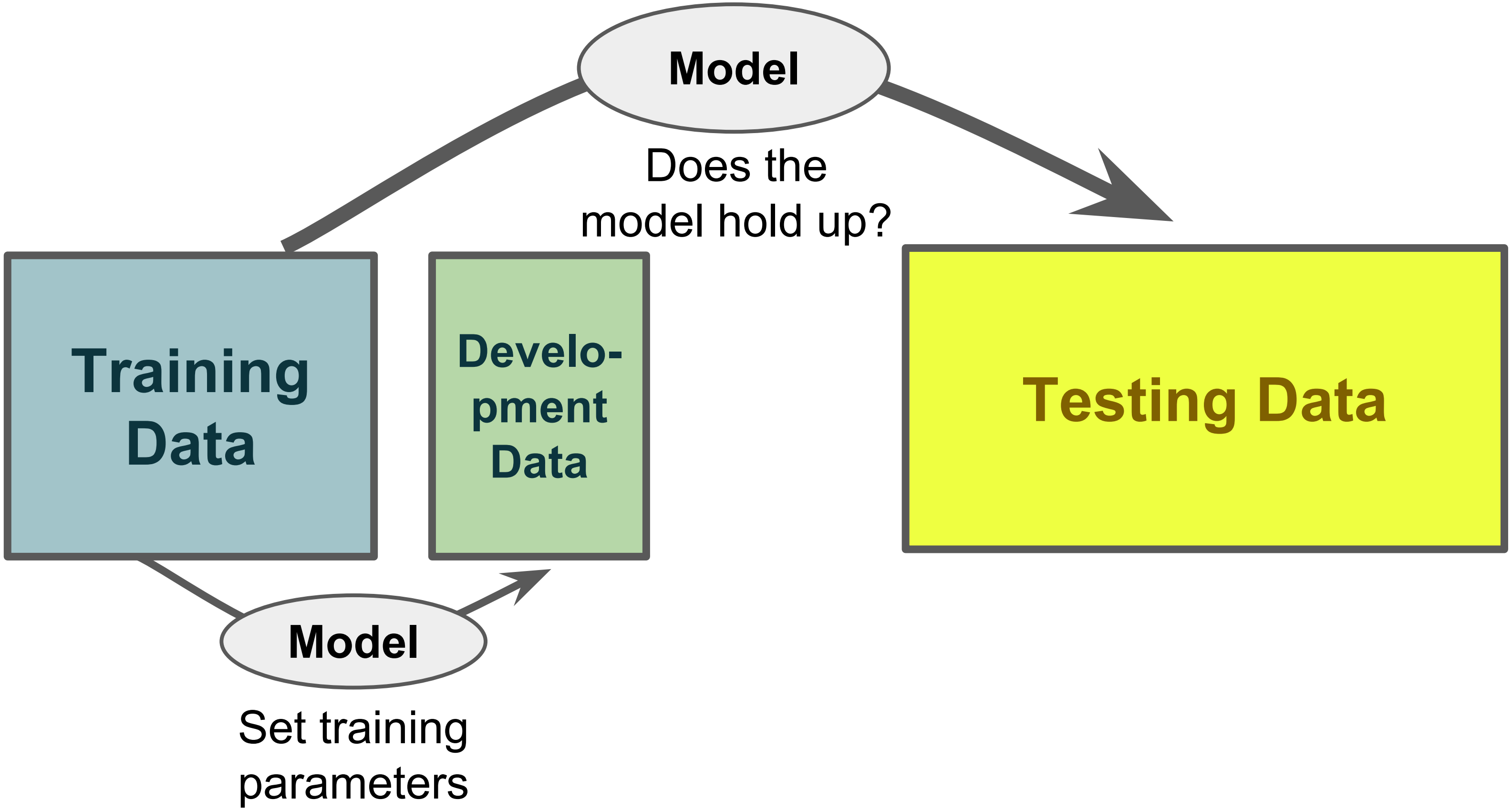
Model

Does the model hold up?

Training Data

Testing Data

Common Goal: Generalize to new data



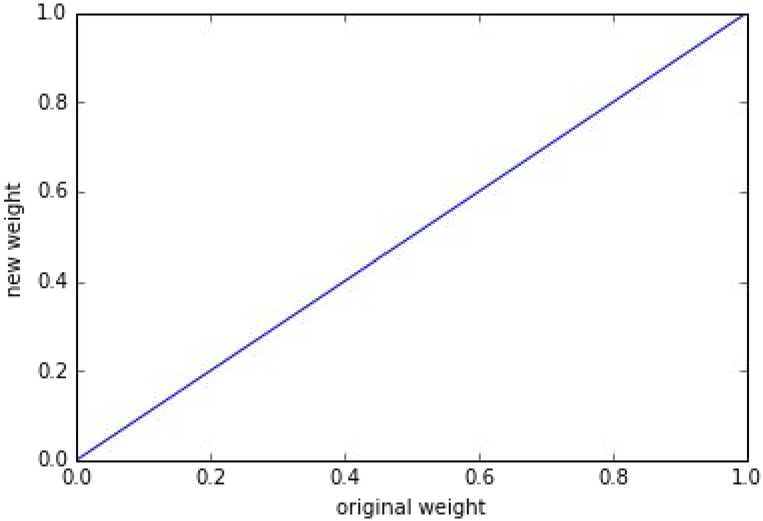
Feature Selection / Subset Selection

Forward Stepwise Selection:

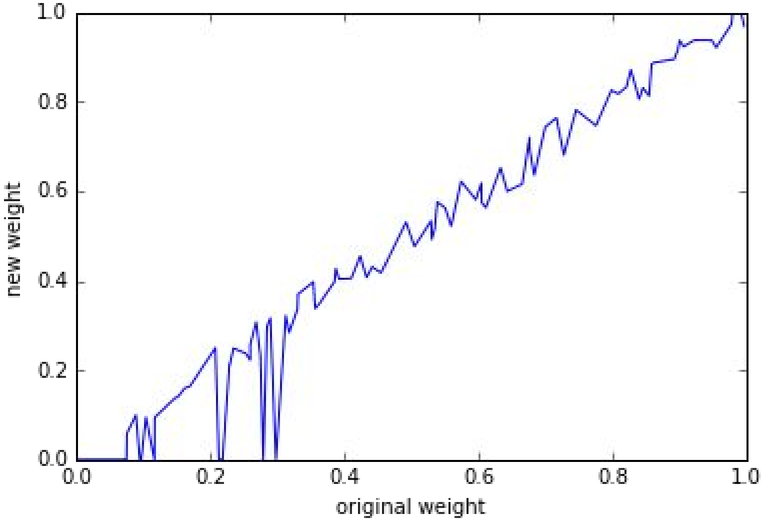
- start with current_model just has the intercept (mean)
  remaining_predictors = all_predictors
- for i in range(k)

```
        #find best p to add to current_model:
        for p in remaining_prepdictors
                refit current_model with p
        #add best p, based on $RSS_p$ to current_model
        #remove p from remaining predictors
```

Regularization (Shrinkage)



No selection (weight=beta)



forward stepwise

Why just keep or discard features?

Regularization (L2, Ridge Regression)

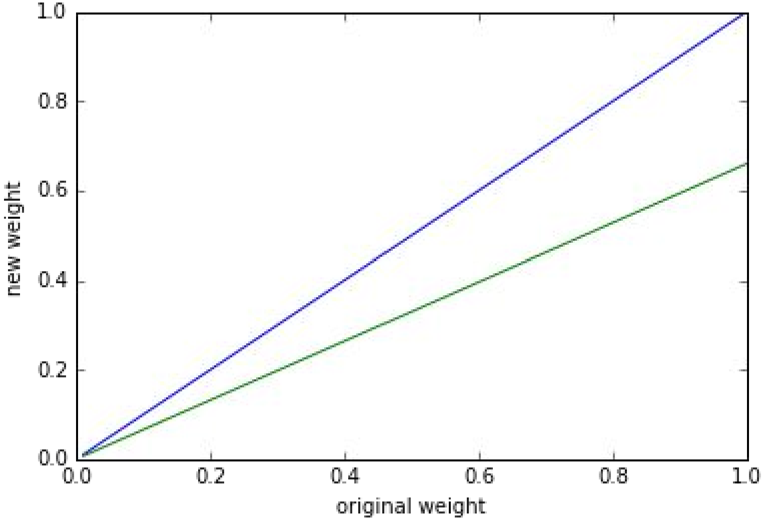
Idea: Impose a penalty on size of weights:

Ordinary least squares objective:

$$\hat{\beta} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2\}$$

Ridge regression:

$$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{m}\beta_j^2\}$$



Regularization (L2, Ridge Regression)

Idea: Impose a penalty on size of weights:

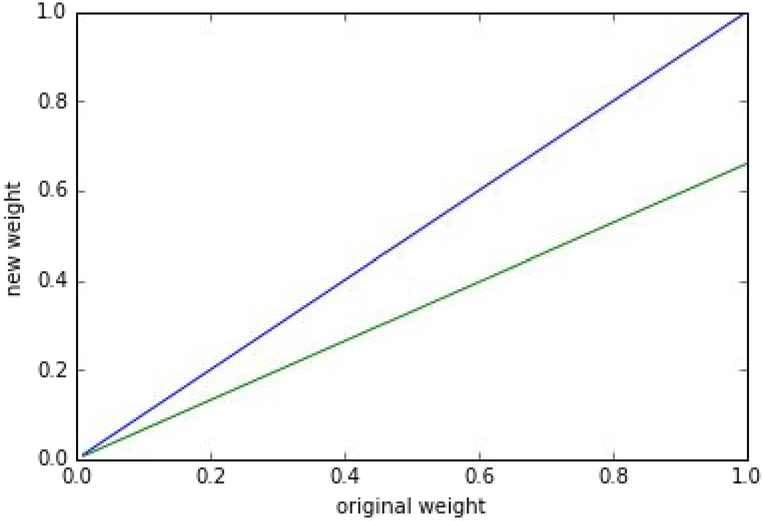
Ordinary least squares objective:

$$\hat{\beta} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2\}$$

Ridge regression:

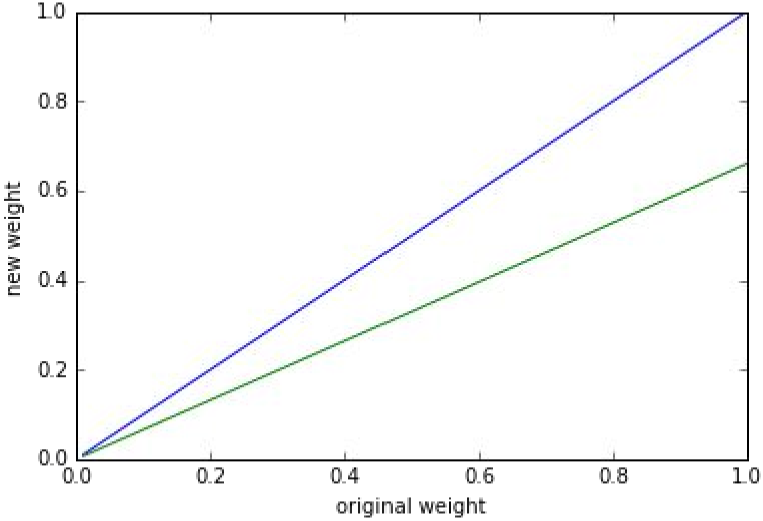
$$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{m}\beta_j^2\}$$

$$\lambda||\beta||_2^2$$



Regularization (L2, Ridge Regression)

Idea: Impose a penalty on size of weights:



Ordinary least squares objective:

$$\hat{\beta} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2\}$$

Ridge regression:

$$\hat{\beta}^{ridge} = argmin_{\beta}\{\sum_{i=1}^{N}(y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{m}\beta_j^2\}$$

$$\lambda||\beta||_2^2$$

In Matrix Form: $\mathrm{RSS}(\lambda) = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta$

$$\hat{\beta}^{ridge} = (X^TX + \lambda I)^{-1}X^Ty$$

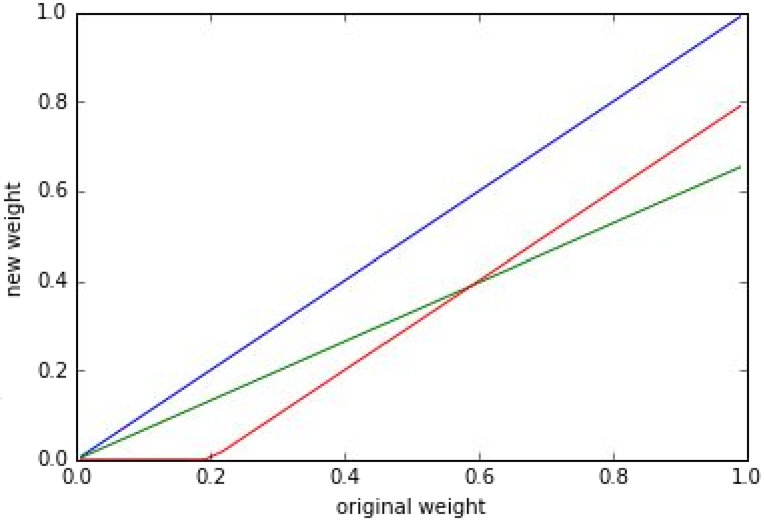
I: m x m identity matrix

Regularization (L1, The “Lasso”)

Idea: Impose a penalty and zero-out some weights

The Lasso Objective:

$$\hat{\beta}^{lasso} = argmin_{\beta}\{\frac{1}{2}\sum_{i=1}^{N}(Y_i - \sum_{j=1}^{m} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{m}|\beta_j|\}$$



$$\lambda||\beta||_1$$

No closed form matrix solution, but often solved with coordinate descent.

Application: m ≅ n or m >> n

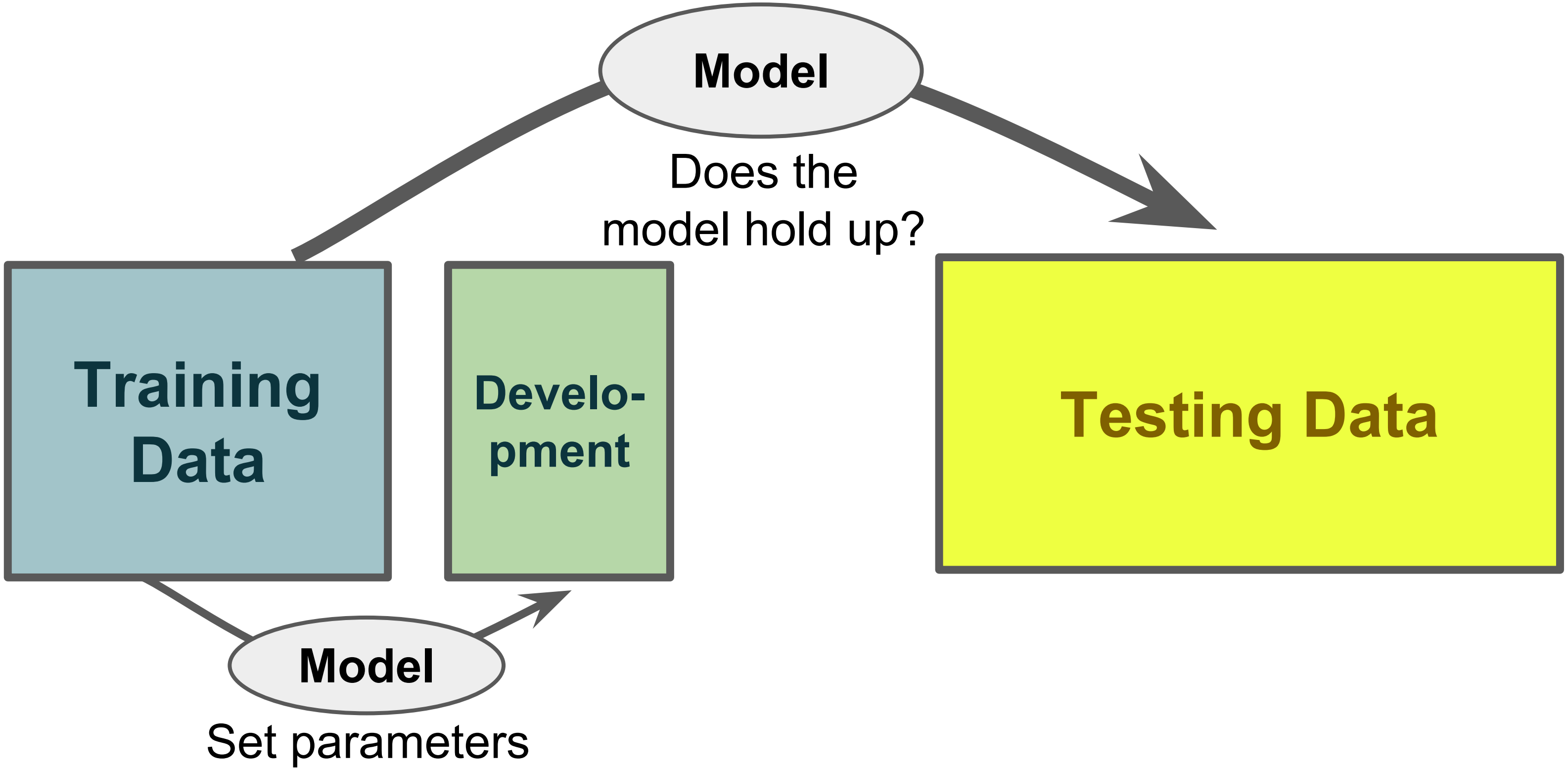
Regularization Comparison

Review, 3/31 - 4/5

- Confidence intervals
- Bootstrap

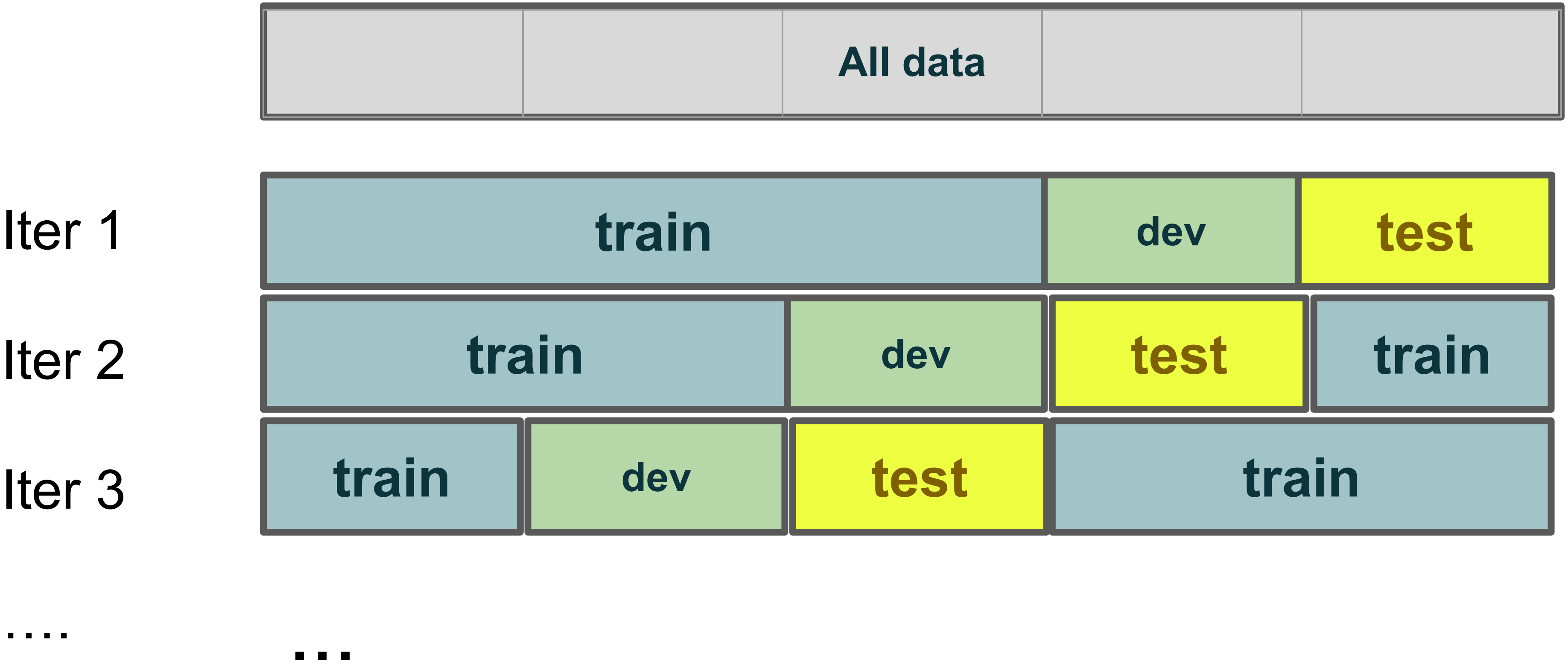
- Prediction Framework: Train, Development, Test
- Overfitting: Bias versus Variance
- Feature Selection: Forward Stepwise Regression
- Ridge Regression (L2 regularization)
- Lasso Regression (L1 regulatization)

Common Goal: Generalize to new data



N-Fold Cross-Validation

Goal: Decent estimate of model accuracy



All data

Iter 1 train dev test

Iter 2 train dev test train

Iter 3 train dev test train

....

Supervised vs. Unsupervised

Supervised

- Predicting an outcome $E(y|X)$
- Loss function used to characterize quality of prediction

$$L(y, \hat{y}) = (y - \hat{y})^2$$

Supervised vs. Unsupervised

Supervised

- Predicting an outcome $E(y|X)$
- Loss function used to characterize quality of prediction

$$L(y, \hat{y}) = (y - \hat{y})^2$$

Unsupervised

- No outcome to predict
- Goal: Infer properties of $P(X)$ without a supervised loss function.
- Often larger data.
- Don’t need to worry about conditioning on another variable.

K-Means Clustering

Clustering: Group similar observations, often over unlabeled data.

K-means: A “prototype” method
(i.e. not based on an algebraic model).

Euclidean Distance: $d(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{m}(x_{ij} - x_{i'j})^2} = ||x_i - x_{i'}||$

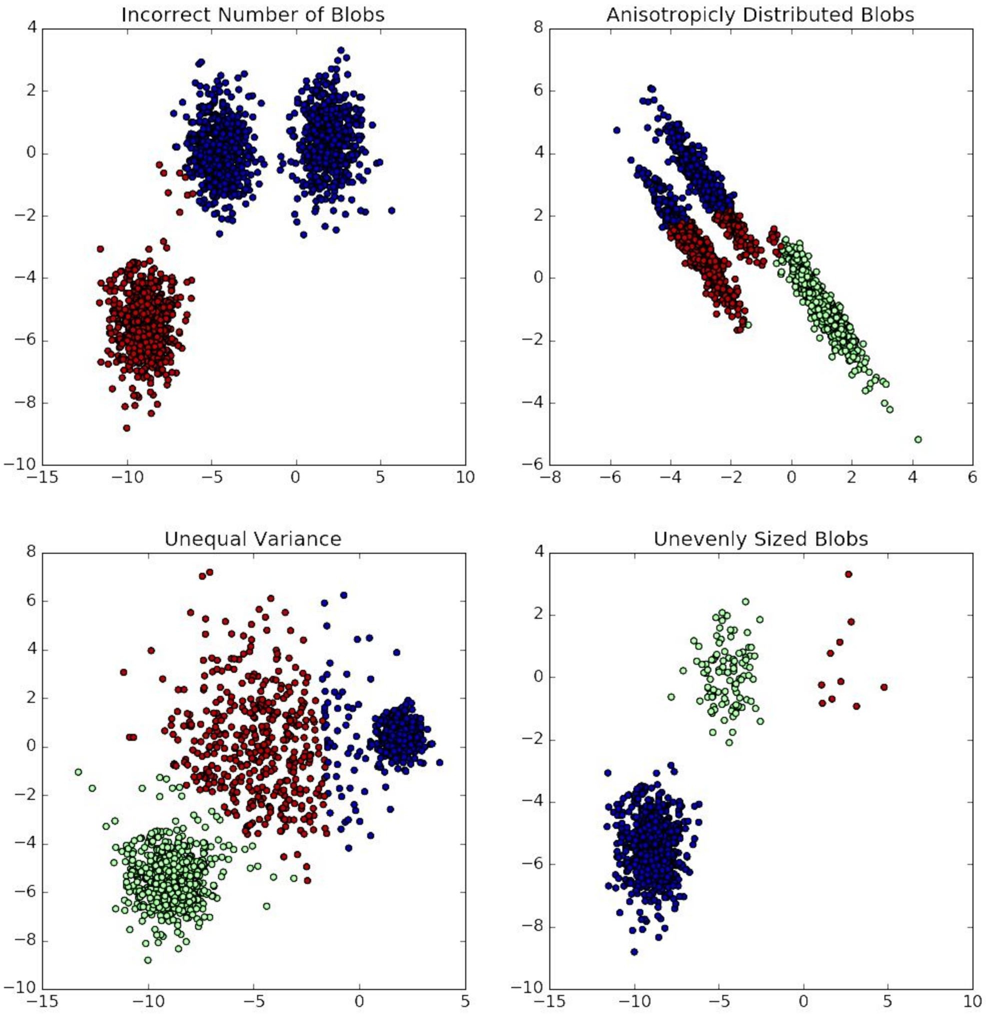
```
centers = a random selection of k cluster centers
until centers converge:
    1. For all $x_i$, find the closest center (according to $d$)
    2. Recalculate centers based on mean of euclidean distance
```

Review 4-7

- Cross-validation
- Supervised Learning
- Euclidean distance in m-dimensional space
- K-Means clustering

K-Means Clustering

Understanding K-Means



(source: Scikit-Learn)

Dimensionality Reduction - Concept

Dimensionality Reduction - PCA

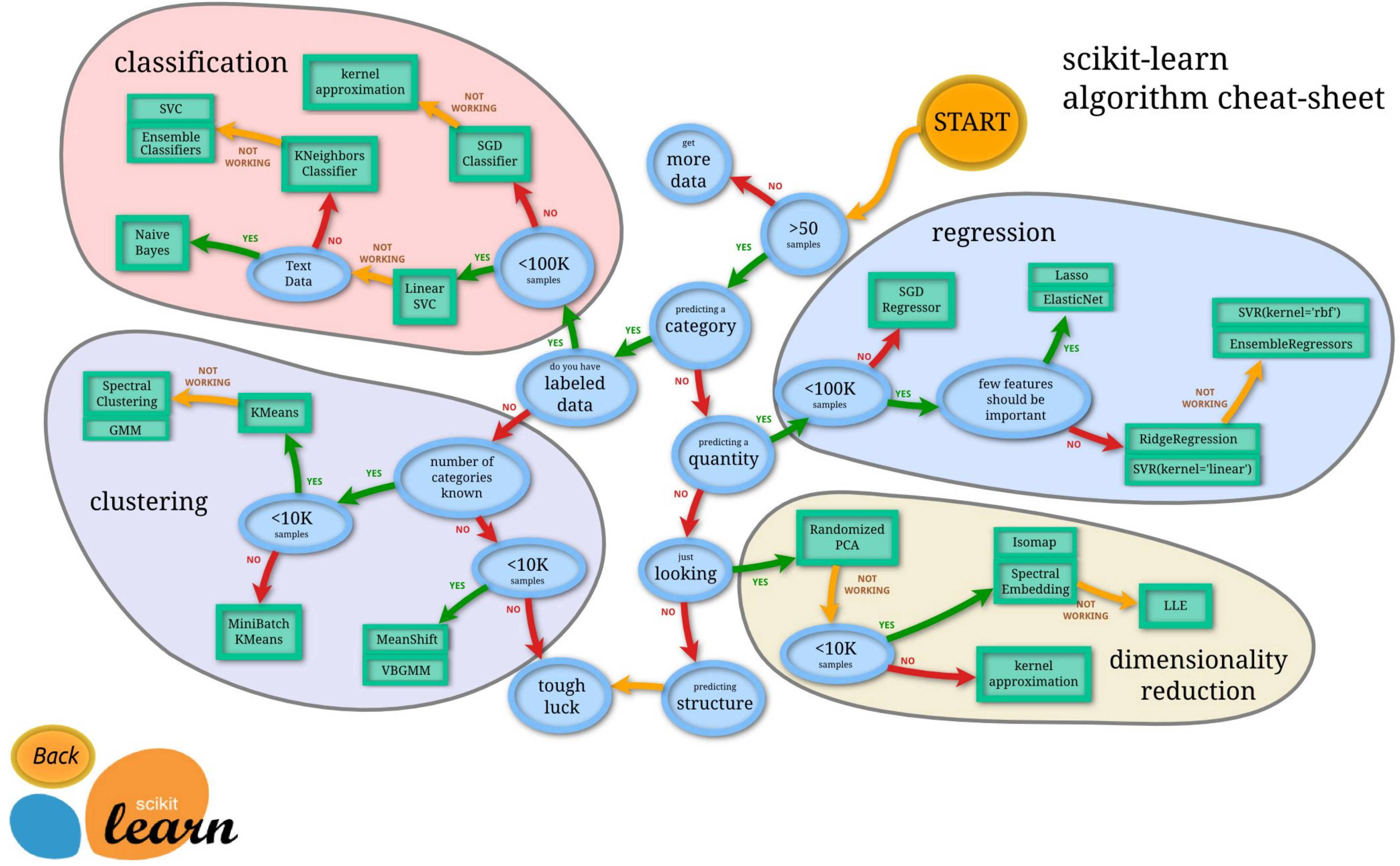
Linear approximates of data in q dimensions.

Found via *Singular Value Decomposition:*

$$X = UDV^T$$

Review 4-11

- K-Means Issues
- Dimensionality Reduction
- PCA
    - What is V (the components)?
    - Percentage variance explained


scikit-learn
algorithm cheat-sheet
START
>50 samples
NO
get more data
YES
predicting a category
YES
do you have labeled data
YES
classification
<100K samples
YES
Linear SVC
NOT WORKING
Text Data
YES
Naive Bayes
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NO
SGD Classifier
NOT WORKING
kernel approximation
NO
clustering
number of categories known
YES
<10K samples
YES
KMeans
NOT WORKING
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
tough luck
NO
predicting a quantity
YES
regression
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
NO
just looking
YES
dimensionality reduction
Randomized PCA
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
NO
predicting structure
Back
scikit learn

Classification: Regularized Logistic Regression

$\lambda||\beta||_2^2$

$\lambda||\beta||_1$

Classification: Naive Bayes

Bayes classifier: choose the class most likely according to P(y|X).
(y is a class label)

Classification: Naive Bayes

Bayes classifier: choose the class most likely according to P(y|X).
(y is a class label)

Naive Bayes classifier: Assumes all predictors are independent given y.

$$P(Y = y|A = a, B = b, C = c) = p(y|a)p(y|b)p(y|c)$$

$$\mathrm{P}(y|X) = \prod_{i=1}^{m} \mathrm{P}(y|X_i)$$

Classification: Naive Bayes

$$\mathrm{P}(y|X) = \frac{\mathrm{P}(y)\mathrm{P}(X|y)}{\mathrm{P}(X)}$$

Bayes Rule:

P(*A*|*B*) = P(*B*|*A*)P(*A*) / P(*B*)

$$\mathrm{P}(y|X) = \prod_{i=1}^{m} \mathrm{P}(y|X_i)$$

Classification: Naive Bayes

Posterior

$$\boxed{P(y|X)} = \frac{\boxed{P(y)}\boxed{P(X|y)}}{P(X)}$$

Likelihood

Prior

Classification: Naive Bayes

Posterior

$$\boxed{\mathrm{P}(y|X)} = \frac{\boxed{\mathrm{P}(y)}\boxed{\mathrm{P}(X|y)}}{\mathrm{P}(X)}$$

Likelihood

Prior

$$\mathrm{P}(y|X) \propto \mathrm{P}(y, X_1, ..., X_m) \propto \mathrm{P}(y) \prod_{i=1}^{m} \mathrm{P}(X_i|y)$$

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

$$\hat{y} = arg \max_{y} \ \mathrm{P}(y) \prod_{i=1}^{m} \mathrm{P}(X_i|y)$$

Classification: Naive Bayes

$$\underbrace{\boxed{\mathrm{P}(y|X)}}_{\textbf{Posterior}} = \frac{\underbrace{\boxed{\mathrm{P}(y)}}_{\textbf{Prior}}\overbrace{\boxed{\mathrm{P}(X|y)}}^{\textbf{Likelihood}}}{\mathrm{P}(X)}$$

$$\mathrm{P}(y|X) \propto \mathrm{P}(y, X_1, \ldots, X_m) \propto \mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)$$

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

$$\hat{y} = arg\max_{y} \overbrace{\boxed{\mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)}}^{\textbf{Unnormalized Posterior}}$$

Gaussian Naive Bayes

Assume $\mathrm{P}(\mathrm{X}|\mathrm{Y})$ is *Normal*

$$\hat{y} = arg\max_{y}\ \mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)$$

Gaussian Naive Bayes

Assume P(X|Y) is *Normal*

Then, training is:

1. Estimate P(Y = k); $\boldsymbol{\pi}_k$ = count(Y = k) / Count(Y = *)
2. MLE to find parameters (***μ***, **σ**) for each class of Y.
   (the “class conditional distribution”)

$$\hat{y} = arg\max_{y} \ \mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)$$

Gaussian Naive Bayes

Assume P(X|Y) is *Normal*

Then, training is:

1. Estimate P(Y = k); $\boldsymbol{\pi}_k$ = count(Y = k) / Count(Y = *)
2. MLE to find parameters ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$) for each class of Y.
   (the "class conditional distribution")

$$\hat{y} = arg\max_{y}\ \mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)$$

Gaussian Naive Bayes

Assume P(X|Y) is *Normal*

Then, training is:

1. Estimate P(Y = k); $\boldsymbol{\pi}_k$ = count(Y = k) / Count(Y = *)
2. MLE to find parameters (***μ***, **σ**) for each class of Y. (the “class conditional distribution”)

$$\hat{y} = arg\max_{y}\ \mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)$$

Example Project

https://docs.google.com/presentation/d/1jD-FQhOTaMh82JRc-p81TY1QCUbtpKZGwe5U4A3gml8/

Review: 4-14, 4-19

- Types of machine learning problems
- Regularized Logistic Regression
- Naive Bayes Classifier
- Implementing a Gaussian Naives Bayes

- Application of probability, statistics, and prediction for measuring county mortality rates from Twitter.

Gaussian Naive Bayes

Assume P(X|Y) is *Normal*

Then, training is:

1. Estimate $P(Y = k)$; $\boldsymbol{\pi}_k = \text{count}(Y = k) \,/\, \text{Count}(Y = *)$
2. MLE to find parameters ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$) for each class of Y. (the "class conditional distribution")

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

$$\hat{y} = arg \max_{y} \; \mathrm{P}(y) \prod_{i=1}^{m} \mathrm{P}(X_i|y)$$

Gaussian Naive Bayes

MLE: For which parameters does the observed data have the highest probability.

$$L(\theta) = \prod_{i=1}^{n} f(X_i;\theta) \qquad l(\theta) = log \sum_{i=1}^{n} f(X_i;\theta)$$

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

Unnormalized Posterior

$$\hat{y} = arg \max_{y} \boxed{\mathrm{P}(y) \prod_{i=1}^{m} \mathrm{P}(X_i|y)}$$

Gaussian Naive Bayes

Assume P(X|Y) is *Normal*

Then, training is:

1. Estimate P(Y = k); π_k = count(Y = k) / Count(Y = *)
2. MLE to find parameters (μ, σ) for each class of Y ("the class conditional distribution")

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

Without knowing P(*X*), can we turn this into the (normalized) posterior?

Unnormalized Posterior

$$\hat{y} = arg\max_{y} \boxed{\mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)}$$

Gaussian Naive Bayes

Use the Law of Total Probability, for all i = 1 ... k, where A_1 ... A_k **partition** Ω:

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

Without knowing $\mathrm{P}(X)$, can we turn this into the (normalized) posterior?

Unnormalized Posterior

$$\hat{y} = arg\max_{y} \boxed{\mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)}$$

Gaussian Naive Bayes

Use the Law of Total Probability, for all i = 1 … k, where A_1 … A_k **partition** Ω:

$$\mathrm{P}(A_i|B) = \frac{\mathrm{P}(B, A_i)}{\mathrm{P}(B)} = \frac{\mathrm{P}(B|A_i)\mathrm{P}(A_i)}{\sum_{i=1}^{k} \mathrm{P}(B|A_i)\mathrm{P}(A_i)}$$

Maximum a Posteriori (MAP): Pick the class with the maximum posterior probability.

Without knowing $\mathrm{P}(X)$, can we turn this into the (normalized) posterior?

Unnormalized Posterior

$$\hat{y} = arg\max_{y} \boxed{\mathrm{P}(y)\prod_{i=1}^{m}\mathrm{P}(X_i|y)}$$

Gaussian Naive Bayesian Inference

Use the Law of Total Probability, for all i = 1 ... k, where *A*$_1$ ... *A*$_k$ **partition** Ω:

$$\mathrm{P}(A_i|B) = \frac{\mathrm{P}(B, A_i)}{\mathrm{P}(B)} = \frac{\mathrm{P}(B|A_i)\mathrm{P}(A_i)}{\sum_{i=1}^{k} \mathrm{P}(B|A_i)\mathrm{P}(A_i)}$$

discrete

$$\mathrm{P}(A|B) = \frac{\mathrm{P}(B|A)\mathrm{P}(A_i)}{\int \mathrm{P}(B|A)\mathrm{P}(A)dA}$$

continuous

A is "marginalized" out

Without knowing $\mathrm{P}(X)$, can we turn this into the (normalized) posterior?

Unnormalized Posterior

$$\hat{y} = arg\,\max_{y} \boxed{\mathrm{P}(y) \prod_{i=1}^{m} \mathrm{P}(X_i|y)}$$

Gaussian Naive Bayesian Inference

Q: What distinguishes Bayesian inference? **A**: Assume a

$\mathrm{P}(\theta)$ – prior

Bayesian Inference

$$Z = X_{training}$$

Given:

$\mathrm{P}(Z|\theta)$ – probability density or mass function (likelihood)

$\mathrm{P}(\theta)$ – prior

Goal: Compute the $\text{posterior} = \dfrac{\text{(prior)(likelihood)}}{\text{evidence}} = \dfrac{P(\theta)P(Z|\theta)}{P(Z)}$

Bayesian Inference

$$Z = X_{training}$$

Given:

$\mathrm{P}(Z|\theta)$ – probability density or mass function (likelihood)

$\mathrm{P}(\theta)$ – prior

Goal: Compute the $\text{posterior} = \dfrac{\text{(prior)(likelihood)}}{\text{evidence}} = \dfrac{P(\theta)P(Z|\theta)}{P(Z)}$

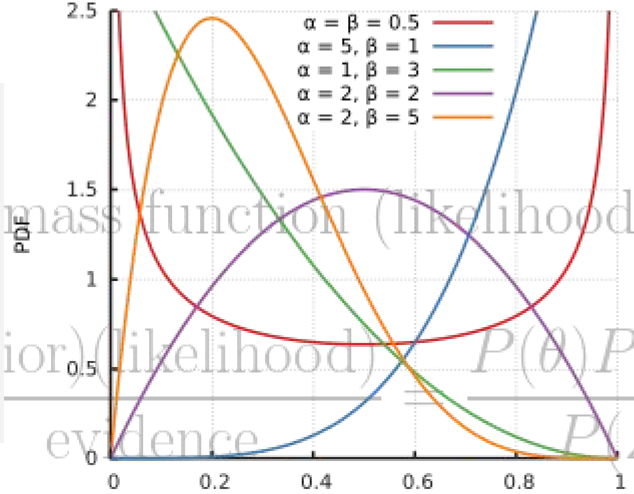
Types of priors:

- Uninformative (Improper: not a probability (e.g. constant))
- Belief-based
- **Conjugate** to a likelihood: if the posterior is in the same family as the prior.

Bayesian Inference

Example: Beta(α, β) is conjugate to a Bernoulli likelihood.

https://en.wikipedia.org/wiki/Conjugate_prior#Table_of_conjugate_distributions



mass function (likelihood)

$$\frac{\text{ior)(likelihood)}}{\text{evidence}} = \frac{P(\theta)P(Z|\theta)}{P(Z)}$$

Types of priors:

- Uninformative (Improper: not a probability (e.g. constant))
- Belief-based

- **Conjugate** to a likelihood: if the posterior is in the same family as the prior.

Bayesian Inference

$$Z = X_{training}$$

Given:

$\mathrm{P}(Z|\theta)$ – probability density or mass function (likelihood)

$\mathrm{P}(\theta)$ – prior

Goal: Compute the $\text{posterior} = \dfrac{(\text{prior})(\text{likelihood})}{\text{evidence}} = \dfrac{P(\theta)P(Z|\theta)}{P(Z)}$

Bayesian Inference

$$Z = X_{training}$$

Given:

$\mathrm{P}(Z|\theta)$ – probability density or mass function (likelihood)
$\mathrm{P}(\theta)$ – prior

Goal: Compute the $\text{posterior} = \dfrac{\text{(prior)(likelihood)}}{\text{evidence}} = \dfrac{P(\theta)P(Z|\theta)}{P(Z)}$

$$\boxed{\mathrm{P}(\theta|Z) = \frac{\mathrm{P}(\theta)\mathrm{P}(Z|\theta)}{\int \mathrm{P}(\theta)\mathrm{P}(Z|\theta)d\theta}}$$

Bayesian Inference

$$Z = X_{training}$$

Given:

$\mathrm{P}(Z|\theta)$ – probability density or mass function (likelihood)

$\mathrm{P}(\theta)$ – prior

Goal: Compute the $\text{posterior} = \dfrac{\text{(prior)(likelihood)}}{\text{evidence}} = \dfrac{P(\theta)P(Z|\theta)}{P(Z)}$

$$\boxed{\mathrm{P}(\theta|Z) = \frac{\mathrm{P}(\theta)\mathrm{P}(Z|\theta)}{\int \mathrm{P}(\theta)\mathrm{P}(Z|\theta)d\theta}}$$

$\mathrm{P}(z^{new}|Z) = \int \mathrm{P}(z^{new}|\theta)\mathrm{P}(\theta|Z)d\theta$ -- predictive distribution

Bayesian Inference

$$Z = X_{training}$$

Given:

$\mathrm{P}(Z|\theta)$ – probability density or mass function (likelihood)

$\mathrm{P}(\theta)$ – prior

Goal: Compute the $\text{posterior} = \dfrac{\text{(prior)(likelihood)}}{\text{evidence}} = \dfrac{P(\theta)P(Z|\theta)}{P(Z)}$

$$\boxed{\mathrm{P}(\theta|Z) = \frac{\mathrm{P}(\theta)\mathrm{P}(Z|\theta)}{\int \mathrm{P}(\theta)\mathrm{P}(Z|\theta)d\theta}}$$

$$\boxed{\mathrm{P}(z^{new}|Z) = \overbrace{\int \mathrm{P}(z^{new}|\theta)\mathrm{P}(\theta|Z)d\theta}^{\textbf{Like a posterior-weighted average of } \mathbf{P(Z^{new}|\theta)}} \text{ -- predictive distribution}}$$

Review, 4-21

- How to turn an unnormalized posterior into a normalized posterior
- What is Bayesian Inference?
- Typical definition of a posterior
- Predictive Distribution

Bayesian Vs. Frequentist

Frequentist

- Limiting relative frequencies => probability is an observed property
- Parameters fixed and unknown => no need for probability of parameter
- Procedures for long-run frequencies (e.g. 95% CI)

Bayesian Vs. Frequentist

Bayesian

- Probability is degree of belief
  => can derive probability of many things
- Can estimate probability of parameters
- Can draw inferences about parameter
  probability distribution, point estimates, intervals

Frequentist

- Limiting relative frequencies => probability is an observed property
- Parameters fixed and unknown => no need for probability of parameter
- Procedures for long-run frequencies (e.g. 95% CI)

Bayesian Vs. Frequentist

Pro Bayes:

- Estimating distributions => uncertainty built in
- No need to choose model; always “admissible”
- Automatic regularization

Con:

- Need to assume prior (even if nothing can obviously work)
- Approximate solutions: tend to be a little less accurate for simple classification / regression problems

Bayesian Vs. Frequentist

Pro Bayes:

- Estimating distributions => uncertainty built in
- No need to choose model; always “admissible”
- Automatic regularization

There is at least one situation where the model performs at least as good as any other model.

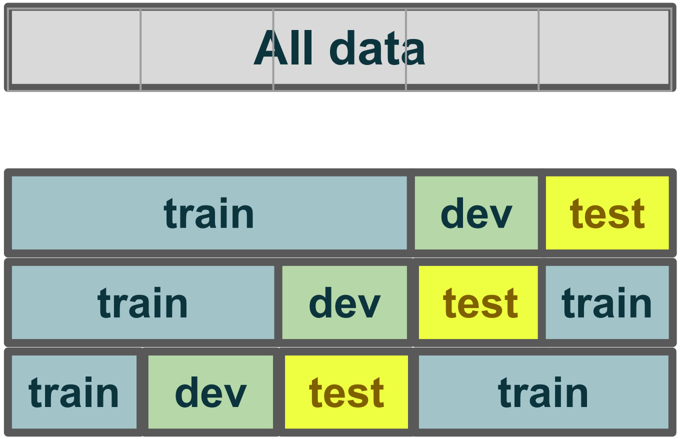
Con:

- Need to assume prior (even if nothing can obviously work)
- Approximate solutions: tend to be a little less accurate for simple classification / regression problems

Revisiting N-Fold Cross-Validation

Goal:

Decent estimate of model accuracy



All data

train | dev | test

train | dev | test | train

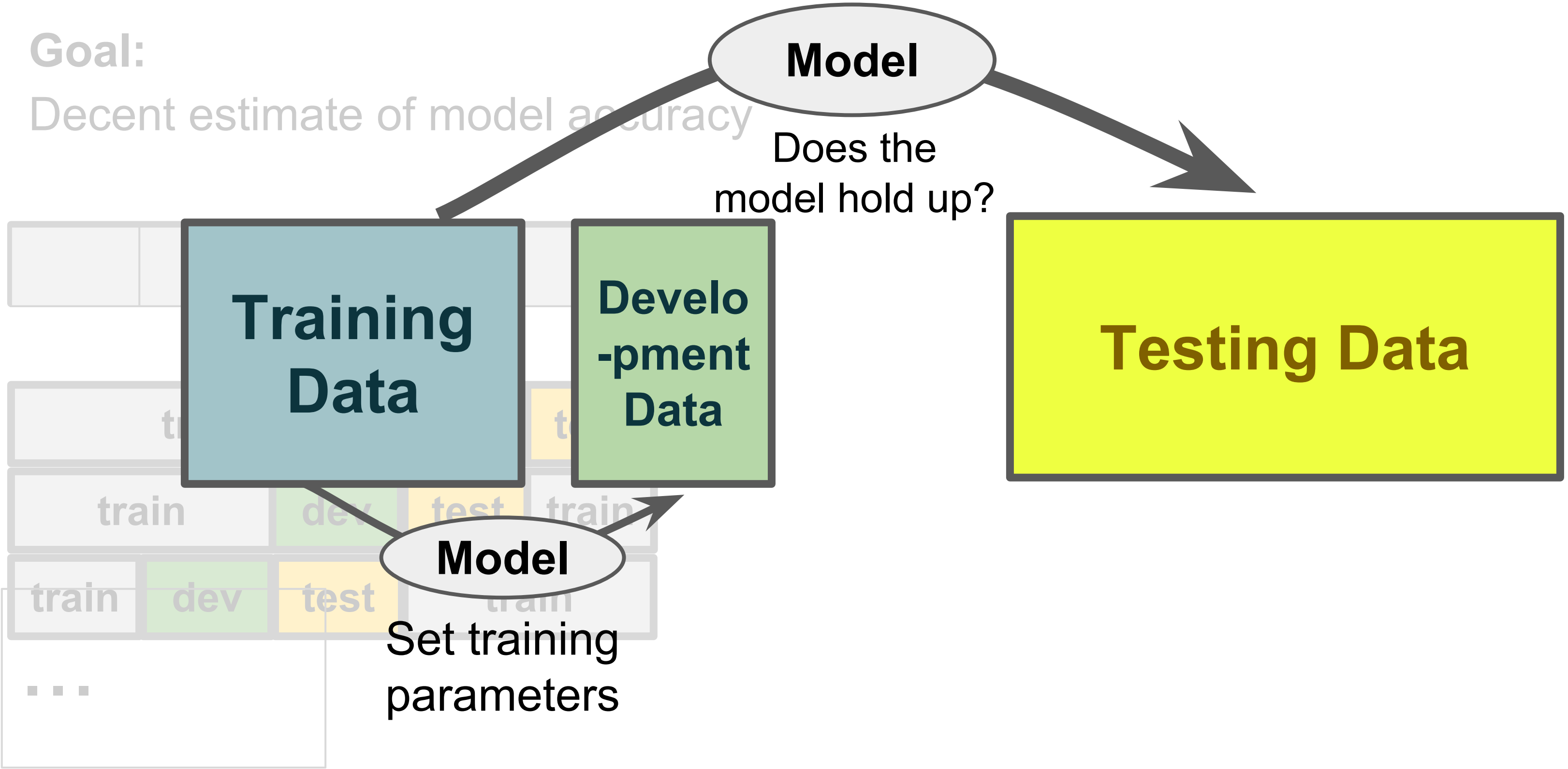
train | dev | test | train

...

Revisiting N-Fold Cross-Validation

Goal:

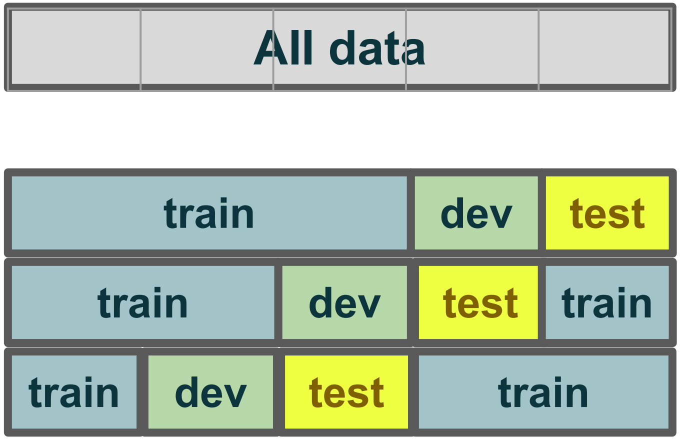
Decent estimate of model accuracy



Revisiting N-Fold Cross-Validation

Goal:

Decent estimate of model accuracy



All data

train | dev | test

train | dev | test | train

train | dev | test | train

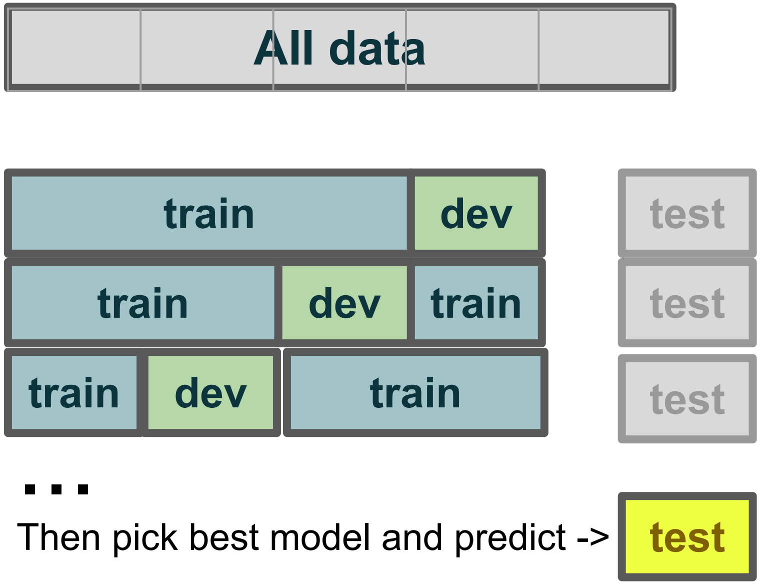
…

Revisiting N-Fold Cross-Validation

Goal:

Select a super-reliable penalty (alpha)

(this is overkill)

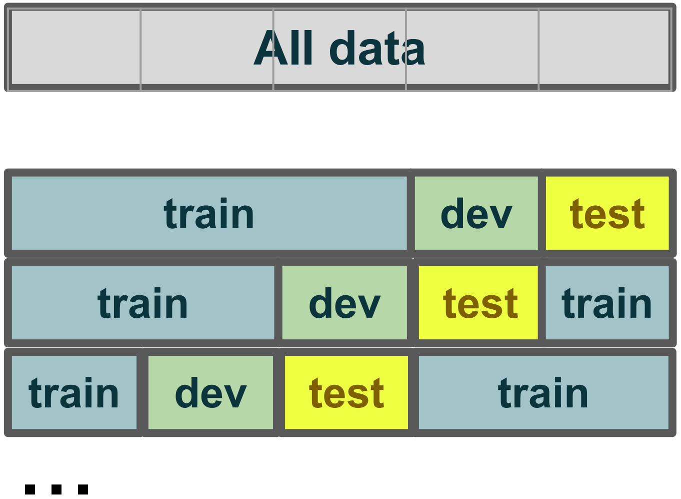


Then pick best model and predict ->

Revisiting N-Fold Cross-Validation

Goal:

Decent estimate of model accuracy



All data

train | dev | test

train | dev | test | train

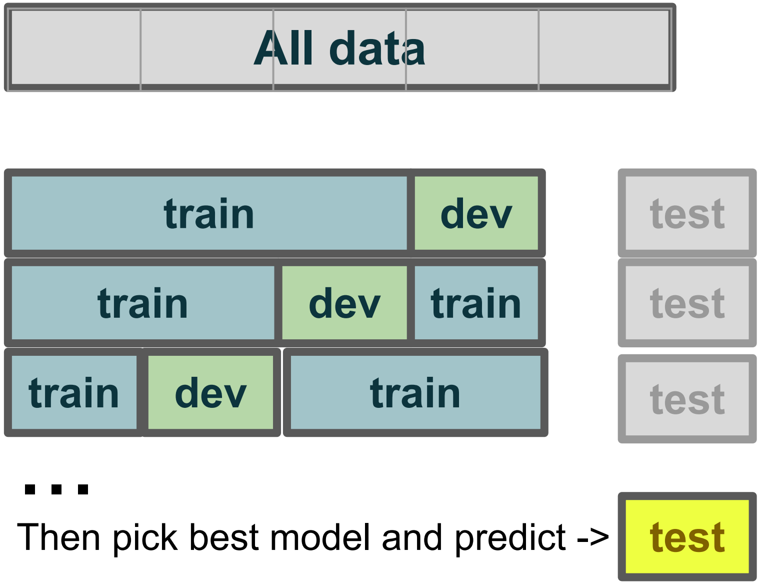
train | dev | test | train

...

≠

Goal:

Select a super-reliable penalty (alpha)
(this is overkill)



All data

train | dev | test

train | dev | train | test

train | dev | train | test

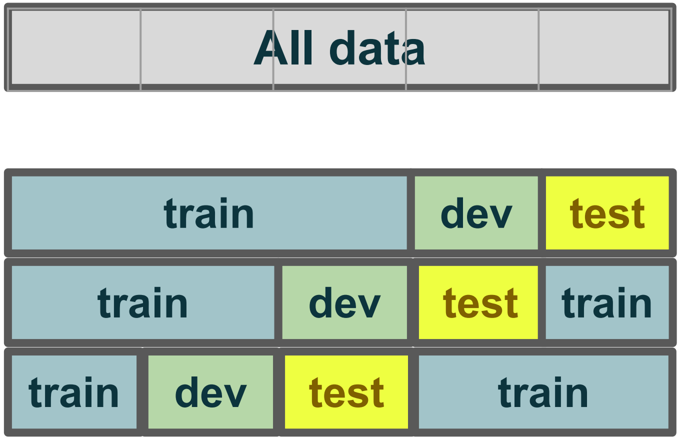
...

Then pick best model and predict -> test

Revisiting N-Fold Cross-Validation

Goal:

Decent estimate of model accuracy



All data

train | dev | test

train | dev | test | train

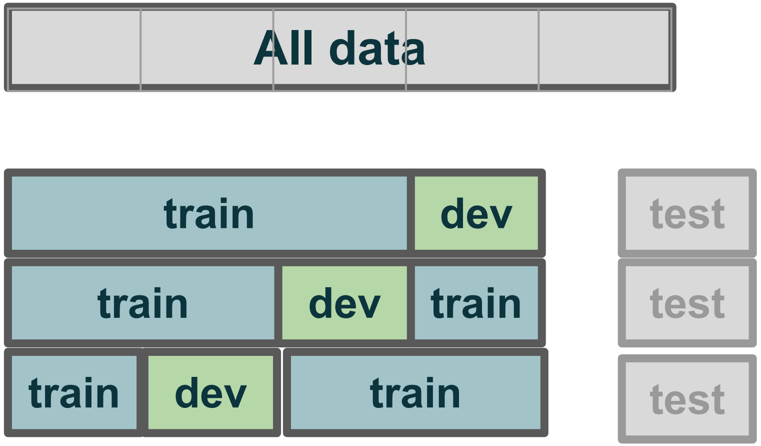
train | dev | test | train

...

Example: Assignment 3

Goal:

Select a super-reliable penalty (alpha)
(this is overkill)



All data

train | dev

train | dev | train

train | dev | train

test

test

test

...

Then pick best model and predict -> test

Introduction Time Series Analysis

Goal: Understanding temporal patterns of data (or real world events)

Common tasks:

- **Trend Analysis:** Extrapolate patterns over time (typically descriptive).
- **Forecasting:** Predicting a future event (predictive).
  (contrasts with "cross-sectional" prediction -- predicting a different group)

Introduction to Causal Inference (Revisited)

X causes Y   as opposed to   X is associated with Y

Changing X will change the distribution of Y.

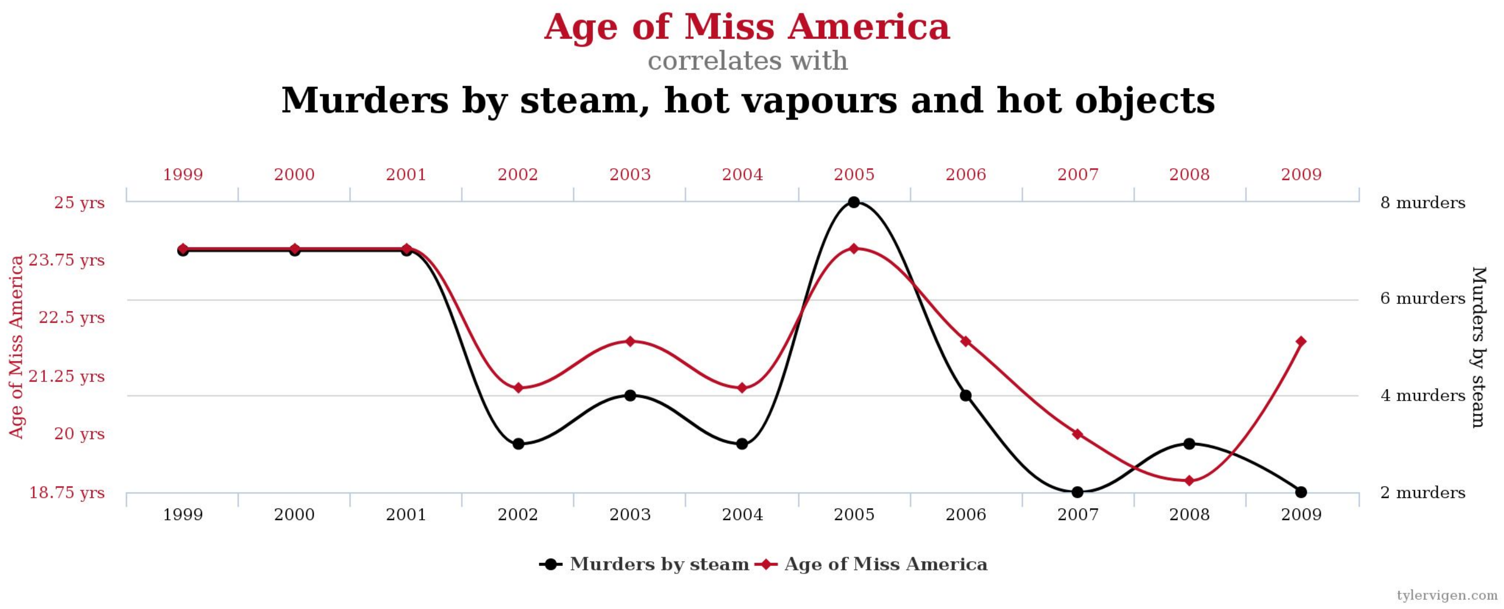
X causes Y $\nLeftrightarrow$ Y causes X

Spurious Correlations

Extremely common in time-series analysis.

Spurious Correlations

Extremely common in time-series analysis.



Introduction to Causal Inference (Revisited)

X causes Y as opposed to X is associated with Y

Changing X will change the distribution of Y.

X causes Y $\not\leftrightarrow$ Y causes X

$$P(Y=1|X=1) - P(Y=1|X=0)$$

Counterfactual Model: Exposed or Not Exposed: X = 1 or 0

$$Y = \begin{cases} C_0 & \text{if } X = 0 \\ C_1 & \text{if } X = 1 \end{cases}$$

Causal Odds Ratio:

$$\frac{\left(\frac{P(C_1=1)}{P(C_1=0)}\right)}{\left(\frac{P(C_0=1)}{P(C_0=0)}\right)}$$

Simpson’s “Paradox”

	Y=1	Y=0	Y=1	Y=0
X=1	.15	.225	.1	.025
X=0	.0375	.0875	.2625	.1125
	Z = men		Z = women	

http://vudlab.com/simpsons/

Autocorrelation

“(a.k.a. Serial correlation).”

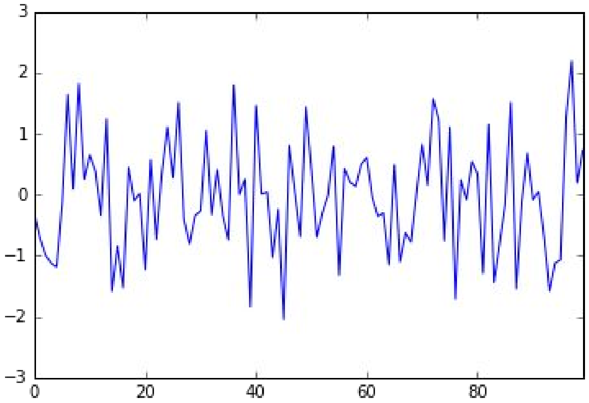
Quantifying the strength of a temporal pattern in serial data.

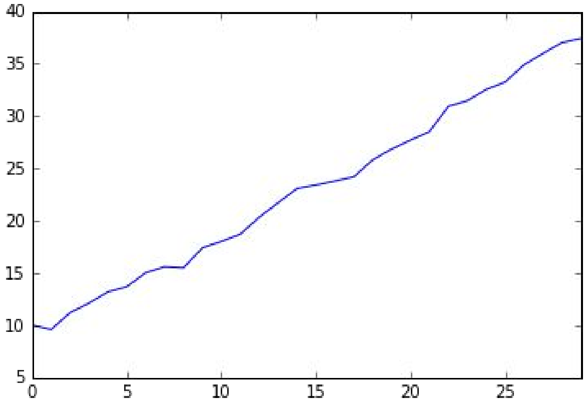
Requirements:

- Assume regular measurement (hourly, daily, monthly...etc..)

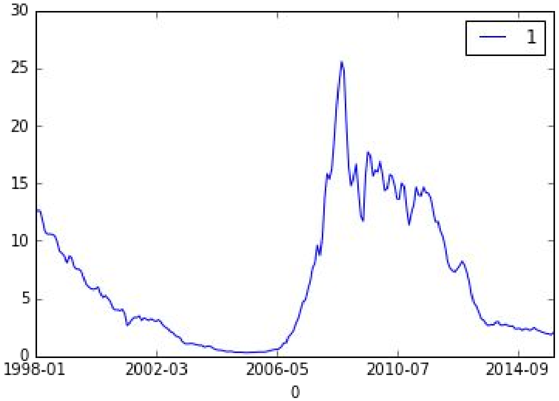
Autocorrelation

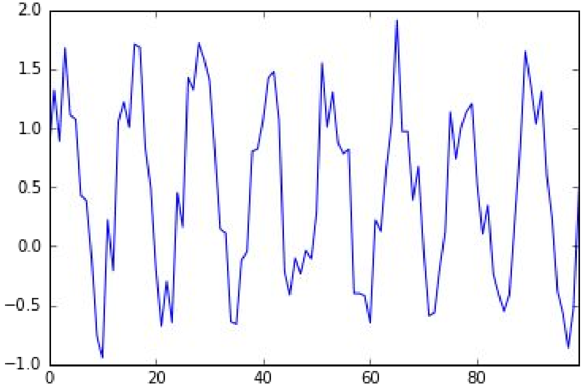
Quantifying the strength of a **temporal pattern** in serial data.





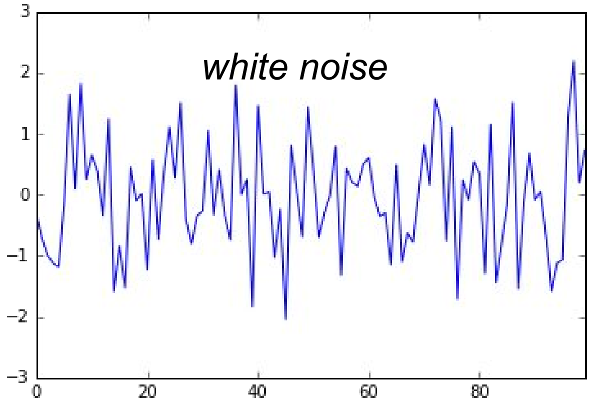
Which have temporal patterns?

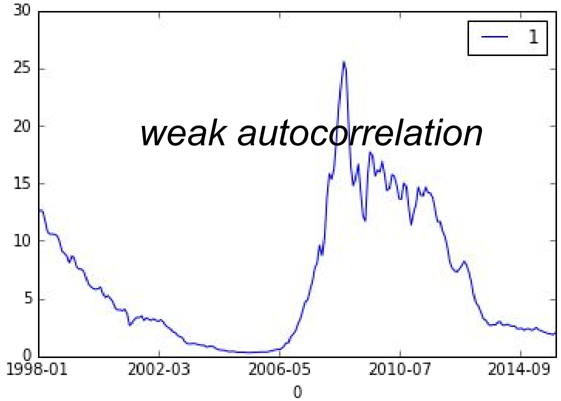




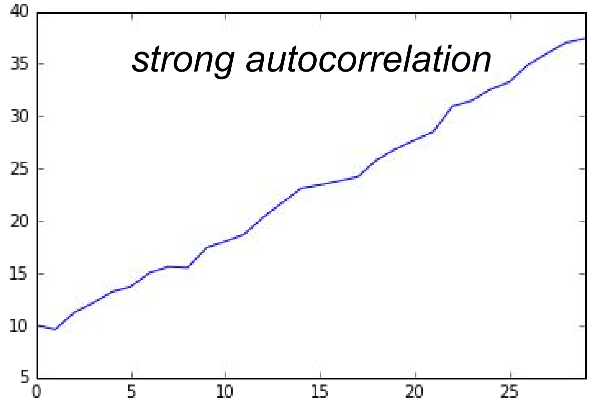
Autocorrelation

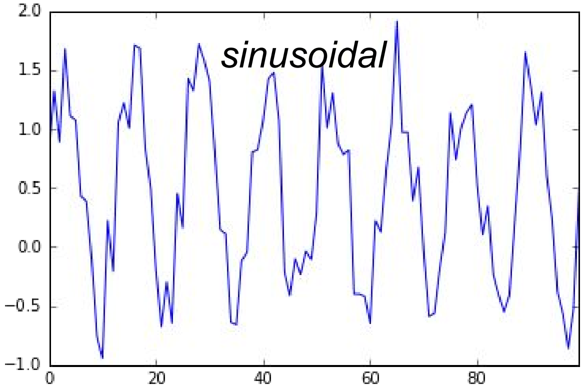
Quantifying the strength of a **temporal pattern** in serial data.


white noise


weak autocorrelation

Which have temporal patterns?

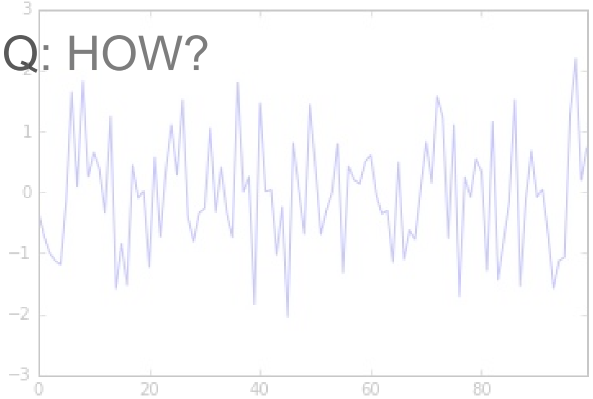

strong autocorrelation

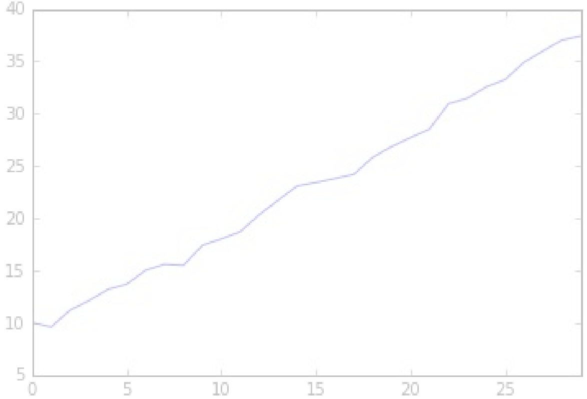

sinusoidal

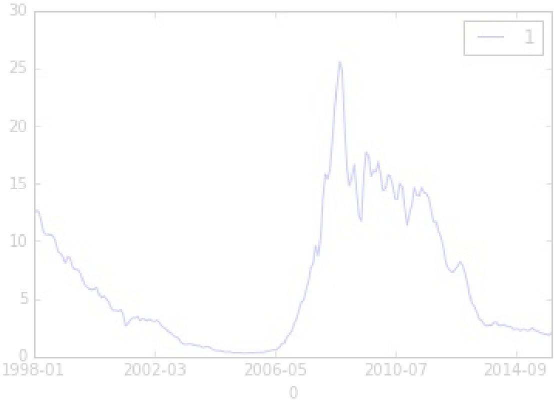
Autocorrelation

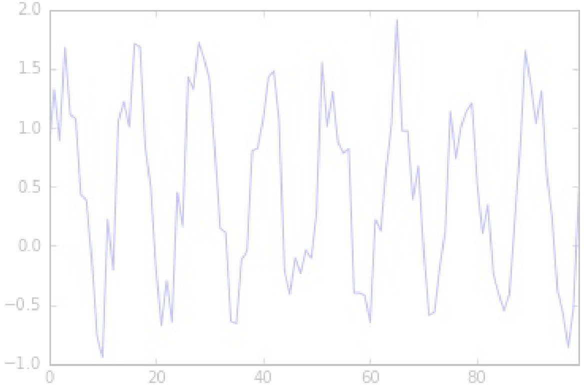
Quantifying the strength of a **temporal pattern** in serial data.

Q: HOW?



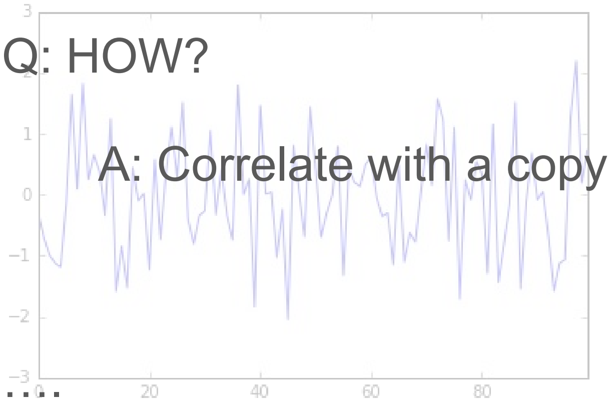






Autocorrelation

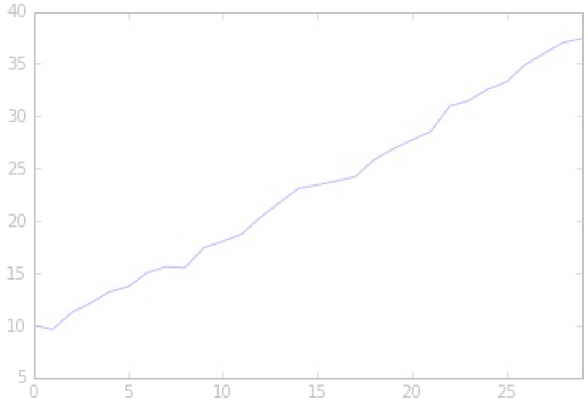
Quantifying the strength of a **temporal pattern** in serial data.

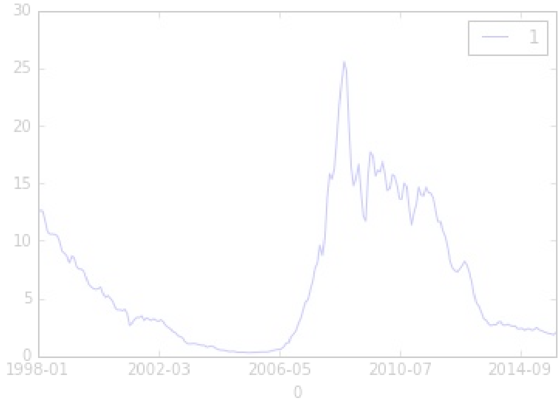


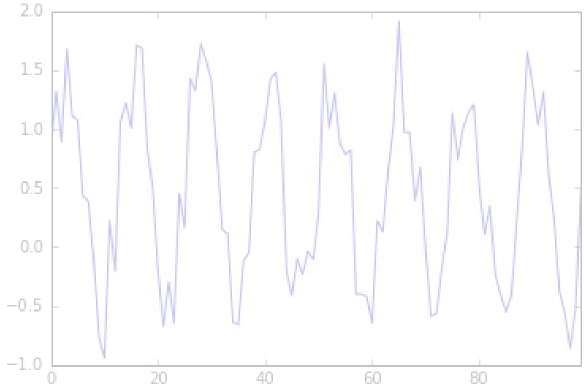
Q: HOW?

A: Correlate with a copy of self, shifted slightly.

....







Autocorrelation

Quantifying the strength of a **temporal pattern** in serial data.

Q: HOW?

A: Correlate with a copy of self, shifted slightly.

```
Y = [3, 4, 4, 5, 6, 7, 7, 8]

correlate(Y[0:7], Y[1:8])  #lag=1

correlate(Y[0:-2], Y[2:8])  #lag=2

....
```

Autocorrelation

Quantifying the strength of a **temporal pattern** in serial data.

Q: HOW?

A: Correlate with a copy of self, shifted slightly.

```
Y = [3, 4, 4, 5, 6, 7, 7, 8]

correlate(Y[0:7], Y[1:8])  #lag=1

correlate(Y[0:-2], Y[2:8])  #lag=2

....
```

Review, 4-26 and 4-28

- Bayesian verse Frequentist Learning
- Why / when to use Dev within folds of N-Fold CV
- Time series -- what distinguishes
- Causal Inference
- Autocorrelation
  - Type of univariate time series
  - Lag Plots

Autoregressive Model

AR Models: $Y_t = f(Y_{t-1}, Y_{t-2}, Y_{t-3}, ..., Y_{t-n}, \epsilon_t)$

Linear AR model: $Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_n Y_{t-p} + \epsilon_t$

Autoregressive Model

AR Models: $Y_t = f(Y_{t-1}, Y_{t-2}, Y_{t-3}, ..., Y_{t-n}, \epsilon_t)$

Linear AR model: $Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_n Y_{t-p} + \epsilon_t$

Notation:

$$\text{AR}(1)\text{: } \hat{Y}_t = \beta_0 + \beta_1 Y_{t-1}$$
$$\text{AR}(2)\text{: } \hat{Y}_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2}$$
$$\text{AR}(3)\text{: } \hat{Y}_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \beta_3 Y_{t-3}$$

Autoregressive Model

AR Models: $Y_t = f(Y_{t-1}, Y_{t-2}, Y_{t-3}, \ldots, Y_{t-n}, \epsilon_t)$

Linear AR model: $Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_n Y_{t-p} + \epsilon_t$

Notation:

$$\text{AR}(1)\text{: } \hat{Y}_t = \beta_0 + \beta_1 Y_{t-1}$$
$$\text{AR}(2)\text{: } \hat{Y}_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2}$$
$$\text{AR}(3)\text{: } \hat{Y}_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \beta_3 Y_{t-3}$$

$$\text{AR}(0)\text{: } \hat{Y}_t = \beta_0$$

Moving Average

Based on error;   (a “smoothing” technique).

Q: Best estimator of random data (i.e. white noise)?

Moving Average

Based on error;   (a “smoothing” technique).

Q: Best estimator of random data (i.e. white noise)?

A: The mean

$$\hat{Y}_t^{MA} = \frac{Y_t + Y_{t-1} + Y_{t-2} + ... + Y_{t-p}}{p+1}$$

Moving Average

Based on error;   (a “smoothing” technique).

Q: Best estimator of random data (i.e. white noise)?

A: The mean

$$\hat{Y}_t^{MA} = \frac{Y_t + Y_{t-1} + Y_{t-2} + \ldots + Y_{t-p}}{p+1}$$

Simple Moving Average

Moving Average Model

In a regression model (AR**MA** or ARI**MA**), we consider error terms

$$Y_t = f(\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \epsilon_{t-3}, \ldots)$$

Moving Average Model

In a regression model (AR**MA** or ARI**MA**), we consider error terms

$$Y_t = f(\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \epsilon_{t-3}, \ldots)$$

$$\hat{Y}_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_p \epsilon_{t-p}$$

Moving Average Model

In a regression model (AR**MA** or ARI**MA**), we consider error terms

$$Y_t = f(\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \epsilon_{t-3}, \ldots)$$

$$\hat{Y}_t = \mu + \epsilon_t + \theta_1 \boxed{\epsilon_{t-1}} + \theta_2 \boxed{\epsilon_{t-2}} + \ldots + \theta_p \boxed{\epsilon_{t-p}}$$

attributed to “shocks” -- independent, from a normal distribution

Notation:

$$\text{MA}(1)\text{:}\ \hat{Y}_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1}$$

$$\text{MA}(2)\text{:}\ \hat{Y}_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

ARMA Models

AutoRegressive (AR) Moving Average (MA) Model

ARMA(p, q):

$$\hat{Y}_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \\ \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$$

ARMA(1, 1):

$$\hat{Y}_t = \beta_1 Y_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1}$$

example: Y is sales; error may be effect from coupon or advertising

(credit: Ben Lambert)

Time-series Applications

- ARMA
  - Economic indicators
  - System performance
  - Trend analysis

    (often situations where there is a general trend and random “shocks”)
- Univariate Models in General
  - Anomaly Detection
  - Forecasting
  - Season Trends
  - Signal Processing
- Integration as predictors within multivariate models

```
statsmodels.tsa.arima_model
```

Review: 5-3

- Autoregression Model
- Notation
- Simple Moving Average
- Moving Average Model
- ARMA
- Applications of Time Series Models