APMA E4990.02: Introduction to Supervised Learning

Lecture 2

Homework 0 corrections

- Problem 1 - clarified what ‘inner product’ meant and clarified the product rule question.

- Problem 3 - Parts f/g contained an error. For part f), compute the variance in terms of N and p (not p hat). For part g), I give you an explicit p hat now in order to compute the confidence interval.

- Data set for final project: **Use NYC taxi data.** I recommend checking out *Big Query* (free membership to start):

- **BigQuery:** https://console.cloud.google.com/marketplace/details/city-of-new-york/nyc-tlc-trips?filter=solution-type:dataset&filter=category:encyclopedic

- **NYC Taxi Data:** https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- **PythonAnywhere for hosting :** https://www.pythonanywhere.com
- **Homework 1 Posted - Due 2 weeks from today.** Homework 0 due next week.

Clarifications from last time

$$Z_n := \frac{Y - np}{\sqrt{p(1-p)n}}$$

This is a precise statement of the central limit theorem.

$$\lim_{n \to +\infty} P(a \leq Z_n \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-z^2/2} dz$$

We now look at our observation:

$$\hat{Z}_n := \frac{\hat{Y} - np}{\sqrt{p(1-p)n}} = \frac{15 - 500(0.02)}{\sqrt{0.02(1-0.02)500}} = 1.44$$

Clarifications from last time

$$Z_n := \frac{Y - np}{\sqrt{p(1-p)n}} \qquad \lim_{n \to +\infty} P(a \leq Z_n \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-z^2/2} dz$$

We now look at our observation:

$$\hat{Z}_n := \frac{\hat{Y} - np}{\sqrt{p(1-p)n}} = \frac{15 - 500(0.02)}{\sqrt{0.02(1-0.02)500}} = 1.44$$

$$P(Y \geq 15) = P(Z_n \geq 1.44) \to \frac{1}{\sqrt{2\pi}} \int_{1.44}^{+\infty} e^{-z^2} dz$$

$$= 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{1.44} e^{-z^2} dz = 1 - \Phi(1.44)$$

Unix Importance

- All production level data science code is run on Linux/Unix servers. The scripts used to pull in data, merge it and run models is all run via scripts on these machines (eg. EC2 instances on AWS).

- Commonly need to move, edit files on remote servers, scp various pieces of code/data, start AWS clusters, etc. All of this requires basic knowledge of the command line.

- My opinion is that learning without doing is pointless, so we will learn Linux by example when you set up your own web server.

Unix Knowledge

File manipulation first
- mkdir, ls, cd, touch, mv (-r), cp (-r)

Making life on the command line easier
- emacs shortcuts on the command line
- up arrow, CTRL-R history search, CTRL-E, CTRL-U

A terminal based editor
- pico, nano, vim or emacs.

Text manipulation and display
- cat head tail grep (-r)

Network
- curl, wget

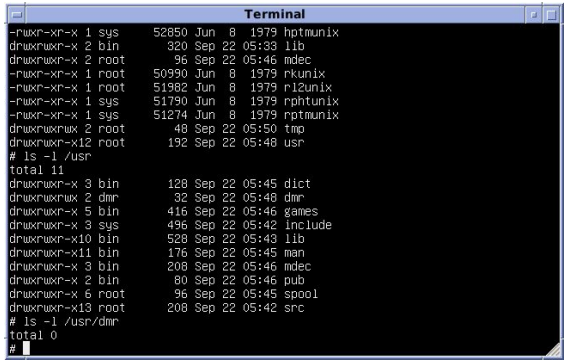
STDIN/OUT
- redirection (>1 >2),

Process chaining and standard in/out
- pipe (|), redirecting

Process manipulation
- bg, CTRL-Z, CTRL-C
- environmental variables
- ps (list of processes)
- what is TOP
- lsof (what files are open by what processes)

Common tasks:
- operating on a bunch of files using xargs or a loop
- shell scripting
- .bashrc file
- working on remote machines with sshd
- awk/sed for file processing.

http://ryanstutorials.net/linuxtutorial/cheatsheet.php



Basic Navigation

pwd
Where am I in the system.

ls [path]
Perform a listing of the given path or your current directory.

Common options: -l, -h, -a

cd [path]
Change into the given path or into your home directory.

Path
A description of where a file or directory is on the filesystem.

Absolute Path
One beginning from the root of the file system (eg. /etc/sysconfig).

Relative Path
One relative to where you currently are in the system (eg. Documents/music).

~ (tilde)
Used in paths as a reference to your home directory (eg. ~/Documents).

. (dot)

Used in paths as a reference to your current directory (eg. ./bin).

.. (dot dot)
Used in paths as a reference to your current directories parent directory (eg. ../bin).

TAB completion
Start typing and press TAB. The system will auto complete the path. Press TAB twice and it will show you your alternatives.

More About Files

file [path]
Find out what type of item a file or directory is.

Spaces in names
Put whole path in quotes (") or a backslash (\ ) in front of spaces.

Hidden files and directories
A name beginning with a . (dot) is considered hidden.

Permissions

r (read) w (write) x (execute)

Owner or User, Group and Others

ls -l [path]
View the permissions of a file or all items in a directory.

chmod <permissions> <path>
Change permissions. Permissions can be either shorthand (eg. 754) or longhand (eg. g+x).

Manual Pages

man <command>
View the man page for a command.

man -k <search term>
Search for man pages containing the search term.

Press q to exit man pages

Vi / Vim

View our Vim Cheat sheet

Filters

head
Show the first n lines.

tail
Show the last n lines.

sort
Sort lines in a given way.

wc
How many words, characters and lines.

grep
Search for a given pattern.

See more on our Grep Cheat sheet

More filters can be found here.

File Manipulation

mkdir <directory name>
Create a directory

rmdir <directory name>
Remove a directory (only if empty).

touch <file name>
Create a blank file.

cp <source> <destination>
Copy the source file to the destination.

mv <source> <destination>
Move the source file to the destination.

May also be used to rename files or directories.

rm <path>
Remove a file or directory.

Common options: -r -f

Wildcards

May be used anywhere in any path.

*****
Zero or more characters (eg. b*).

?
Single character (eg. file.???).

[ ]
Range (eg. b[aio]t).

Wildcards

May be used anywhere in any path.

*****
Zero or more characters (eg. b*).

?
Single character (eg. file.???).

[ ]
Range (eg. b[aio]t).

Piping and Redirection

>
Redirect STDOUT to a file.

>>
Append STDOUT to the end of a file.

2>
Redirect the STDERR to a file.

<
Pass the contents of a file to a program as STDIN.

|
Feed the STDOUT of the program on the left as STDIN to the program on the right.

Process Management

CTRL + C
Cancel the currently running process.

kill <process id>
Cancel the given process.

Include the option -9 to kill a stubborn process.

ps
Obtain a listing of processes and their id's.

Including the option aux will show all processes.

CTRL + Z
Pause the currently running process and put it in the background.

jobs
See a list of current processes in the background.

fg <job number>
Move the given process from the background to the foreground.

Review from Lecture 1

- Introduced examples of machine learning applications and methods.
  - Used examples from Amazon, Netflix, The New York Times, Booking.com and Tinder, Lyft/Uber.
  - Discussed **Supervised, Unsupervised** and **Reinforcement** methods of machine learning.
  - Supervised Learning - Linear Regression, Decision Trees, Graphical models.
- How do we learn from data?
  - Overall pipeline: Data pulling -> Merging/Cleaning -> Model Training -> Evaluation.
  - MNIST digit recognition via KNN.
  - Logistic Regression example for churn at The New York Times.
- Model Evaluation and Complexity
  - Cross Validation - having a hold-out set for testing.
  - Hyperparameter optimization and overfitting. How complex is “just right”?

Outline

- Introduction to Supervised Learning
  - Problem Statement: Regression and Classification
  - Cross Validation - the test/train split.
  - Definition, and comparison of Lp norms.
  - How do we find the solution? The special case of p=2.
  - When do unique solutions exist?
  - The problem of dependent features.
- General Solution Finding Procedure - Gradient Descent
  - Gradient Descent, and Introduction to Convex Optimization.
  - Concrete Examples of Linear Regression and Gradient Descent in Python in an iPython Notebook.
- Introduction to Decision Trees
  - How decision trees are constructed - variance reduction.
  - Concrete Examples of Decision Trees in Python with an iPython Notebook.

Supervised Learning - Problem Statement

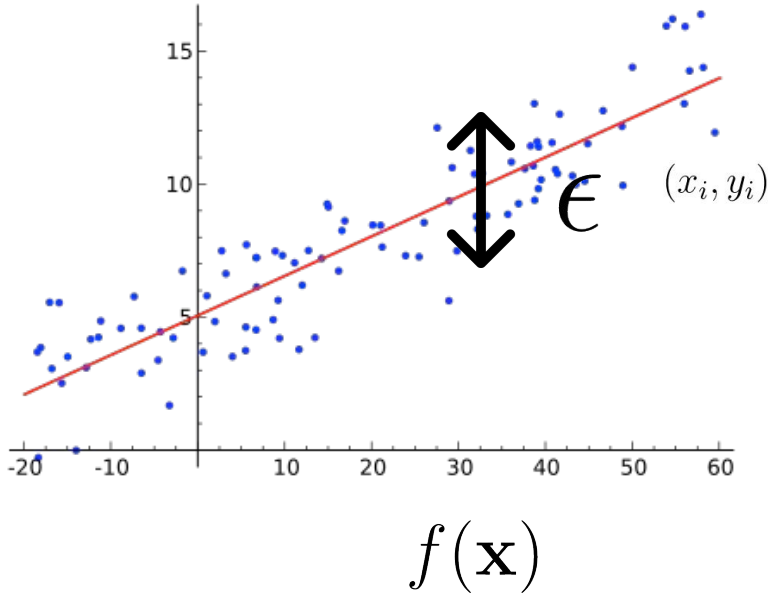
We assume that y, the response variable, is a function of covariates **X** (ie. variables, features).

$$y = f(\mathbf{x}) + \epsilon$$

Systematic information that x provides about y

Irreducible error

- The irreducible error ϵ <u>is natural and cannot be avoided</u>. For example, for any given attributes of an individual, height will have a natural variance. Or predicting income given someone's education, GPA, SAT scores, etc (assuming we chose the right distribution)

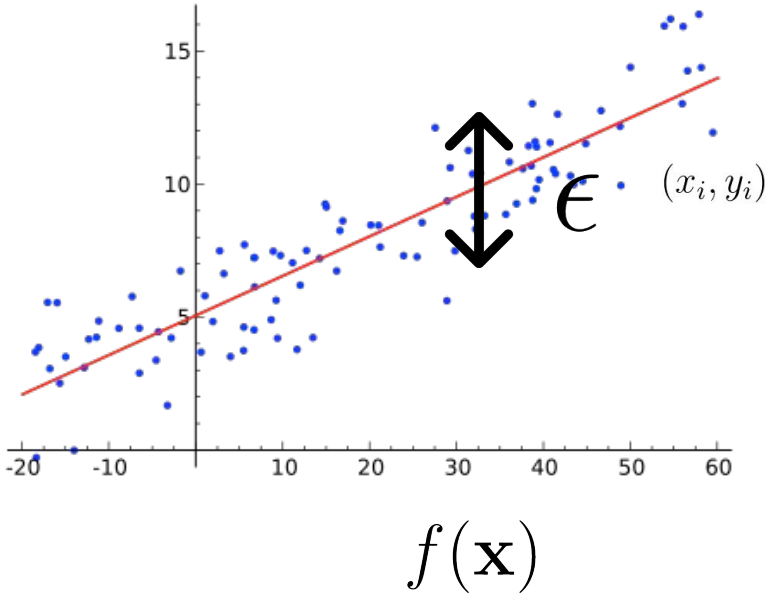


Supervised Learning - Problem Statement

We assume that y, the response variable, is a function of covariates **X** (ie. variables, features).

$$y = f(\mathbf{x}) + \epsilon$$

Our goal is to find an *estimator* $\hat{Y} = \hat{f}(\mathbf{X})$



- Our estimate of f will generally not be perfect, so we will have errors:
  - $f - \hat{f}$ which is known as **reducible error.**
  - ϵ which is known as **irreducible error.**

Goal: Our goal is to find the best f and to understand ϵ

To fix ideas in your head

In [5]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd
import pylab as plt
import seaborn
import numpy.random as nprnd
import random


%matplotlib inline

df = pd.read_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv', index_col=0)
df.head()
```

Goal: Predict sales units from advertising spend.
This is part of homework 1

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

$X =$

	TV	radio	newspaper
1	230.1	37.8	69.2
2	44.5	39.3	45.1
3	17.2	45.9	69.3
4	151.5	41.3	58.5
5	180.8	10.8	58.4

$Y =$

sales
22.1
10.4
9.3
18.5
12.9

Advertising Data

$\mathbf{X} =$

	TV	radio	newspaper
1	230.1	37.8	69.2
2	44.5	39.3	45.1
3	17.2	45.9	69.3
4	151.5	41.3	58.5
5	180.8	10.8	58.4

Matrix

$Y =$

sales
22.1
10.4
9.3
18.5
12.9

Vector

$\hat{Z}$

Hat represents an estimator of a statistical quantity Z from the data.

For example:

$$\hat{\mathbb{E}}(Y) := \frac{1}{N}\sum_{i=1}^{N} Y_i$$

$(\mathbf{x}_i, y_i)$ Training sample

$\mathbf{x}_i$ Training of x only

$\mathbf{x}$ Particular fixed value of x

Don’t worry if it seems like a lot at first. We will ease into it over time.

Supervised Learning - Problem Statement

We assume that y, the response variable, is a function of covariates **X** (ie. variables, features).

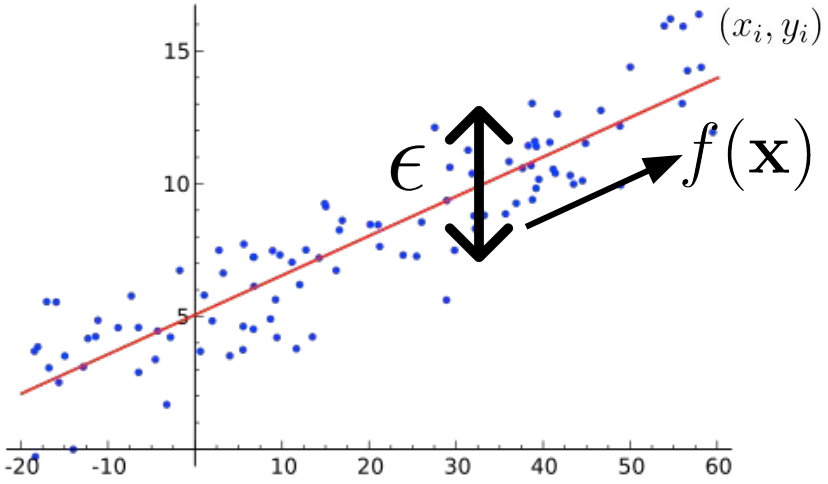
$$y = f(\mathbf{x}) + \epsilon$$

$\hat{Y}$ is our *estimator*. Our best guess at f given data.

$$\mathbb{E}(Y - \hat{Y})^2 = \mathbb{E}(f(\mathbf{X}) + \epsilon - \hat{Y})^2$$

$$= \mathbb{E}(f(\mathbf{X}) - \hat{Y})^2 + 2\mathbb{E}(\epsilon(f(\mathbf{X}) - \hat{Y})) + \mathbb{E}(\epsilon^2)$$

$$= \underbrace{\mathbb{E}(f(\mathbf{X}) - \hat{Y})^2}_{\text{reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{irreducible}}$$



We cannot do anything to reduce ϵ, so we need to find the best $\hat{Y}$

Supervised Learning - Learning from Data

In statistical learning, we only have access to samples from our distribution (x_i, y_i)

$$\hat{Y} = \hat{f}(\mathbf{X})$$

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(\mathbf{x}_i))^2$$

We approximate the expectation with sampling from the empirical distribution, since this is all we have access to.

We then write our estimator as a function of our variables x.

Goal: Find $\hat{Y} = \hat{f}(\mathbf{X})$ which minimizes the above

Supervised Learning - What is f?

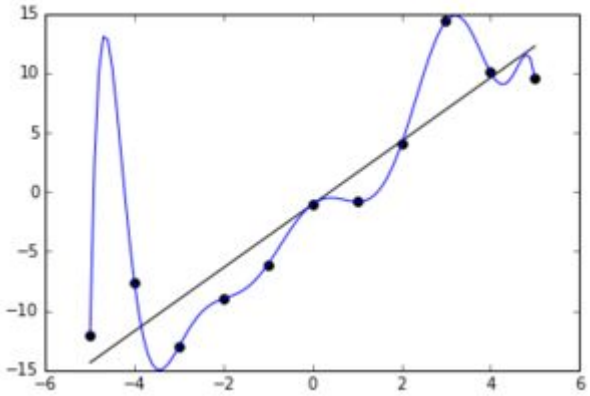
In statistical learning, we only have access to samples from our distribution (x_i, y_i)

$$\hat{Y} = \hat{f}(\mathbf{X})$$

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(\mathbf{x}_i))^2$$

Why not just choose $\hat{f}(x_i) = y_i$ **?**

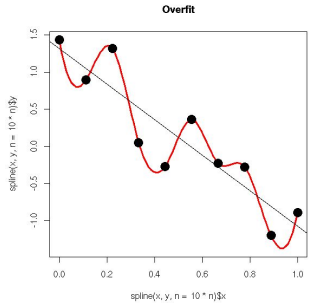


Supervised Learning - What is f?

In statistical learning, we only have access to samples from our distribution (x_i, y_i)

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(\mathbf{x}_i))^2$$



$$\hat{Y} = \hat{f}(\mathbf{X})$$

Why not just choose $\hat{f}(x_i) = y_i$ **?**

Answer: This would not generalize to new data since it is defined in terms of the training data only. We need something which can be used to make predictions on any new value of x.

Supervised Learning - What is f?

In statistical learning, we only have access to samples from our distribution (x_i, y_i)

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(\mathbf{x}_i))^2$$

We need to start from some hypothesis for $\hat{Y} = \hat{f}(\mathbf{X})$ which can be applied to new values of x.

$$\hat{Y} = \hat{f}(\mathbf{X})$$

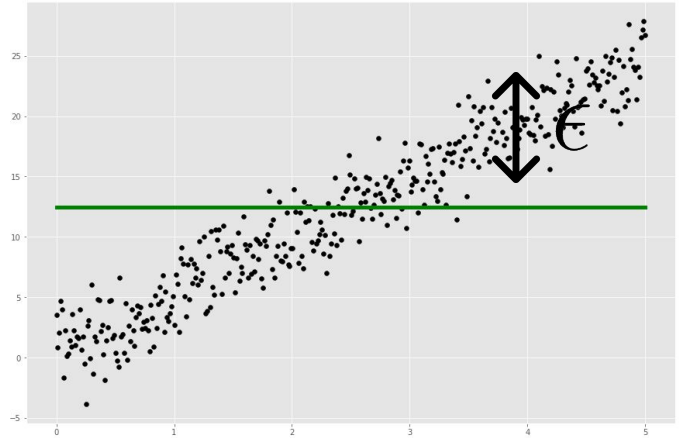
Some possible choices:

Linear: $\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$

Nearest Neighbor: $\hat{f}(x) = \frac{1}{k}\sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$

Hypothesis 1: Assume f is constant

$$\hat{Y} = \hat{f}(\mathbf{x}) = \text{const.}$$



Solution:

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

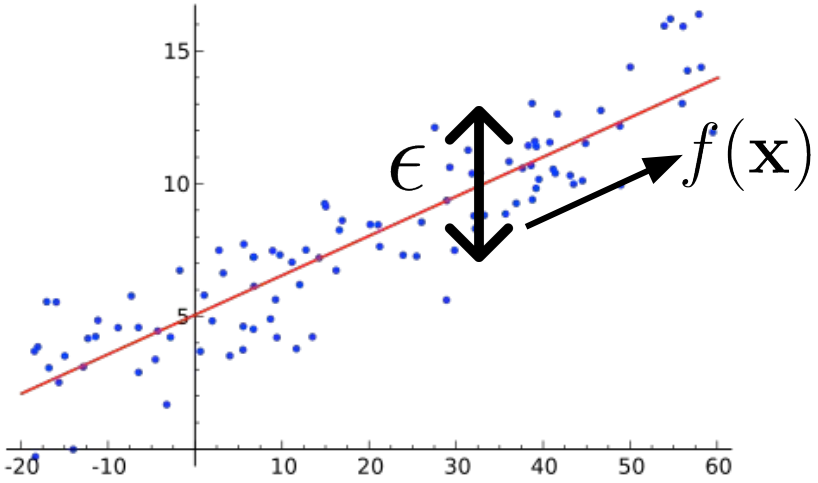
$$\hat{y}^* := \text{argmin}_{\hat{y}} \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 = \frac{1}{N}\sum_{i=1}^{N} y_i$$

The mean of the data

General case

What if f is **not constant,** but **constant for each x**?

$$f(\mathbf{x}) = c_{\mathbf{x}} \quad \Rightarrow \quad y_i = c_{\mathbf{x}_i} + \epsilon_i$$



Solution:

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum^{N}_{i=1}(y_i - \hat{y})^2$$
$$= \sum_{x}\sum_{x_i = x}(y_i - c_x)^2$$

$$\Rightarrow \quad c_{\mathbf{x}} = \frac{1}{|x_i = x|}\sum_{x_i = x} y_i = \mathbb{E}(y|\mathbf{X} = \mathbf{x})$$

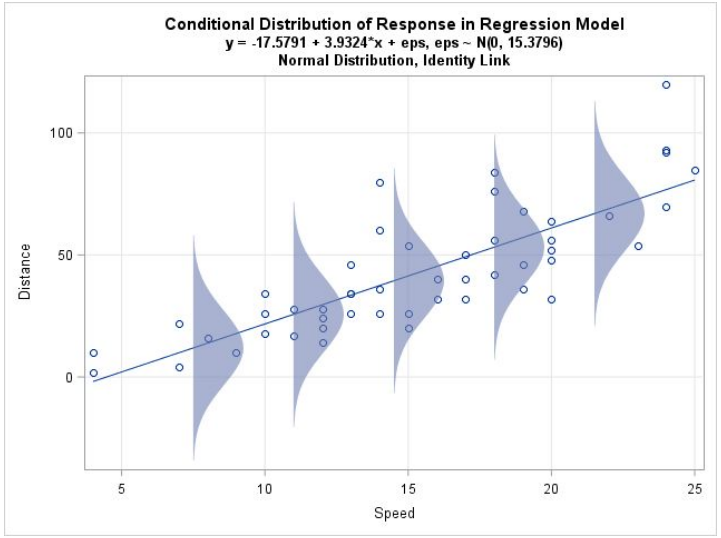
The conditional mean of the data (on x).

We have a form for f!

We seek to find an estimator for the conditional mean of y.

$$f(\mathbf{x}) = \mathbb{E}(y|\mathbf{X} = \mathbf{x})$$

The rest of the error will be the natural uncertainty in the distribution which we cannot model (the irreducible error).



Note: We are not always seeking to predict the mean! This is a special case of the least squares norm (we will see more on this later).

Which of these two assumptions seems better?

$$f(\mathbf{x}) = \mathbb{E}(y|\mathbf{X} = \mathbf{x})$$

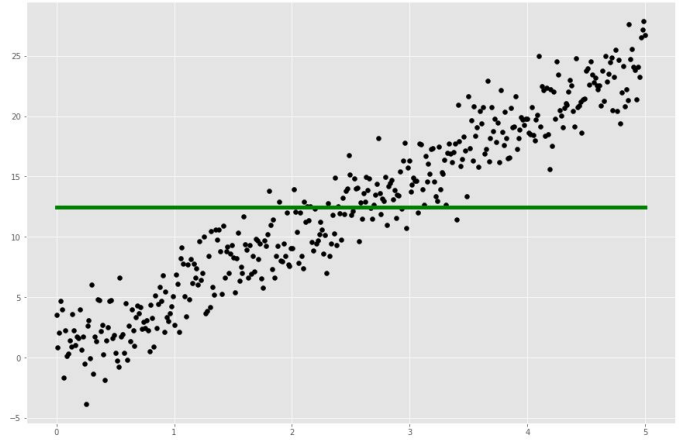
We thus seek an estimator to the above, which we denote as:

$$\hat{f}(\mathbf{x}) = \hat{\mathbb{E}}(Y|\mathbf{X} = \mathbf{x})$$

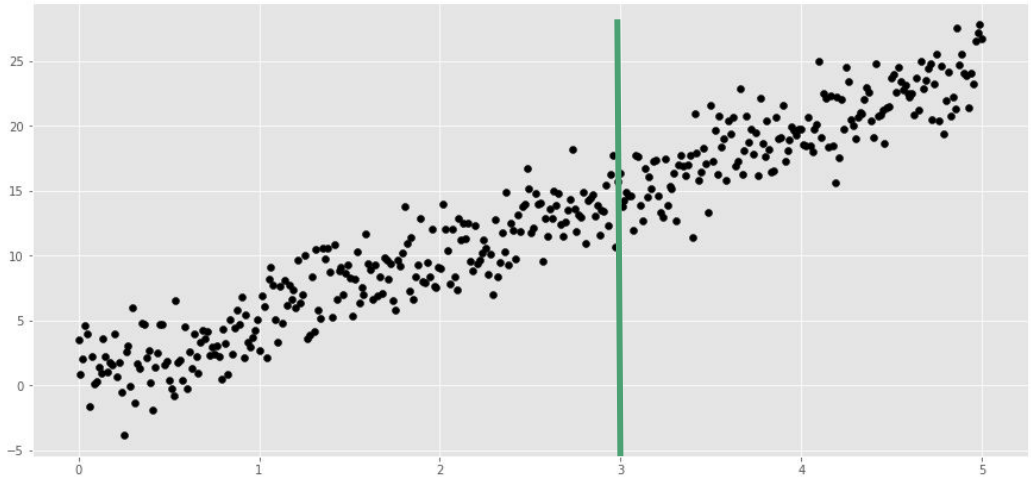
Some possible choices:

Linear: $\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$

Nearest Neighbor: $\hat{f}(x) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$



Different hypothesis functions case 1



$$y = f(x) + \epsilon$$

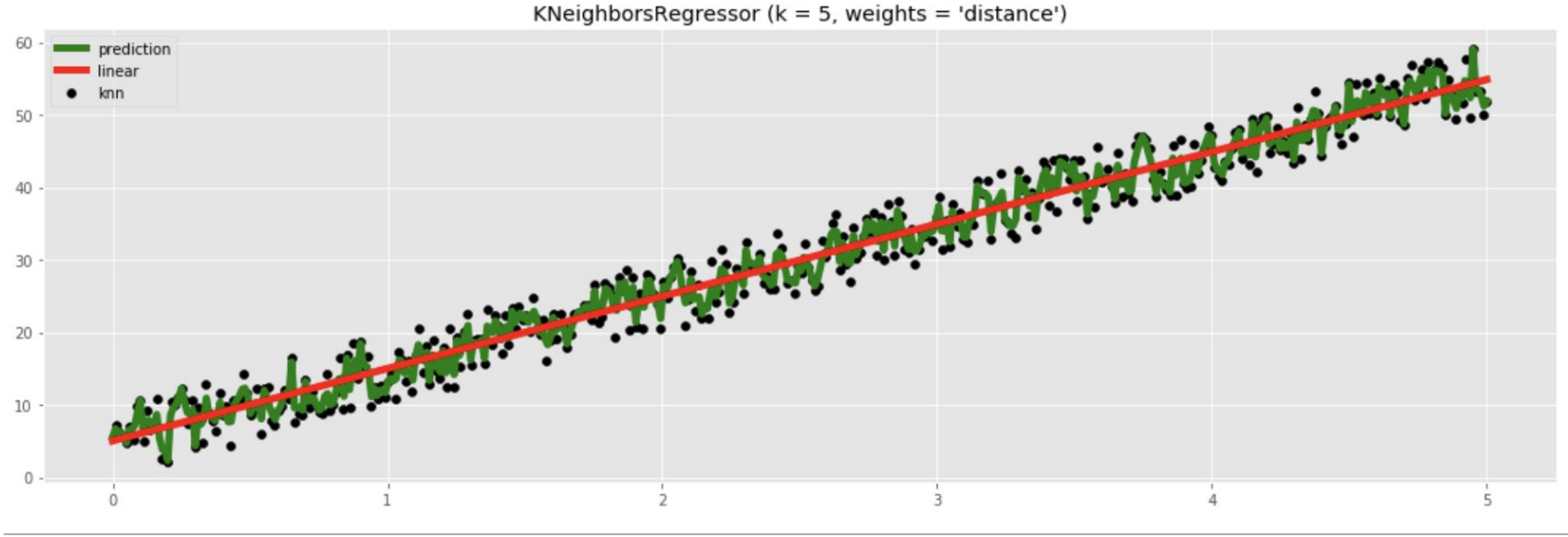
$$\epsilon \sim \mathcal{N}(0, 2)$$

$$\hat{f}(x) = \hat{\mathbb{E}}(y|X = x) = ?$$

Linear: $\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$

Nearest Neighbor: $\hat{f}(x) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$

Different hypothesis functions case 1

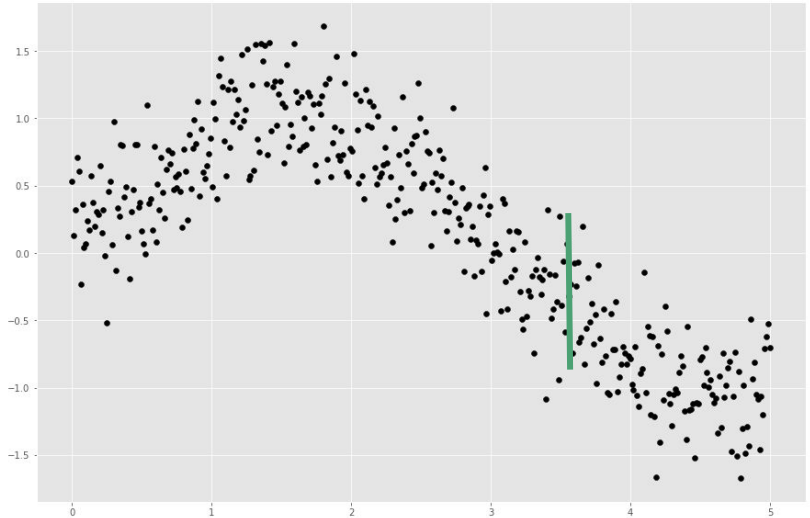


Nearest neighbour has very low bias but high variance.

Linear model has high bias, low variance and seem to model data well.

Much of machine learning is using different hypothesis to balance these two extremes.

Different hypothesis functions case 2



$$y = f(x) + \epsilon$$

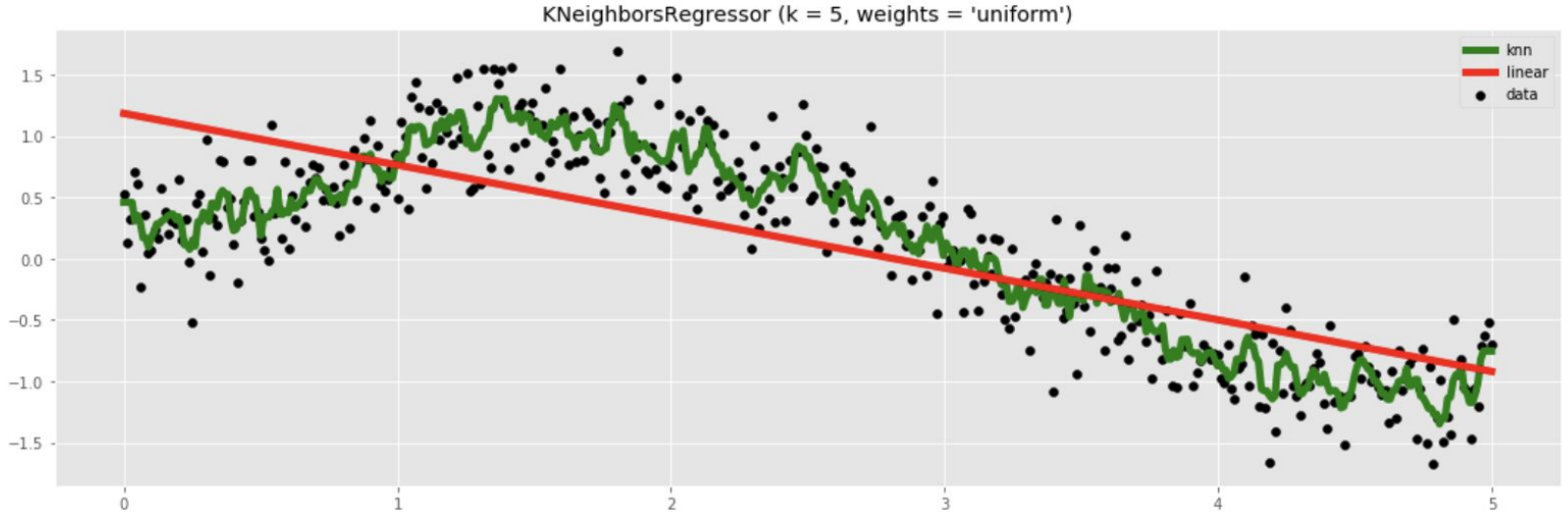
$$\epsilon \sim \mathcal{N}(0, 2)$$

$$\hat{f}(x) = \hat{\mathbb{E}}(y|X = x) = ?$$

Linear: $\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$

Nearest Neighbor: $\hat{f}(x) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$

Different hypothesis functions case 2



In this case the linear model has too strong of a bias, and the nearest neighbor seems to pick up the pattern better

Do we always predict the mean? No

$$\min_{\hat{f}} \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{f}_i| \quad f(\mathbf{x}) = c_{\mathbf{x}}$$

$$= \frac{1}{N} \min_{c_x} \sum_{x} \sum_{x_i = x} |y_i - c_x|$$

- We will encounter many other loss functions in this class.
- The most important thing to remember is that we need to model a statistic of our data which ensures that our reducible error is truly random, and not depending on x.

Do we always predict the mean? No

$$\min_{\hat{f}} \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{f}_i| \quad f(\mathbf{x}) = c_{\mathbf{x}}$$

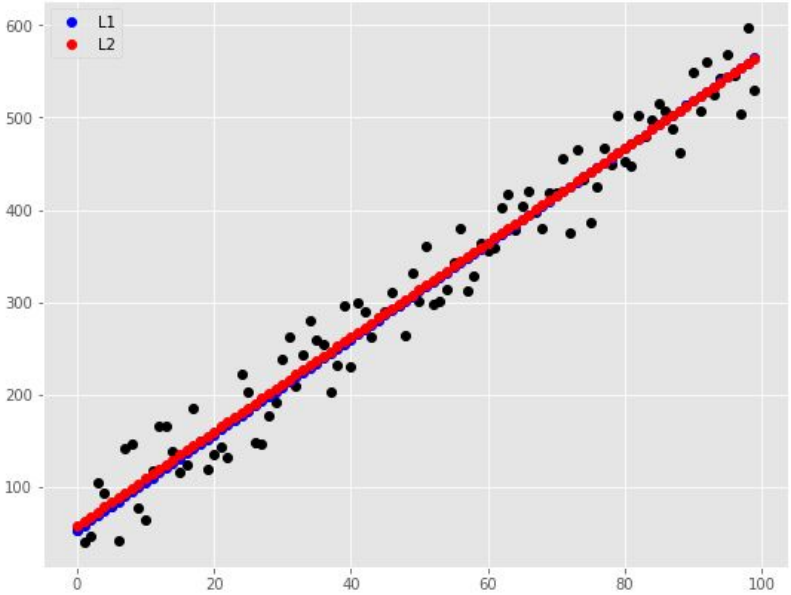
$$= \frac{1}{N} \min_{c_x} \sum_{x} \sum_{x_i = x} |y_i - c_x|$$

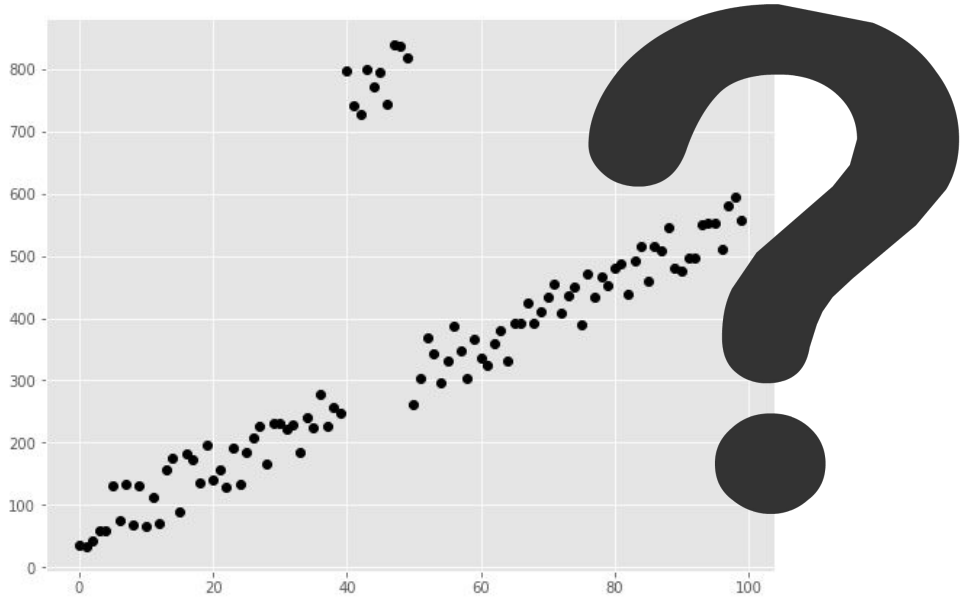
Solution:

$$\hat{f}(\mathbf{x}) = \text{median}(y|\mathbf{X} = \mathbf{x})$$

We will encounter many other loss functions in this class. The most important thing to remember is that we need to model a statistic of our data which ensures that our reducible error is truly random, and not depending on x.

Comparison of L1 and L2

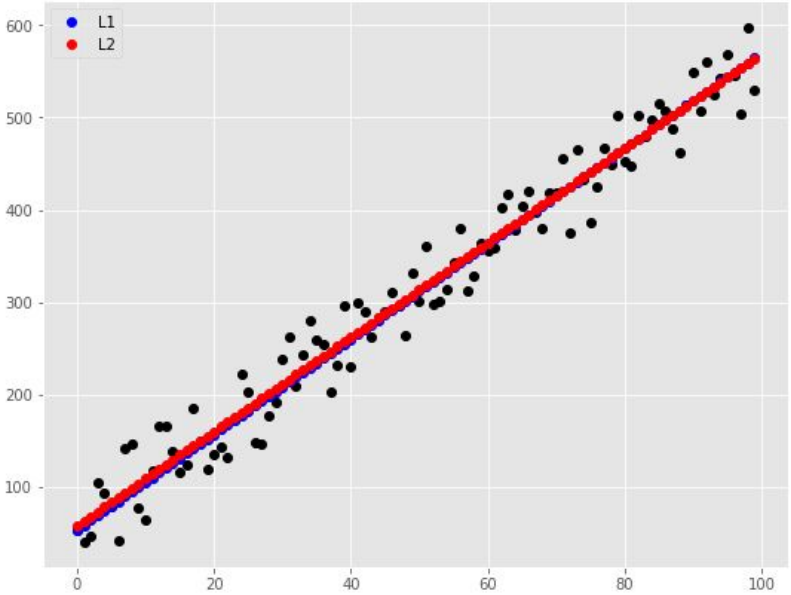




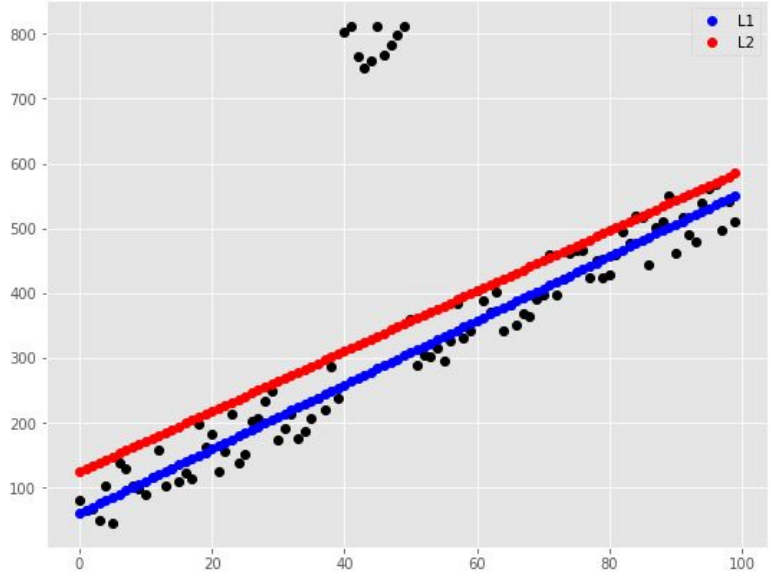
No outliers. L1 and L2 have almost identical solutions.

What if we added outliers?

Comparison of L1 and L2



No outliers. L1 and L2 have almost identical solutions.

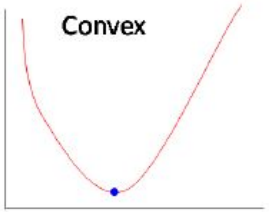


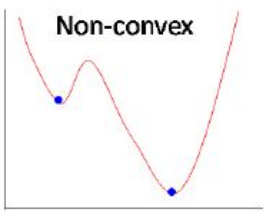
Outliers. L1 is much more sensitive to outliers.

General Goal

More generally, joint random variables $(\mathbf{x}, y)$ with distribution $P(\mathbf{x}, y)$ we seek to minimize:

$$\mathcal{L}(\hat{f}) := \mathbb{E}_P L(y, \hat{f}(\mathbf{x}))) \sim \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$





- The choice of L is determined is often determined by either know.. properties of the distribution of our data, or from desired properties of the solution.
- The choice of the general form of $\hat{f}$ is complex and an active area of research. However most forms are linear and nonlinear combinations of the linear and piecewise constant methods we have just seen.

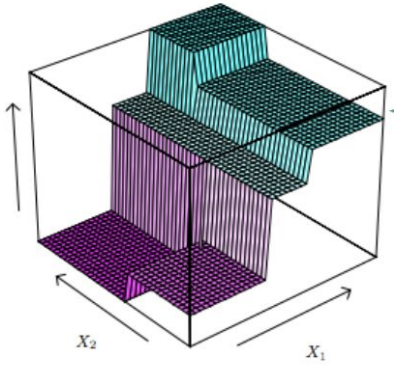
- We must always balance bias and variance with any choice, and this is much of the art of machine learning.

- The **ideal property for any minimization problem is that it is convex. This way we can ensure there exists a unique, stable solution.**

Two broad categories of models

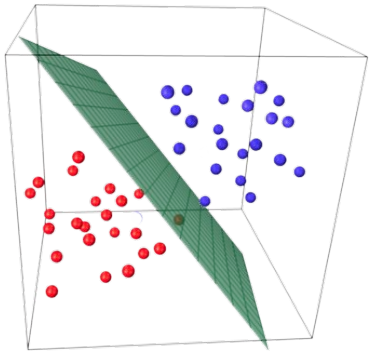
More generally, joint random variables $(\mathbf{x}, y)$ with distribution $P(\mathbf{x}, y)$
we seek to minimize:

$$\mathcal{L}(\hat{f}) := \mathbb{E}_P L(y, \hat{f}(\mathbf{x}))) \sim \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$



Non-parametric

$$\hat{f}(\mathbf{x}_i) = \mathbf{1}_{R_t(\mathbf{x}_i)}(\mathbf{x}_i) \text{ for } \mathbf{x}_i \in R_{t(i)}$$



Parametric

$$\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$$

Losses

$L(a, b)$ examples

$|a - b|^p$ for $p = 1, 2$

$a \log b + (1 - a) \log(1 - b)$

Linear Regression

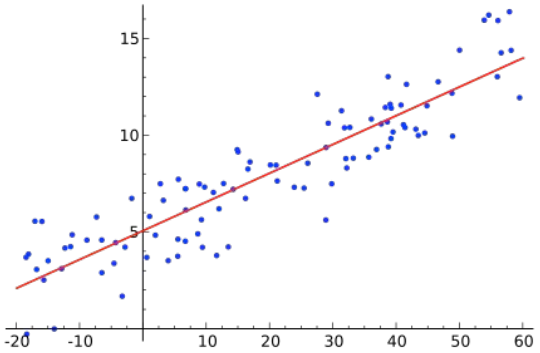
Linear Regression

In statistical learning, recall we only have access to samples from our distribution (x_i, y_i)

$$\mathbb{E}(y - \hat{y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(\mathbf{x}_i))^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta \cdot \mathbf{x_i})^2$$



We assume here that the conditional mean of our distribution is a linear function of the input variable:

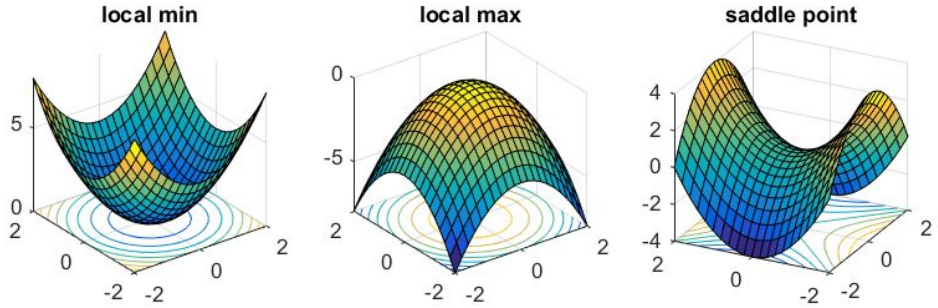
$$\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$$

Goal: Minimize this function and find the best beta.

Convex Analysis (Calculus)

- Recall from last time that our goal is often find some convex function $\mathcal{L} : \mathbf{S} \to \mathbb{R}$ such that we learn the rules of our model by its minimization (ie. coefficients). These are referred to as **parametric methods.**

- This is **not always the case**, with algorithms such as **decision trees** or **neural nets**. However, it covers many cases in supervised learning. These are called **non-parametric methods** (to be covered in this class or next).



Convex Analysis (Calculus)

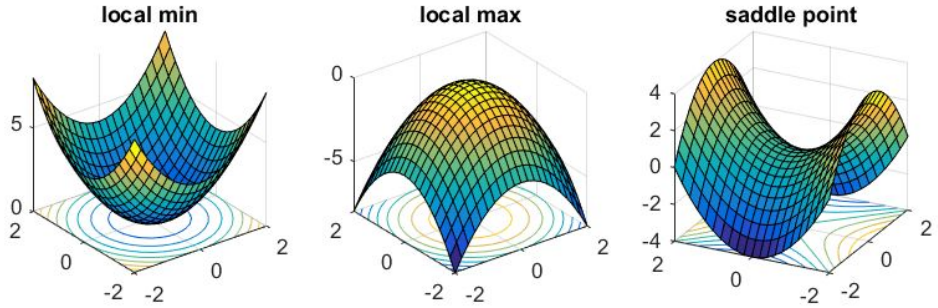
- Assume $\mathcal{L} : \mathbf{S} \to \mathbb{R}$ is continuously differentiable and convex, ie.

$$\mathcal{L}''(\beta) \geq 0 \text{ for all } \beta \in \mathbf{S}$$

$$\exists \beta_0 \text{ such that } \mathcal{L}(\beta_0) \leq \mathcal{L}(\beta) \text{ for all } \beta \in S$$

- There f has a minimum value. Moreover, the minimum is unique when f is strictly convex, ie.

$$\mathcal{L}''(\beta) \geq c > 0 \text{ for all } \beta \in \mathbf{S}$$



Solution to Ordinary Least Squares

Analytical Solutions and Stability Analysis

Analytical Solution to OLS

$$\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^2$$

Claim: When p = 2 and the matrix $X^T X$ is invertible, the above is minimized uniquely by

$$\beta = (X^T X)^{-1} X^T y$$

The solution can make sense when $X^T X$ isn't invertible (in practice it often can be "almost not invertible", but solutions will be unstable and degenerate (more later)

Analytical Solution to OLS when p=2

Proof:

$$\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^2$$

- Need linearly independent features for a unique inversion.

Differentiating, we have

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \beta \cdot x_i)x_j$$

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = 0 \Rightarrow \boxed{\frac{1}{N}\sum_{i=1}^{N}(\beta \cdot x_i)x_{ij} = \frac{1}{N}\sum_{i=1}^{N} y_i x_{ij}.} \Rightarrow \quad X^T X\beta = X^T y.$$

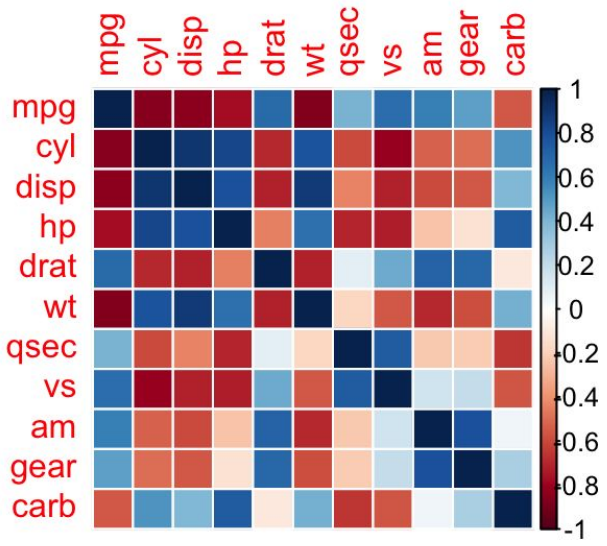
Working with indices is hard! Try not to use them and instead using matrix notation.

$$\Rightarrow \qquad \beta = (X^T X)^{-1} X^T y$$

Analytical Solution to OLS

It easily follows that $$\frac{d^2\mathcal{L}}{d^2\beta} = \frac{2}{N}X^TX$$

- From this, it follows there is a **unique solution** iff the **eigenvalues** of the above matrix are **strictly positive**.
- This occurs precisely when **X has linearly independent features so that the above matrix is positive definite.**



Let's try it out

```
In [100]: pred_cols = ['TV','radio','newspaper']

          X = df[pred_cols]
          y = df['sales']
```

```
In [102]: A=(X.T).dot(X)
```

```
In [105]: Ainv=np.linalg.inv(A)
```

```
In [154]: beta = Ainv.dot(X.T).dot(y)
```

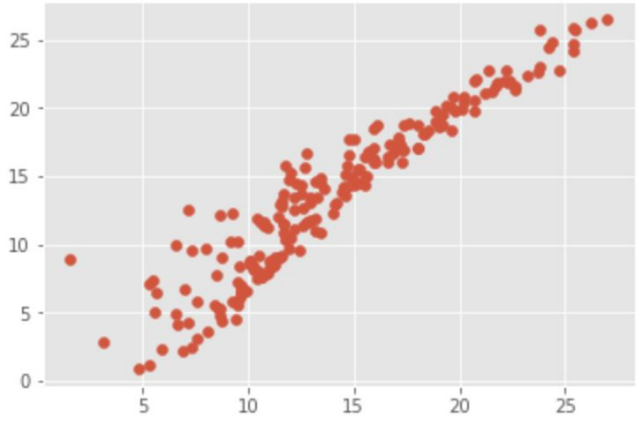
$$A = X^T X$$

$$\beta = (X^T X)^{-1} X^T y$$

Seems easy right?

Let’s see how good it looks

```
In [156]: z = X.dot(beta)

In [158]: plt.scatter(y,z)
Out[158]: <matplotlib.collections.PathCollection at 0x1c2c1c6190>
```



Predicted

Actual

Not bad! We didn’t check our assumptions first on the Hessian, but this seems reasonable at first glance.

Evaluating goodness of fit

How to evaluate?

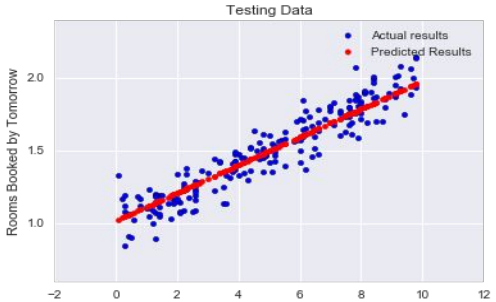
For regression problems, we generally use two possible metrics:

- **Root mean squared error:**

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$$

- **R-squared:**

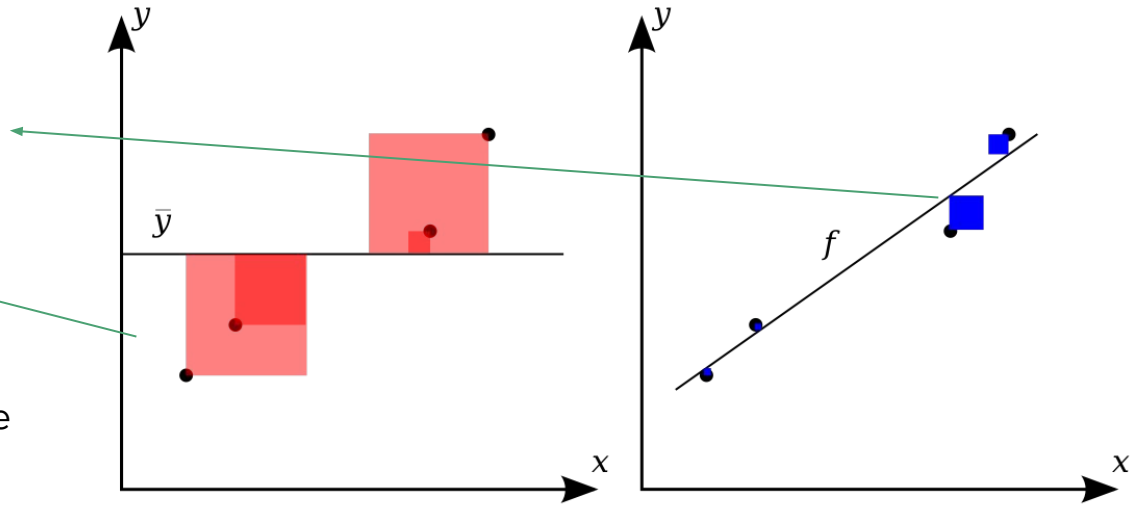
$$1 - \frac{\sum_{i=1}^{N}(f(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$



- This is the fraction of variance that our model is able to explain.

Coefficient of Determination (R^2)

$$1 - \frac{\sum_{i=1}^{N}(f(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$



- This is the percentage of variance that our model is able to explain.
- A perfect model would have the second term be 0, resulting in 1.
- If the model was simply the mean, the result would be 0.

Back to our example

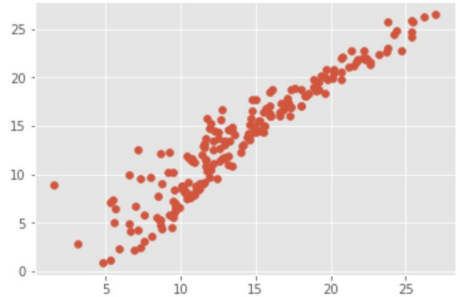
```
In [173]: def rmse(y,z):
              n = len(y)
              return np.sqrt((1.0/n)*np.sum([(y[i]-z[i])**2 for i in range(1,n)]))

In [175]: def r2(y,z):
              n = len(y)
              ymean=np.mean(y)
              top = (1.0/n)*np.sum([(y[i]-z[i])**2 for i in range(1,n)])
              bottom = (1.0/n)*np.sum([(y[i]-ymean)**2 for i in range(1,n)])
              return 1-top/bottom

In [176]: rmse(y,z)
Out[176]: 2.0096487117612014

In [177]: r2(y,z)
Out[177]: 0.8508818180109601
```

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$$

$$1 - \frac{\sum_{i=1}^{N}(f(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

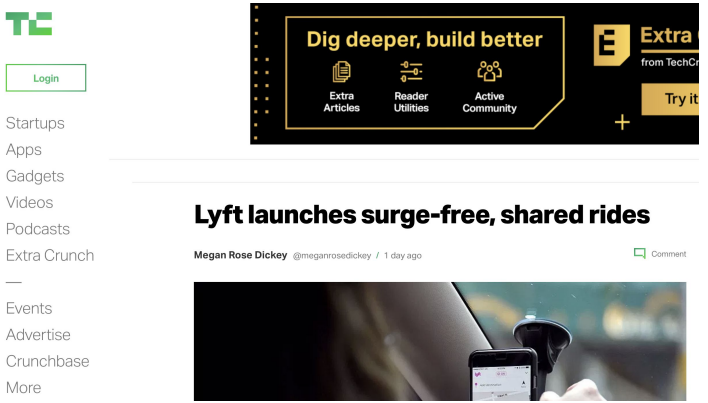
Quite good!

```
In [156]: z = X.dot(beta)

In [158]: plt.scatter(y,z)
Out[158]: <matplotlib.collections.PathCollection at 0x1c2c1c6190>
```



Announcements

- Homework 0 due.
- Project I just finished working on was launched yesterday. Shared Saver.
- Today: Continue supervised learning. Homework 1 due next week.



Interpreting the solution

How do I spend my money?

Advertisement Example

What are the features?

TV: advertising dollars spent on TV for a single product in a given market (in thousands of dollars)

Radio: advertising dollars spent on Radio

Newspaper: advertising dollars spent on Newspaper

Goal: Which advertising channel has the most impact?.

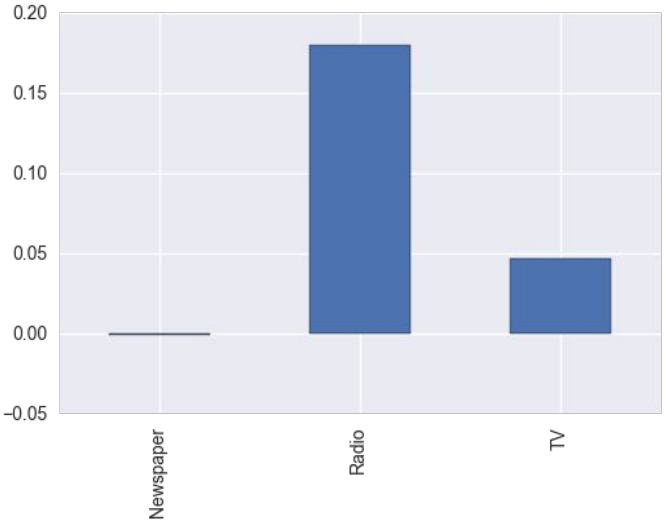
In [263]:
```
# read data into a DataFrame
import pandas as pd
import pylab as plt
import seaborn
from sklearn.linear_model import LinearRegression
import numpy.random as nprnd
import random
import json
pd.set_option('display.max_columns', 500)
%matplotlib inline

df = pd.read_csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv', index_col=0)
df.head()
```

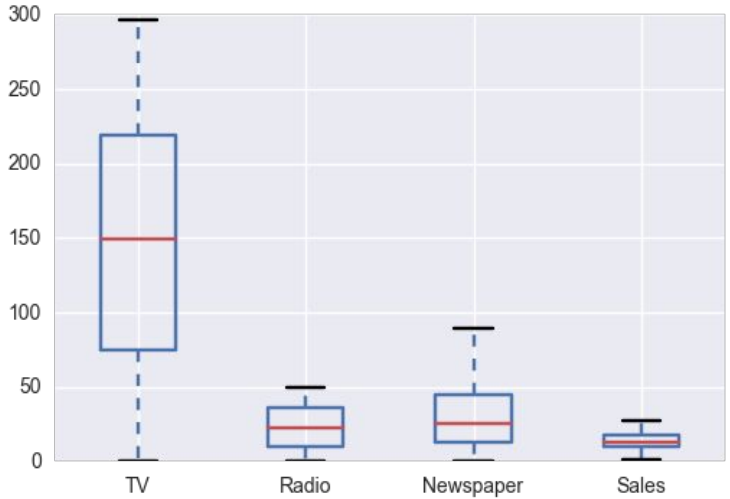
Out[263]:

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

I highly recommend reading the first two chapters of *Introduction to Statistical Learning* (available in the pdf folder in the course repo).

Plot coefficients and range of values





$$\beta = (X^T X)^{-1} X^T y = \begin{pmatrix} 0.00 & 0.19 & 0.05 \end{pmatrix}$$

Conclusion: So newspaper has no influence on sales?

Feature importance & Inference

$$y_i = \sum_{j=1}^{N} \beta_j X_{ij}$$

When there is no relationship between the features, we can conclude that

$$\frac{\partial y_i}{\partial X_{ik}} = \beta_k$$

- If our features were all independent, this would allow us to interpret the significance of each variable, in terms of how much it changes the response variable.
- **Does not work if there is any interaction or dependence between the features however**.
- To **compare coefficients,** we often (but not always) need to **normalize the columns to have the same scale**.

Separate Regression Models

	Coefficient	Std. error	t-statistic	p-value
Intercept	9.312	0.563	16.54	< 0.0001
radio	0.203	0.020	9.92	< 0.0001

Simple regression of sales on newspaper

	Coefficient	Std. error	t-statistic	p-value
Intercept	12.351	0.621	19.88	< 0.0001
newspaper	0.055	0.017	3.30	< 0.0001

	Coefficient	Std. error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

In separate models, we observe that newspaper has a non-zero coefficient, but it was zero when doing multiple regression. Is this possible?

One Regression Model

In this model newspaper has no influence - what gives??

	Coefficient	Std. error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

TABLE 3.4. *For the* Advertising *data, least squares coefficient estimates of the multiple linear regression of number of units sold on radio, TV, and newspaper advertising budgets.*

How is this possible?

This difference stems from the fact that in the simple regression case, the slope term represents the **average effect of a $1,000 increase in newspaper advertising, ignoring other predictors such as TV and radio**. In contrast, in the multiple regression setting, the coefficient for newspaper represents the average effect of increasing newspaper spending by $1,000 while holding TV and radio fixed.

Causation vs Correlation

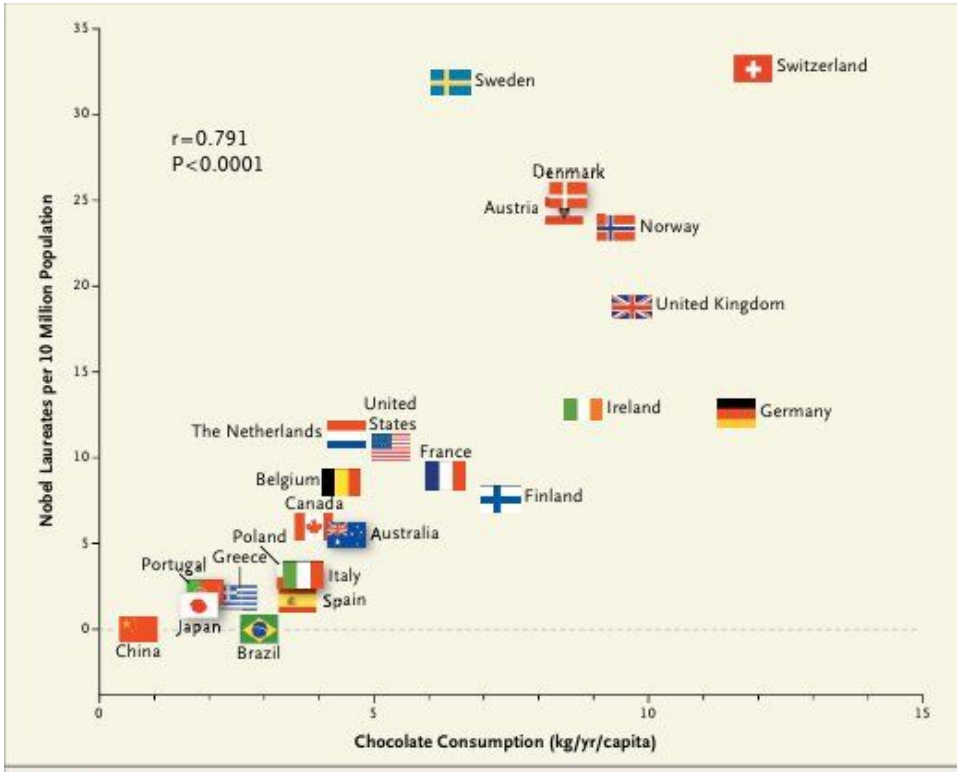
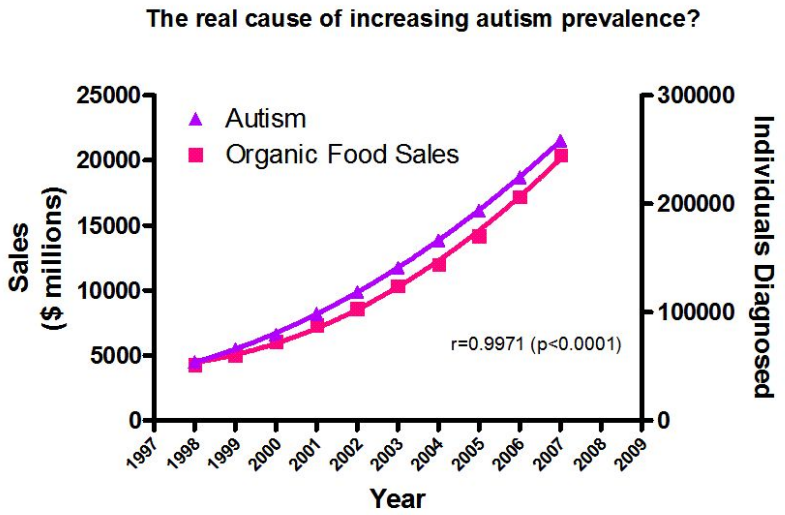


Figure 1. Correlation between Countries' Annual Per Capita Chocolate Consumption and the Number of Nobel Laureates per 10 Million Population.



Sources: Organic Trade Association, 2011 Organic Industry Survey; U.S. Department of Education, Office of Special Education Programs, Data Analysis System (DANS), OMB# 1820-0043: "Children with Disabilities Receiving Special Education Under Part B of the Individuals
with Disabilities Education Act

Feature importance & Inference

Multiple Regression

$$y = \sum_{k=1}^{3} \beta_k x_k$$

Coefficient represents number of increase in sales for a unit increase in variable k assuming all others fixed.

$$\left.\frac{\partial y}{\partial x_k}\right|_{x_j=\text{const}} = \beta_k \text{ for } j \neq k$$

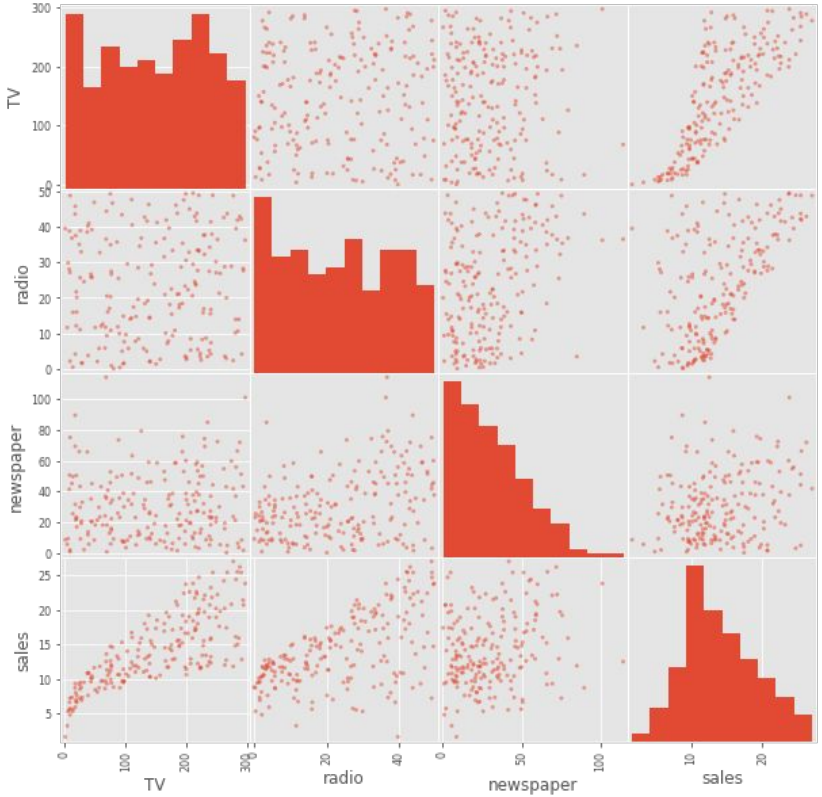
Single Regression

$$y = \beta_k x_k$$

Coefficient represents number of increase in sales for a unit increase in variable k without accounting for other possible influential variables. So newspaper was getting unfair credit!

$$\frac{\partial y}{\partial x_k} = \beta_k$$

Correlation Matrix



	TV	radio	newspaper	sales
TV	1.0000	0.0548	0.0567	0.7822
radio		1.0000	0.3541	0.5762
newspaper			1.0000	0.2283
sales				1.0000

We see that newspaper has a very high correlation with radio. This means that in markets where one spends more money on radio, they also tend to spend more money no newspaper, and newspaper then is correlated with sales unfairly.

Summary

- Interpreting the coefficients in a linear model **can be quite difficult to do**. We saw that individual models **may attribute significance** to a variable which **has no real causal relationship with our outcome variable.**

- Even when we include as many variables as possible, t**here is still always a chance we are missing some other variables which are accounting for the relationship**. This is why **controlled experiments are often the only way to make causal inferences about the impact of dependent variables on independent ones.**

- Checking the **correlation matrix** is a good way of understanding if there are relationships between your variables that could be influencing one another.

- In this lecture we will understand how to define the confidence in the estimated coefficient. In our regularization lecture, we will learn how to effectively remove features which don't have predictive power such as newspaper in this case.

Warning

- At many companies, you will be pressured to answer the **“why”.** Why are users going to unsubscribe, why are they subscribing?
- It’s are to be able to give a good answer to this question given considerations above.
- We will discuss some ways of doing this later though.

Confidence in Coefficients

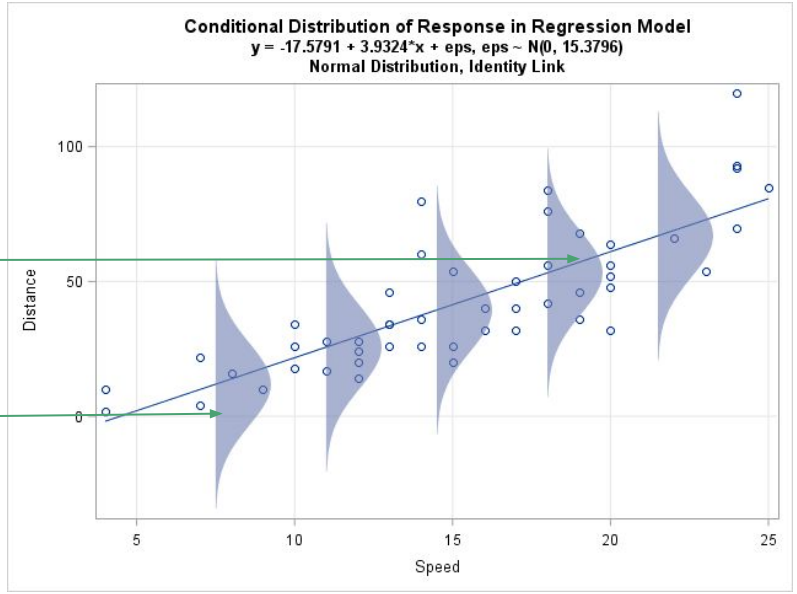
When can we make inferences, and how confident are we?

Understanding uncertainty

$$y = \hat{y} + \epsilon$$

$\hat{y}$ Estimator (mean)

ϵ Error



In machine learning, our goal is to find the best **estimator** in the presence of natural uncertainty **(error).**

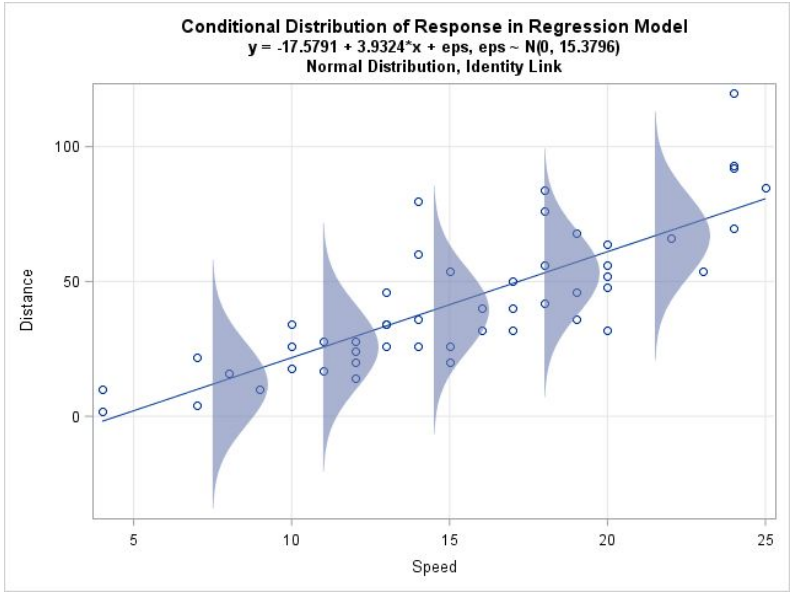
What is the error?

$$y = \hat{y} + \epsilon$$

ϵ Is assumed to be from some distribution, ie.

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

For example: *you could never predict the height of someone from their age, gender, ethnicity or any other collection of variables alone - there is a natural variance in the heights of individuals.*



What is the error?

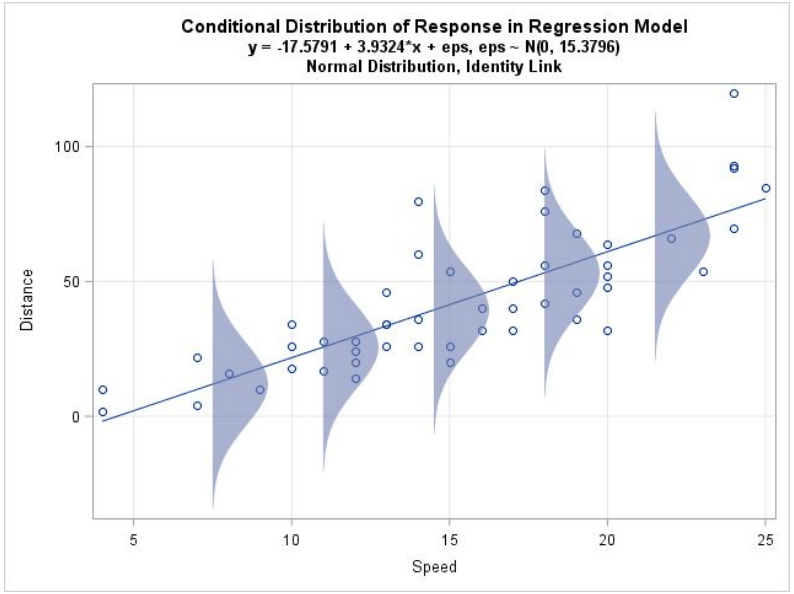
$$y = (\hat{\beta} + \epsilon_\beta) \cdot x + \epsilon$$

ϵ Is assumed to be from some distribution, ie.

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$\mathbb{E}(y|x) = \beta \cdot x$$

There is an error in our estimate of the true parameter which we can improve, and the natural error occuring which is irreducible, ϵ



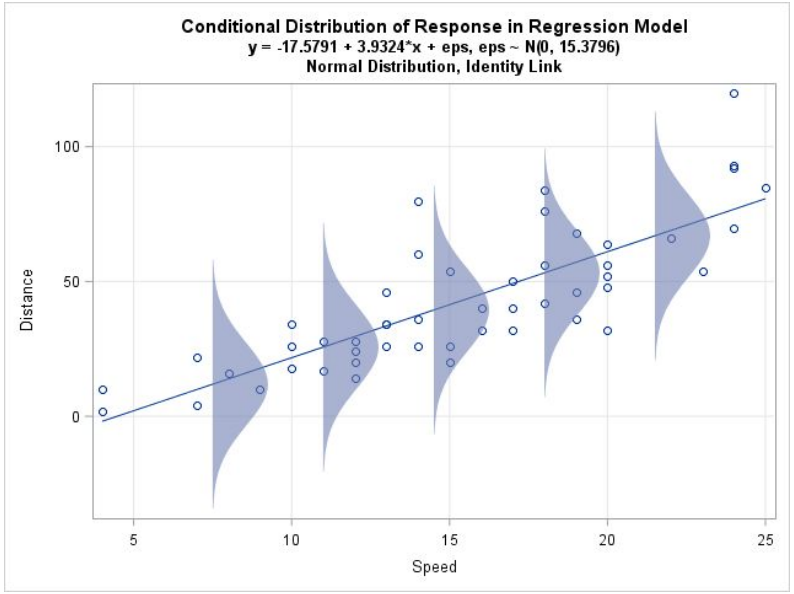
For ordinary least squares, we assume normality.

$$\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^2$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\mathbb{E}(y|x) = \beta \cdot x$$

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-\beta x)^2/2\sigma^2}$$



This will be explained in future lectures and assignments!

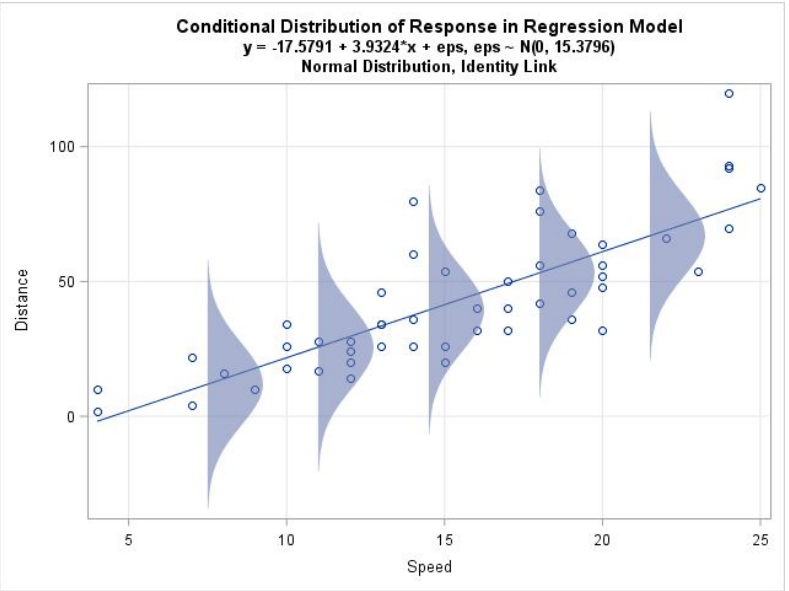
Let's now check some examples to see when this holds.

Prediction vs Inference

An important point which is often confused in machine learning is if and when we need our assumptions on the error to be true.

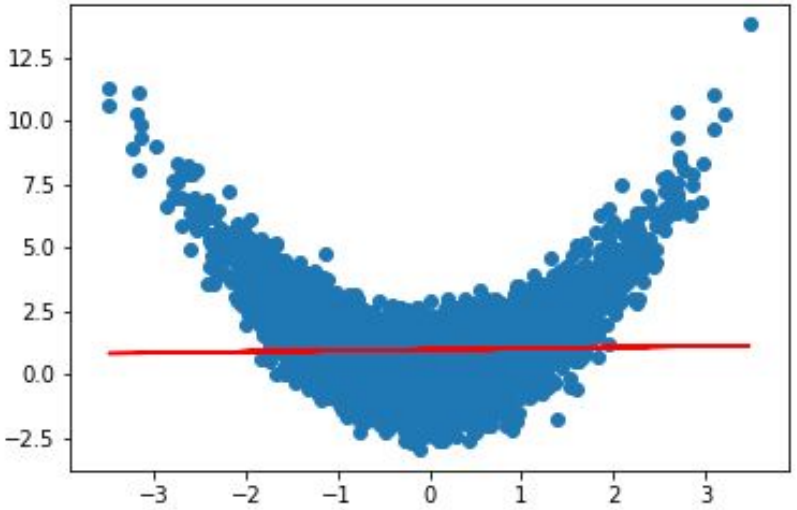
For example, can I perform linear regression even when my residual errors are not normal? **Yes, of course. You're just minimizing a norm.**

Can I interpret the confidence in my estimate of the coefficient? **No.**



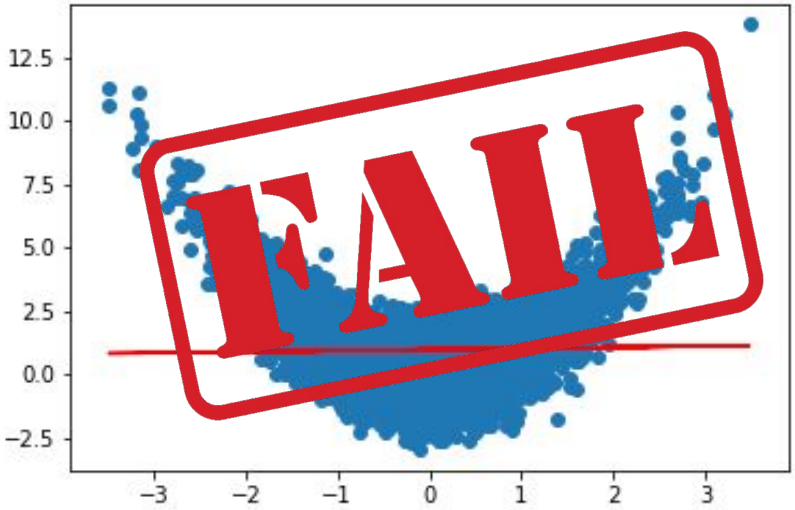
What assumptions does this imply?

Assumption 1 - Linear relationship between dependent and independent variables.



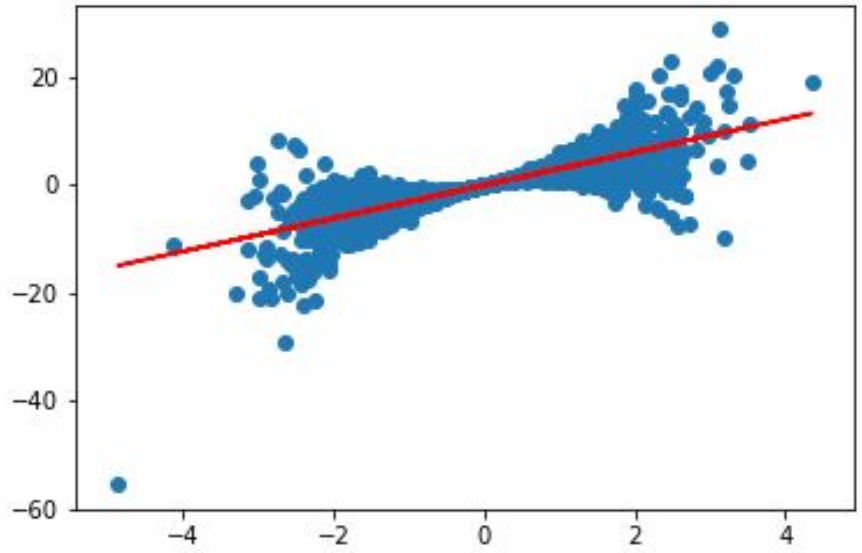
What assumptions does this imply?

Assumption 1 - Linear relationship between dependent and independent variables.



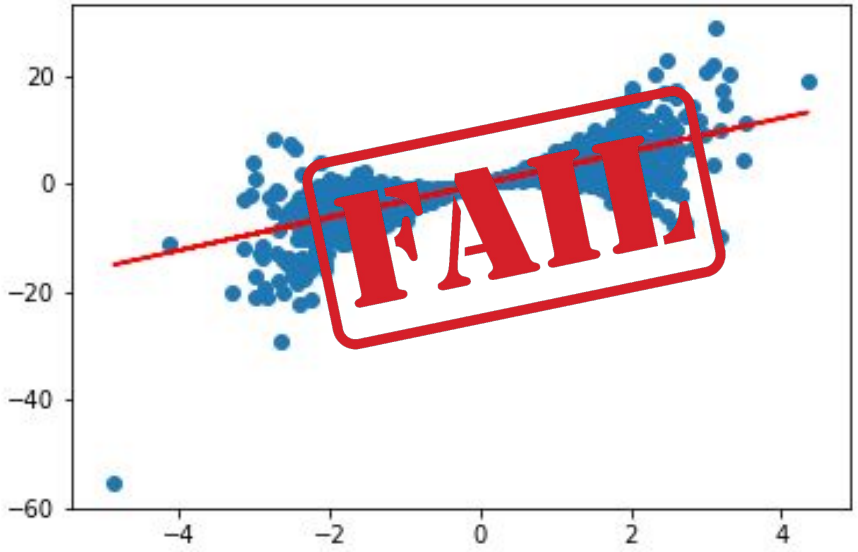
What assumptions does this imply?

Assumption 2 - Heteroscedasticity: constant variance for any value of $\mathbb{E}(y|x) = \beta \cdot x$



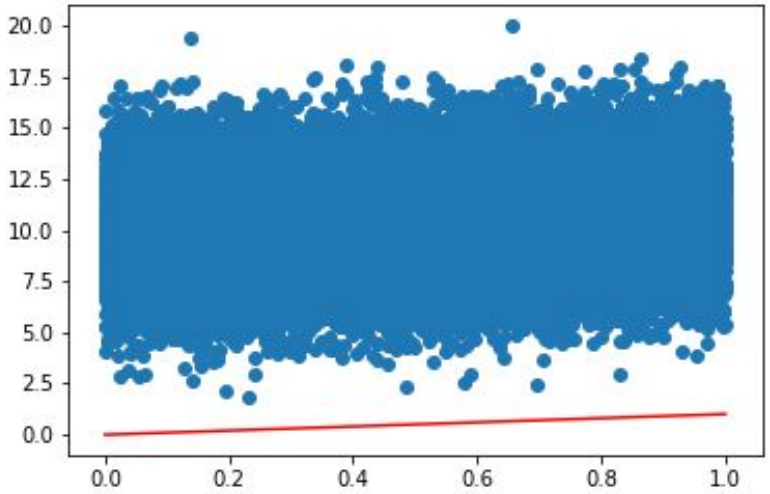
What assumptions does this imply?

Assumption 2 - Heteroscedasticity: constant variance for any value of $\mathbb{E}(y|x) = \beta \cdot x$



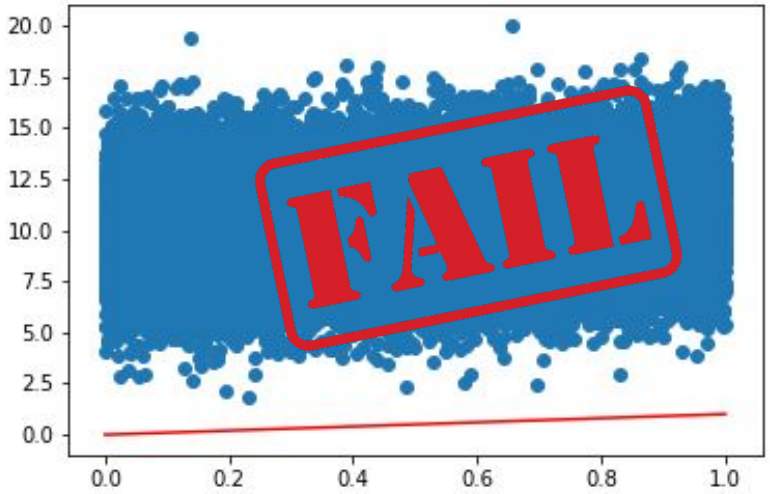
What assumptions does this imply?

Assumption 3 - The residuals are all normally distributed with zero mean.



What assumptions does this imply?

Assumption 3 - The residuals are all normally distributed with zero mean.



Computing the confidence in coefficients

Properties of summing random variables

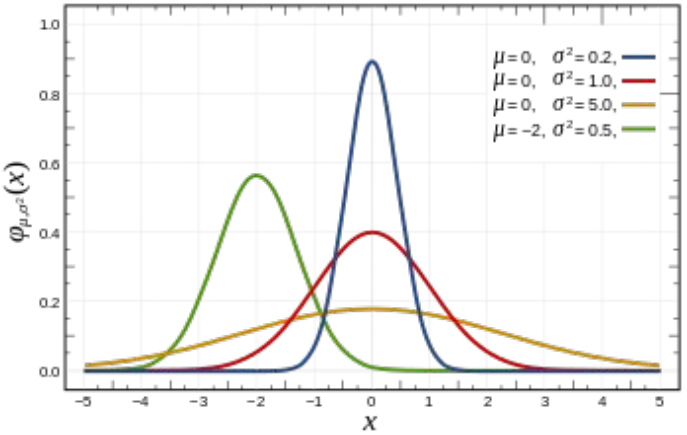
$$X \sim \mathcal{N}(\mu_X, \sigma_X^2)$$

$$Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$$

$$Z = X + Y$$

If X and Y are **independent,** then:

$$Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + {}_Y^2)$$



Intuition of Reason

- The number of heads when flipping a coin many times is (approximately) normally distributed.
- If you flip **two coins** N times, it's the same as flipping one coin with the average mean of the two coins 2N times (ie. another coin!), which is also normally distributed.

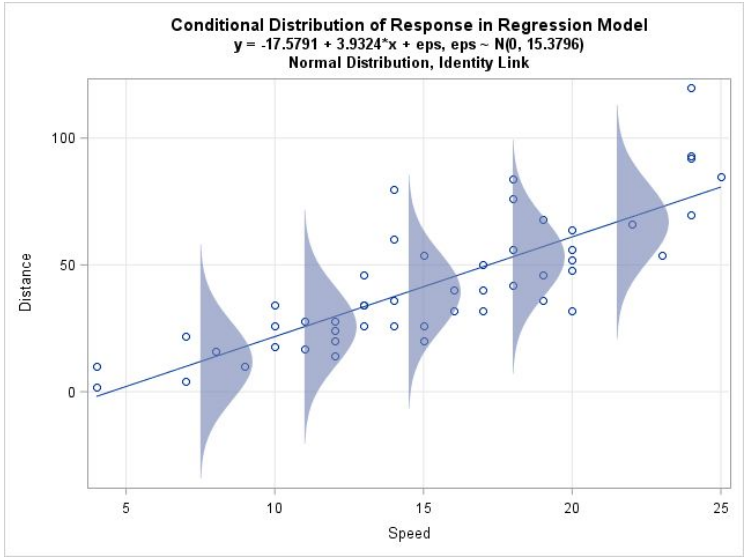
Distribution of slope estimator

$$y_i = \beta \cdot x_i + \epsilon_i$$

$$\sum_i \beta \cdot x_i^2 \sim \mathcal{N}\left(\sum_i y_i \cdot x_i, \sum_i x_i^2\right)$$

$$\beta \sim \mathcal{N}\left(\frac{\sum_i y_i \cdot x_i}{\sum_i x_i^2}, (\sum_i x_i^2)^{-1}\right)$$

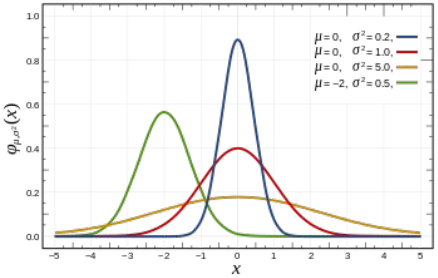
$$\beta \sim \mathcal{N}\left(\hat{\beta}, (\sum_i x_i^2)^{-1}\right)$$



This variance gives us a confidence bound on our estimate of the coefficient.

Confidence of coefficients

$$y_i = \beta \cdot x_i + \epsilon_i$$



ϵ_i is normally distributed and independent.

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

$$\hat{\beta}_1 = \frac{\sum_i x_i y_i}{\sum x_i^2}$$

$$= \frac{\sum x_i(\beta_1 x_i + \epsilon)}{\sum_i x_i^2}$$

$$= \beta_1 + \frac{\sum \epsilon_i x_i}{\sum_i x_i^2}$$

$$\frac{\sum_i \epsilon_i x_i}{\sum x_i^2} \sim \mathcal{N}\left(0, \frac{\sigma^2}{\sum x_i^2}\right)$$

$$\mathcal{N}(0, (X^T X)^{-1}\sigma^2)$$

This is how you measure p values of regression coefficients.

Confidence of coefficients

We assumed that our error was normally distributed, and we obtained the coefficients were centered around our estimator with distribution:

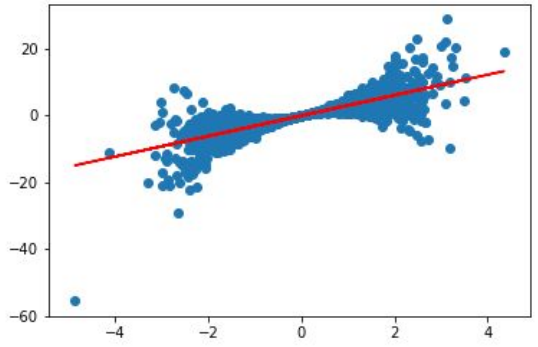
$$\beta \sim \mathcal{N}(\hat{\beta}, \sigma^2(X^T X)^{-1})$$

We can actually make a similar statement for errors from any kind of distribution (not just normal):

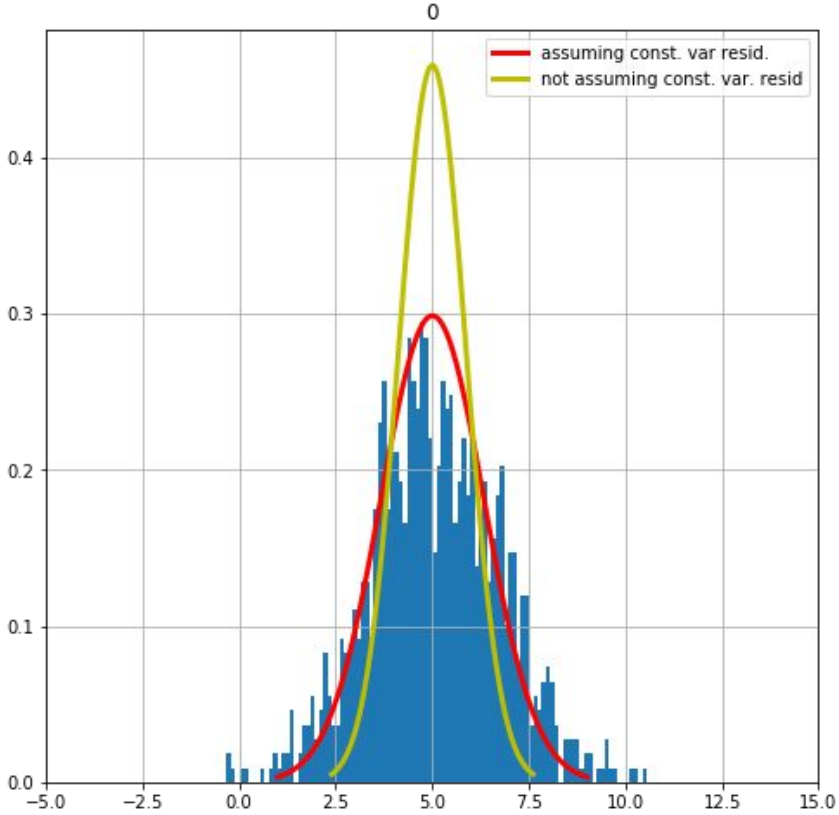
$$\beta \sim \mathcal{N}(\hat{\beta}, (X^T W X)^{-1})$$

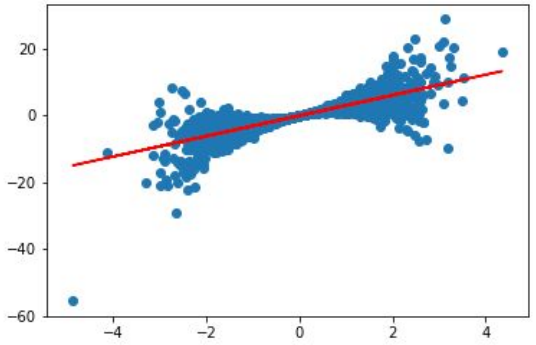
$$W_{ii} := \frac{1}{\sigma_i^2}$$

An even more general result holds, but we will save this for later.

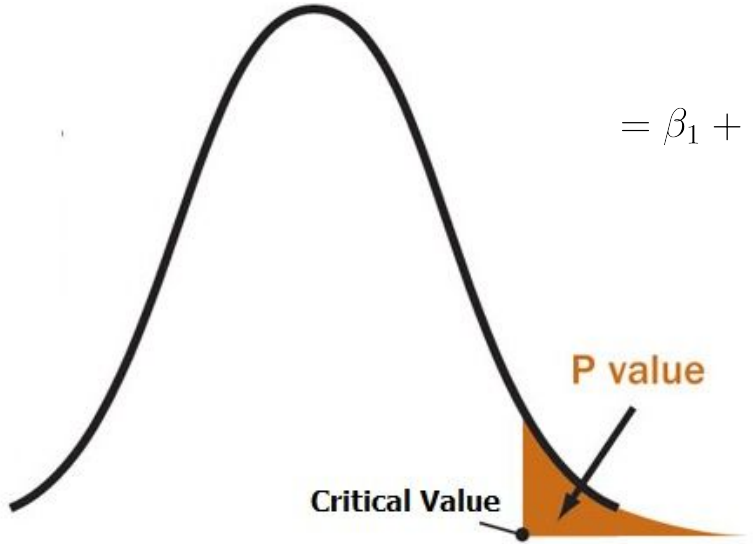


Confidence of coefficients





Confidence via p values



$$= \beta_1 + \frac{\sum \epsilon_i x_i}{\sum_i x_i^2}$$

$$\frac{\sum_i \epsilon_i x_i}{\sum x_i^2} \sim \mathcal{N}\left(0, (X^T X)^{-1}\sigma^2\right)$$

$$t = \frac{\beta_1 - 0}{\mathrm{SE}(\beta_1)}$$

Number of standard deviations away from 0 - how rare is it to observe this value under the hypothesis H0?

$$\mathrm{SE}(\beta_1)^2 \sim \frac{1}{N} \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (x_i - \bar{x})^2}$$

H0: The coefficient is zero and has no effect on y.

H1: The coefficient is non-zero and has an impact on y.

This is an example of **Hypothesis Testing**

Advertisement Example

In [208]:
```
beta=np.linalg.inv(X.T.dot(X)).dot(X.T.dot(y))

yhat = X.dot(beta).values+np.mean(y)
varY=np.sum([(yvals[i] - yhat[i])**2 for i in range(len(y))])/len(y)

cov = np.linalg.inv(X.T.dot(X))*varY
pd.DataFrame(cov)
```

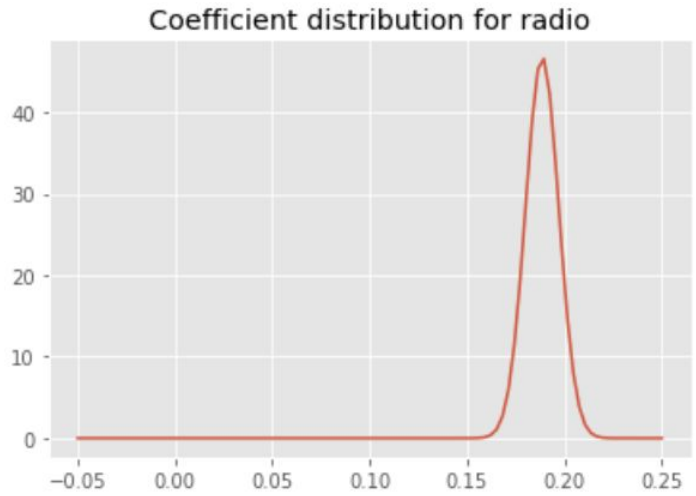
Out[208]:

	0	1	2
0	1.906822e-06	-4.380987e-07	-3.200631e-07
1	-4.380987e-07	7.267028e-05	-1.744461e-05
2	-3.200631e-07	-1.744461e-05	3.377938e-05

$$\hat{\sigma}^2(X^T X)^{-1}$$

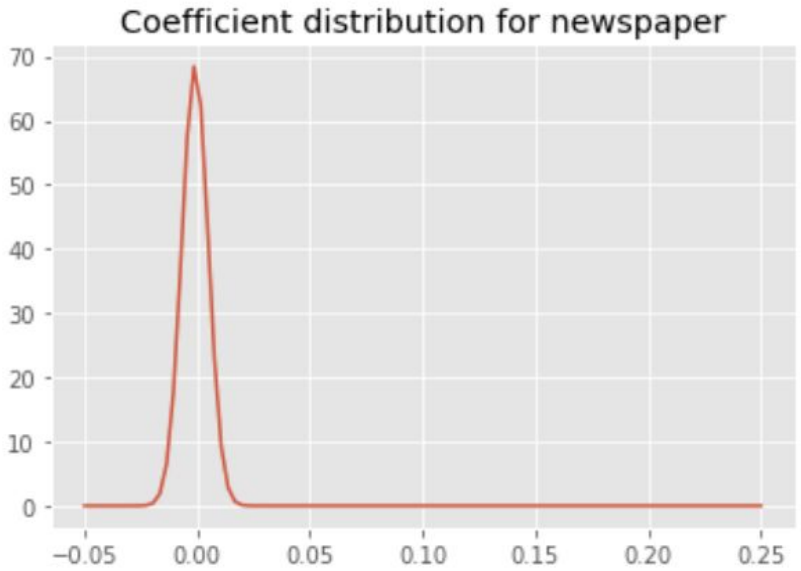
Advertisement Example

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.mlab as mlab
import scipy
import math
names = X2.columns.values
for i in range(3):
    mu = beta[i]
    variance = cov[i,i]
    sigma = math.sqrt(variance)
    x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
    x = np.linspace(-0.05,0.25,100)
    plt.title("Coefficient distribution for {0}".format(names[i]))
    plt.plot(x, scipy.stats.norm.pdf(x, mu, sigma))
    plt.show()
```



Advertisement Example

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.mlab as mlab
import scipy
import math
names = X2.columns.values
for i in range(3):
    mu = beta[i]
    variance = cov[i,i]
    sigma = math.sqrt(variance)
    x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
    x = np.linspace(-0.05,0.25,100)
    plt.title("Coefficient distribution for {0}".format(names[i]))
    plt.plot(x, scipy.stats.norm.pdf(x, mu, sigma))
    plt.show()
```



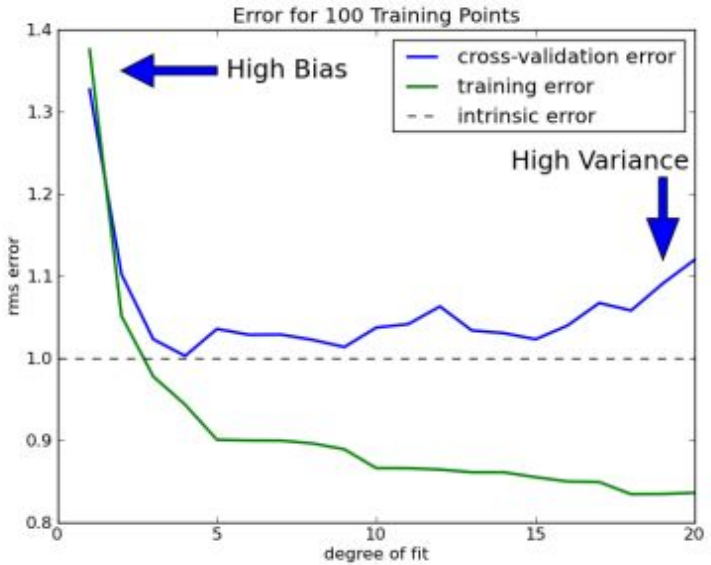
The coefficient for newspaper has about a 50/50 chance of being positive.

Eliminating bad variables?

- 1) For building predictive models, the best strategy is to **optimize for performance by restricting the number of coefficients by penalizing them somehow.**

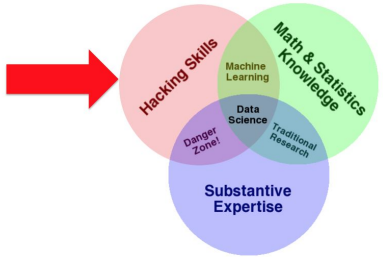
- 2) **If you don't have a lot of data**, **use many fold cross validation** to avoid the chance of having a spurious model.

  *Both will be covered in detail **next lecture.** *



Quick Summary

- Derived the analytical solution for L2 linear regression.
- Discussed prediction vs inference. For inference, our assumptions on the residuals must be correct. For prediction, it is not needed a priori.
- Derived confidence bounds for our coefficients.
- Machine learning is about finding the best fit of models to make prediction. Inference is about making conclusior the impact of a collection of variables on another.

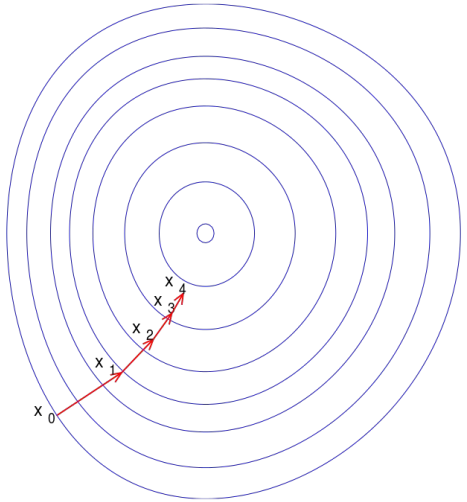


Gradient Descent

How do we minimize a function when there is no analytical solution?

We can’t always solve analytically!

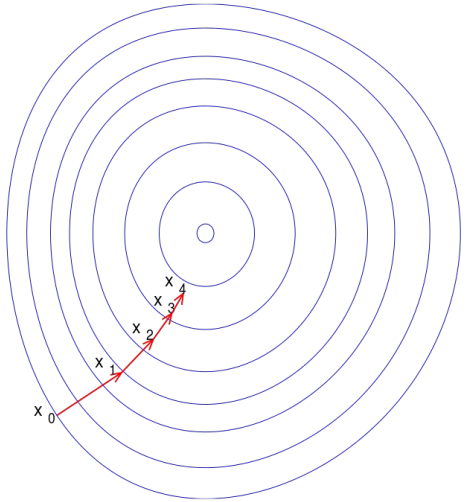
$$\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^2$$



- In these cases, we use the method of gradient descent.
- Almost all other models don’t have any explicit, analytical solution, so we have to use gradient descent.

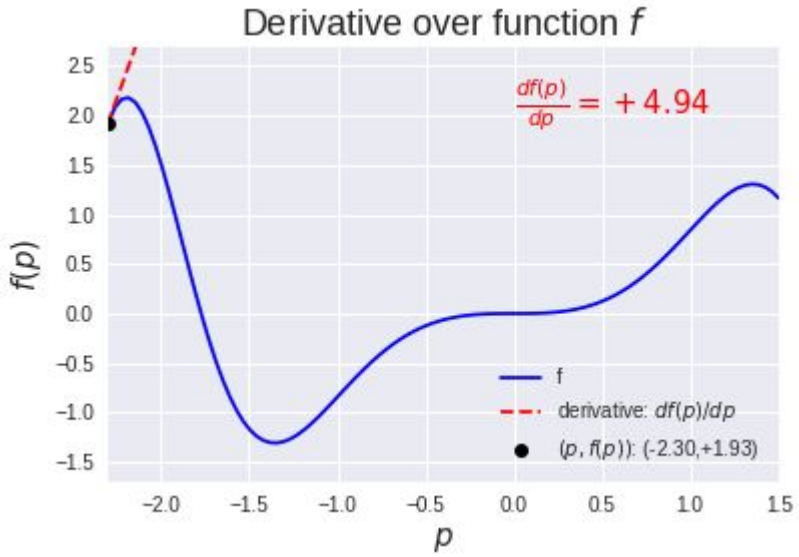
We can’t always solve analytically!

$$\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^{4?}$$

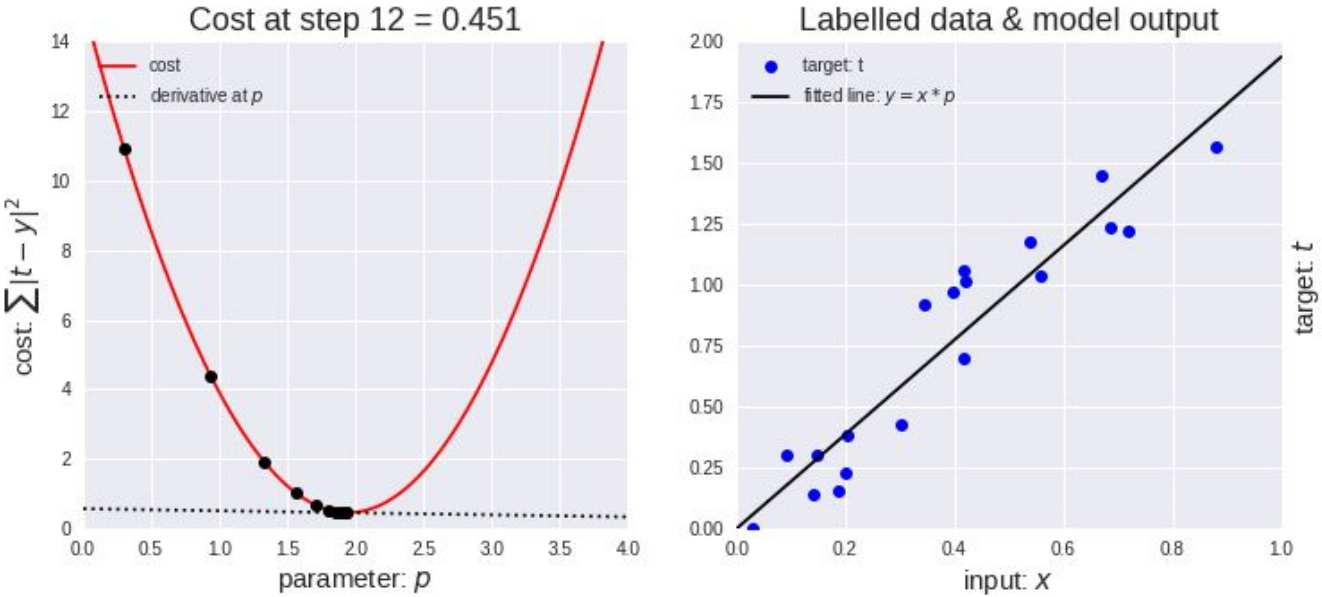


- In these cases, we use the method of gradient descent.
- Almost all other models don’t have any explicit, analytical solution, so we have to use gradient descent.

Illustration



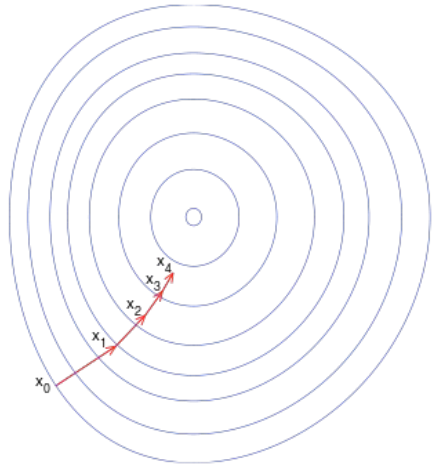
Illustration



Continuous Gradient Descent

Assume that $\beta(t)$ solves the equation:

$$\dot{\beta}(t) = -\nabla \mathcal{L}(\beta(t))$$



And that: $\beta \mapsto \mathcal{L}(\beta)$

- Is strictly convex.
- Twice differentiable.

Then: $\beta(t)$ converges to the minimum of f exponentially fast.

Proof of Convergence of Gradient Descent

Differentiating and using the gradient flow, we have

$$\frac{d}{dt}(\beta(t) - \beta_0)^2 = -2\nabla\mathcal{L}(\beta(t))(\beta(t) - \beta_0)$$

Next: How do we use strict convexity to obtain an estimate?

***Note*:** In reality we alway shave a discrete version of the above, and must choose a ‘learning rate’ (ie. time step size) - we will gloss over this for now.

Proof of Convergence of Gradient Descent

Using Taylor's Law we have

$$\mathcal{L}(\beta) - \mathcal{L}(\beta_0) = \nabla\mathcal{L}(\beta) \cdot (\beta - \beta_0) + \frac{1}{2}(\beta - \beta_0)^T D^2\mathcal{L}(\xi)(\beta - \beta_0)$$

$$\mathcal{L}(\beta_0) - \mathcal{L}(\beta) = \nabla\mathcal{L}(\beta_0) \cdot (\beta_0 - \beta) + \frac{1}{2}(\beta - \beta_0)^T D^2\mathcal{L}(\tilde{\xi})(\beta - \beta_0)$$

$$\frac{d}{dt}(\beta(t) - \beta_0)^2 = -2\nabla\mathcal{L}(\beta(t))(\beta(t) - \beta_0)$$

$$\frac{d}{dt}(\beta - \beta_0)^2 \leq -m|\beta - \beta_0|^2$$

$$\Rightarrow |\beta(t) - \beta_0)| \leq Ce^{-mt}$$

Therefore:

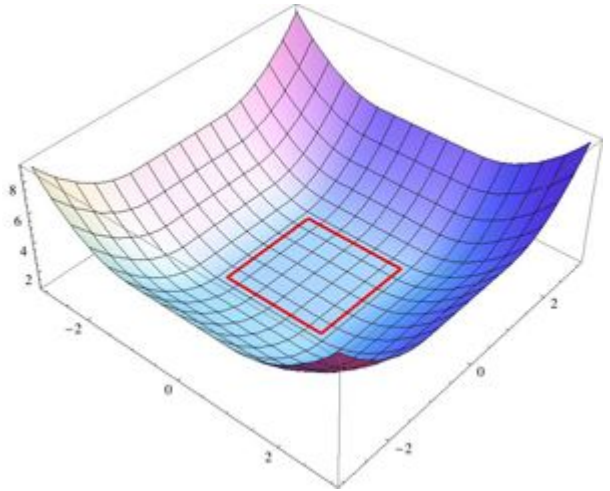
$$-\nabla\mathcal{L}(\beta - \beta_0) = -(\beta - \beta_0)^T(D^2\mathcal{L}(\xi) + D^2\mathcal{L}(\tilde{\xi}))(\beta - \beta_0) \leq -M|\beta - \beta_0|^2$$

Proof of Convergence of Gradient Descent

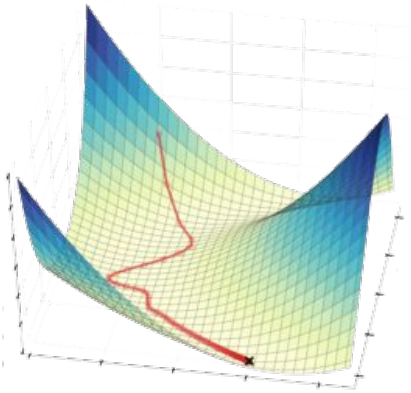
$$-\nabla\mathcal{L}(\beta-\beta_0) = -(\beta-\beta_0)^T(D^2\mathcal{L}(\xi)+D^2\mathcal{L}(\tilde{\xi}))(\beta-\beta_0) \leq -M|\beta-\beta_0|^2$$

$$\Rightarrow |\beta(t)-\beta_0)| \leq Ce^{-mt}$$

Stability is directly related to how large the eigenvalues are of the correlation matrix!



But wait! Did we miss something?



- When we perform gradient descent, we **only have our training samples to actually learn from!**
- This means having sufficient data is crucial to finding the right path.
- Imagine you had to find the bottom of a hill, but could only see some very sparse snapshot of the landscape. It would be hard!

$(\mathbf{x}_i, y_i)$ Training sample

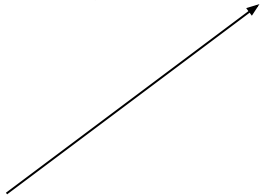
$\mathbf{x}_i$ Training of x only

$\mathbf{x}$ Particular fixed value of x

How is it computed in python?

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \beta \cdot x_i)x_j$$

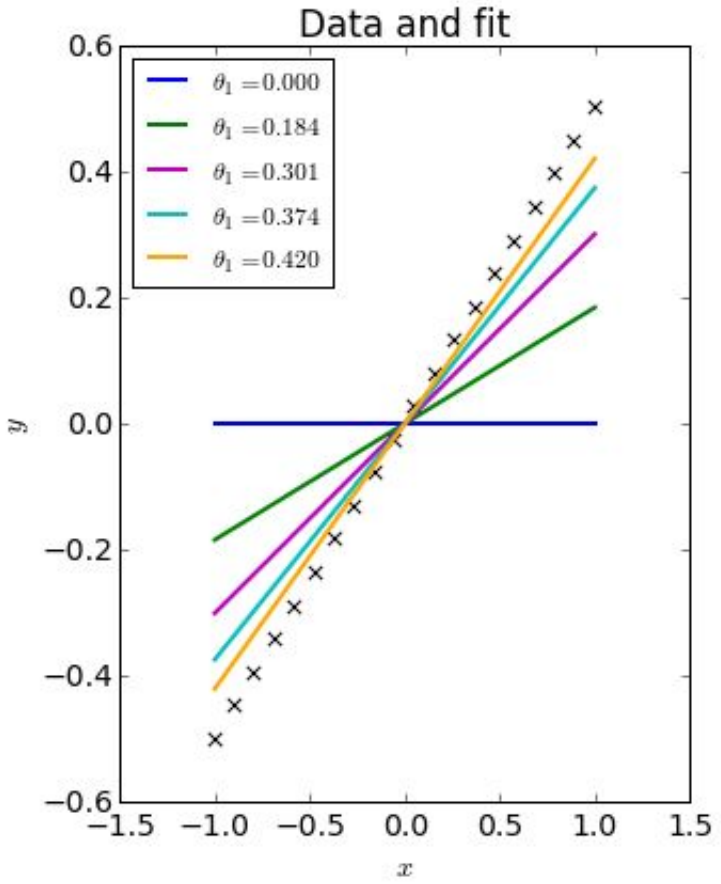
$$\beta_n = \beta_{n-1} - \kappa \nabla_\beta \mathcal{L}(\beta_{n-1})$$

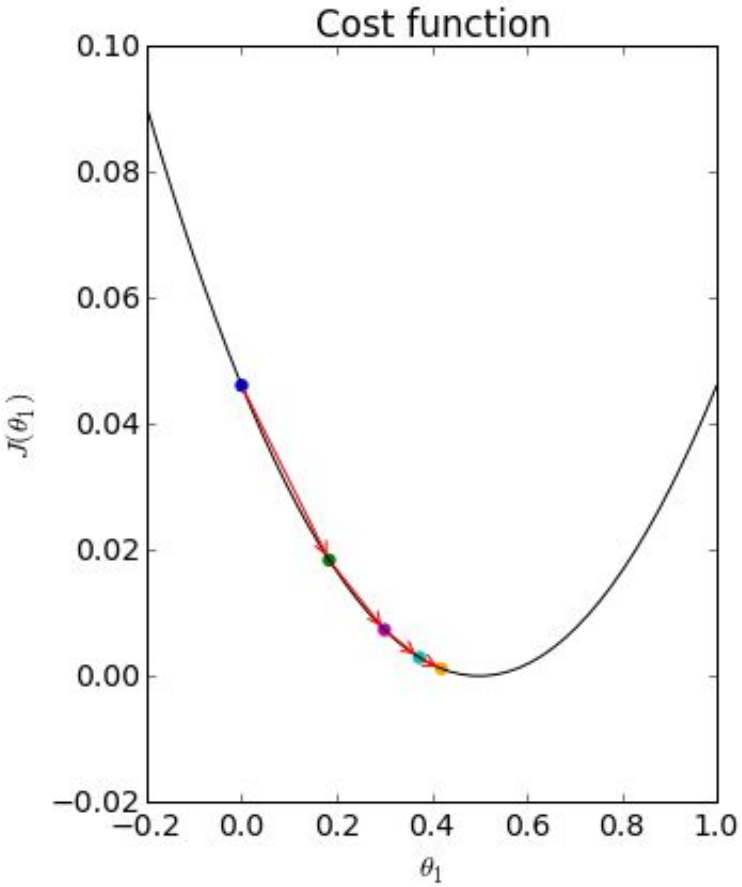


κ is known as the learning rate - when computing we must choose a time step size - Homework 1 covers this.

```python
def step_gradient(b_current, m_current, points, learningRate):
    b_gradient = 0
    m_gradient = 0
    N = float(len(points))
    for i in range(0, len(points)):
        x = points[i, 0]
        y = points[i, 1]
        b_gradient += -(2/N) * (y - ((m_current * x) + b_current))
        m_gradient += -(2/N) * x * (y - ((m_current * x) + b_current))
    new_b = b_current - (learningRate * b_gradient)
    new_m = m_current - (learningRate * m_gradient)
    return [new_b, new_m]
```

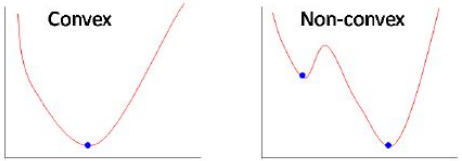
But if the learning rate is too big then, gradient descent may overshoot the minimum and oscillate back and forth. Why?


Data and fit
$\theta_1 = 0.000$
$\theta_1 = 0.184$
$\theta_1 = 0.301$
$\theta_1 = 0.374$
$\theta_1 = 0.420$
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x
y

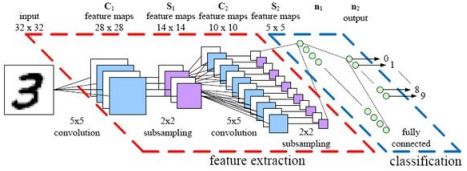

Cost function
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
θ_1
$J(\theta_1)$

The importance of gradient descent.

- All **parametric problems** in predictive machine learning seek to **minimize the distance of some a priori function of the data to the observed values.**
- It only makes sense to try to **minimize functions** which are **convex** (at least locally). Otherwise it's common to get stuck in local minima.
- When there is no analytical solution, the solution must be obtained by following the **steepest descent from a starting point to the minimum of the function.**
- This will be true when we work with more advanced probabilistic methods as well, and more important to understand.
- Stability is directly related to the eigenvalues of the covariance matrix, and also depends on how we choose our learning rate.

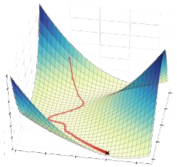


Machine Learning vs Inference



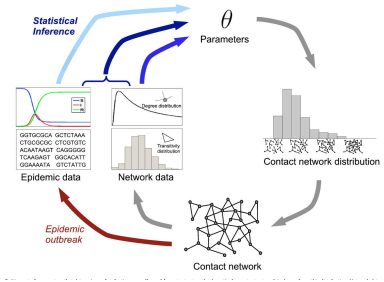
Machine Learning:

- Concerned primarily with computation at this point. Complex models need many systems in parallel trying to find parameters in highly complex non-linear spaces of parameters. The goal is not inducing causal impact but rather being ok with the model being a black box



Inference:

- Concerned with understanding relationship between variables and causal impact. Inferring “true” parameters from data and quantifying this uncertainty.



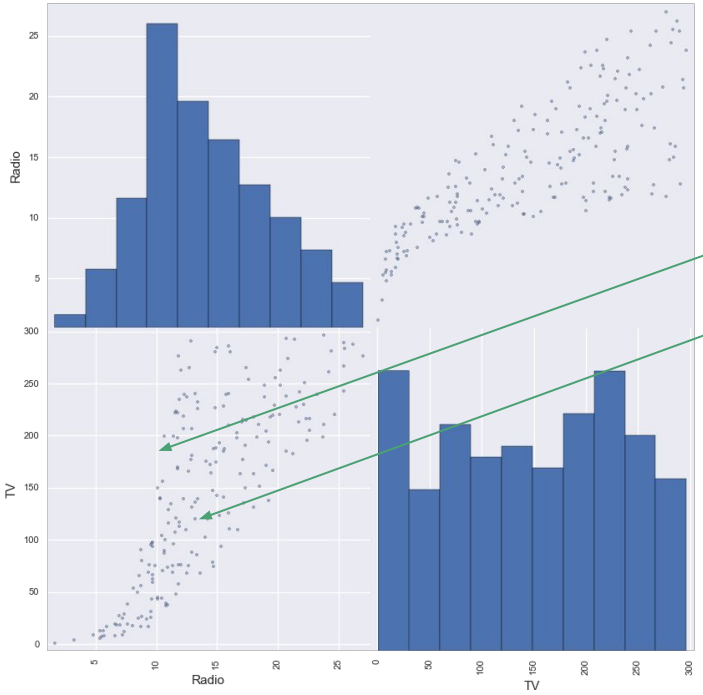
Homework Comments

- HW1 deadline extended until next week.
- Updated code for 3d plot is given.
- Part 6e) is replaced with a problem that we will go over now.
- Estimator for variance is biased but this rarely ever matters. (n-1 vs n)
- np.mean(y) appears in lecture notes when computing estimator for yhat since we didn’t put X[‘const’]=1.

Understanding stability and correlation

Understanding correlation

Note: This is not the same as the data you will do for the homework!



$$\mathbb{R}^{2\times n} \times \mathbb{R}^{n\times 2}$$

$$X^T X = \begin{bmatrix} 150 & 160 & \cdots \\ 17 & 12 & \cdots \end{bmatrix} \begin{bmatrix} 150 & 17 \\ 160 & 12 \\ \cdots & \cdots \end{bmatrix}$$

$$\mathbb{R}^{2\times 2}$$

$$[X^T X]_{ij} = \mathbf{x_i} \cdot \mathbf{x}_j$$

$$i = \text{Radio}$$

$$j = \text{TV}$$

Vectors are in n dimensions (number of data points)

$$[\text{Corr}]_{ij} = \frac{\mathbf{x_i} \cdot \mathbf{x}_j}{\|\mathbf{x_i}\|\|\mathbf{x_j}\|} = \cos(\delta_{ij})$$

Eigenvalues of Covariance Matrix

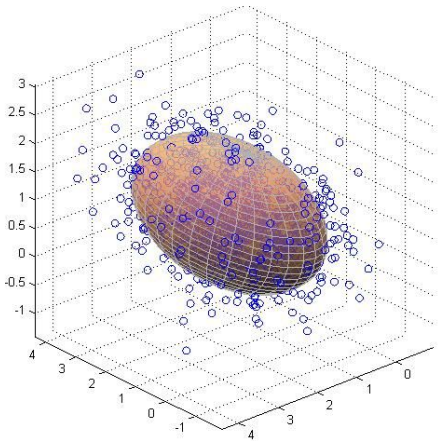
$$\lambda_1 = \max_{\|y\|=1} y^T A y$$

$$y^T A y = \langle y, Ay \rangle$$

$$y = v + \epsilon w$$

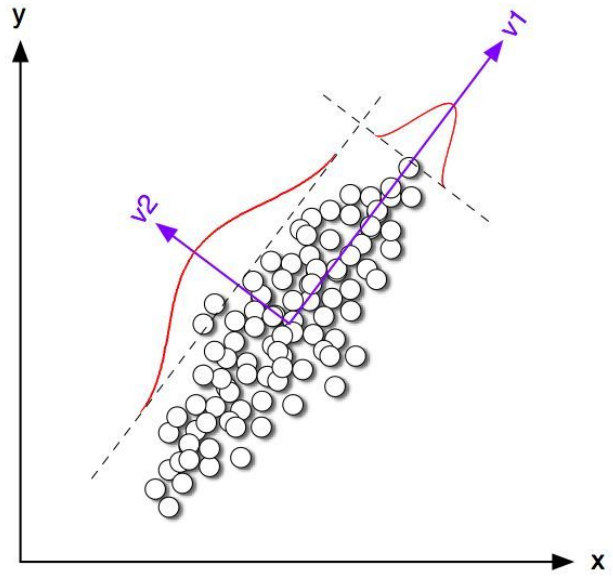
$$\nabla_\epsilon (y^T A y)\Big|_{\epsilon=0} = \langle w, Av \rangle + \langle v, Aw \rangle = \lambda \langle v, w \rangle$$

$$2\langle Av, w \rangle = \lambda \langle v, w \rangle \longrightarrow Av = \tilde{\lambda} v$$



The others are obtained by maximizing in the orthogonal complement to the vector v.

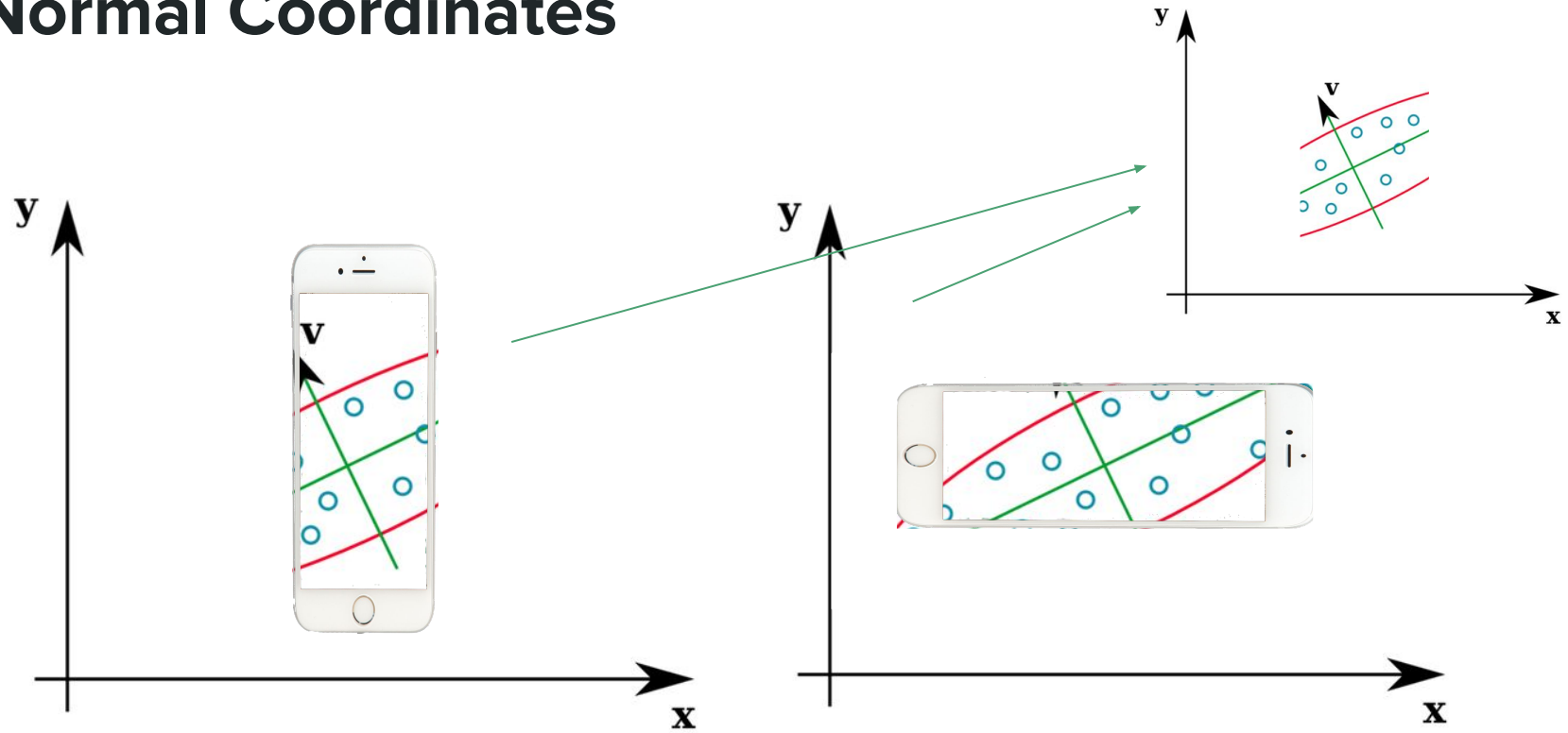
Eigenvalues and Instability



- Strongly correlated variables result in an orthogonal direction with a small eigenvalue.
- This creates the instability we have been discussing.

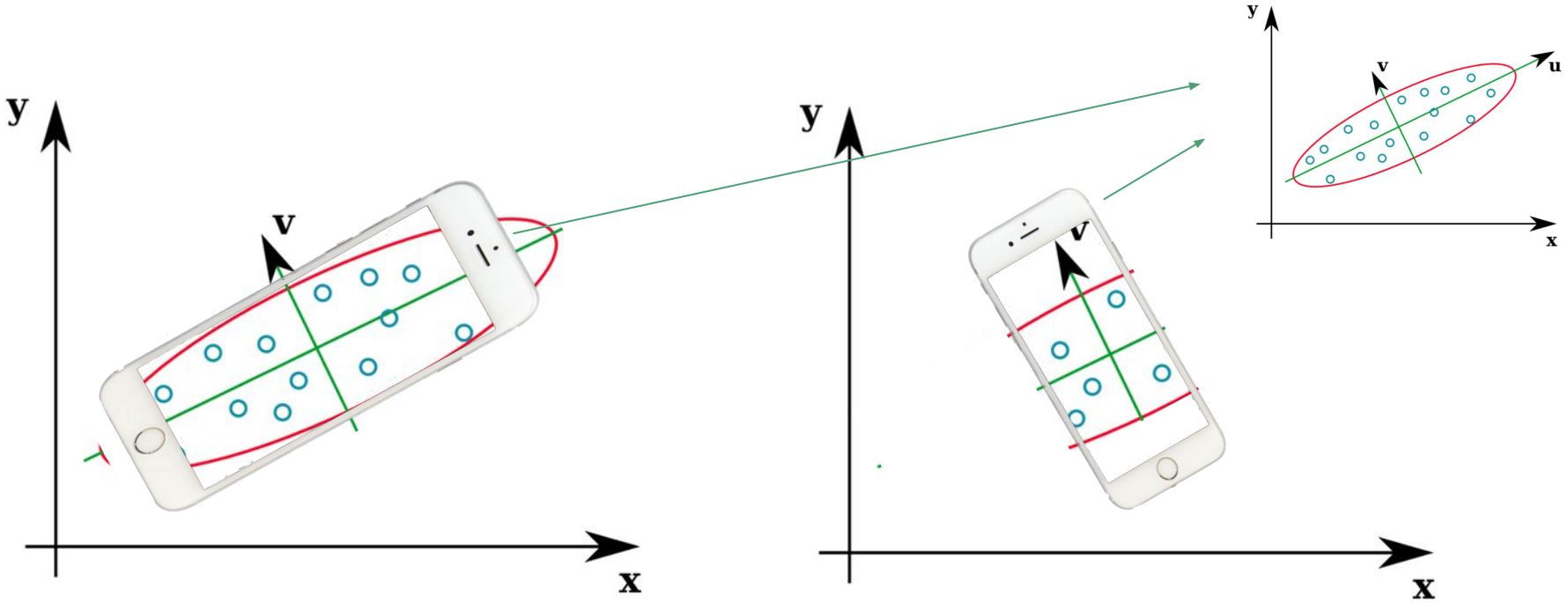
$$Av = \tilde{\lambda}v$$

Normal Coordinates



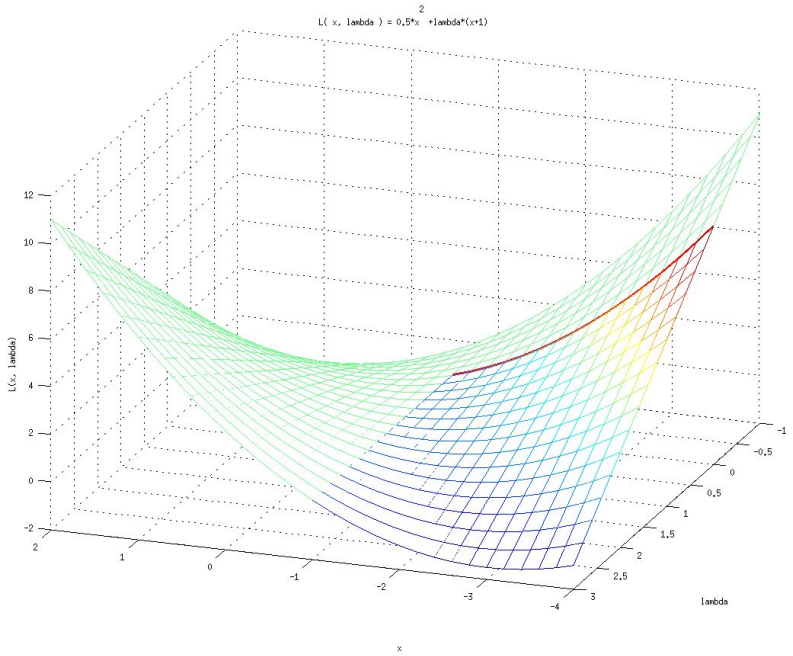
Imagine you can only take a photo of these points in two directions - which ones would you choose?

Eigenvectors



By rotating our camera to “good” angles, we capture more about the distribution of the data! These are the eigenvectors (ie. PCA).

Correlation results in instability

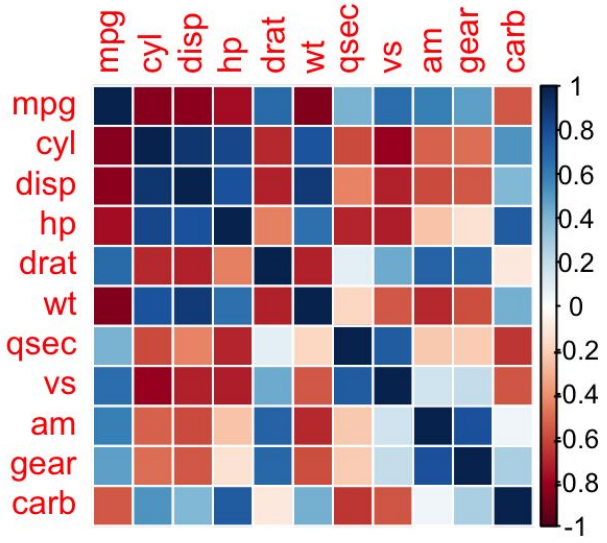


- Flat directions in the feature space mean less stability and more uncertainty about what the true coefficients are!
- When we see gradient descent, the parameters involved can significantly affect solutions when there is no stability.

$$\frac{d^2\mathcal{L}}{d^2\beta} = \frac{2}{N}X^T X$$

Correlation results in instability

- $X^T X$ is **symmetric** and therefore has **nonnegative eigenvalues.**
- They are **positive** precisely when the features of X are **linearly independent.**
- Let's assume for simplicity that **X is mean centered (fine but sometimes reasons why you might not).**



Imagine that X has two columns which are factors of one another. What can go wrong?

$$y = \alpha x_1 + \beta x_2$$

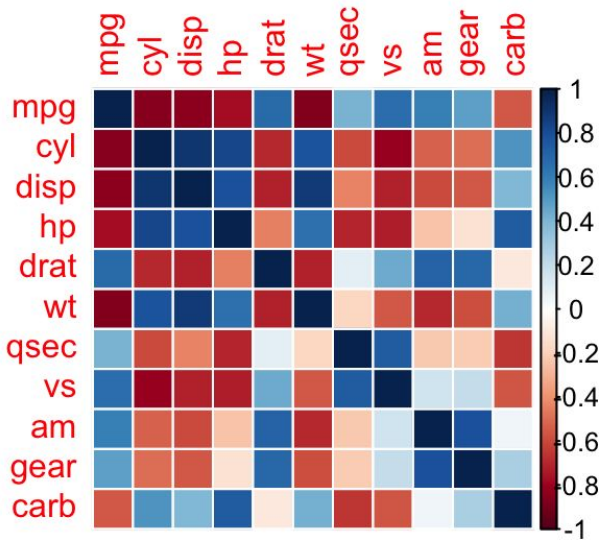
But our real rule is: $y = 5x_1$

Correlation results in instability

$$y = \alpha x_1 + \beta x_2$$

But our real rule is: $y = 5x_1$

Then $y = -1000x_1 + 10005x_2$ is also a solution



Why is this a problem?

Correlation results in instability

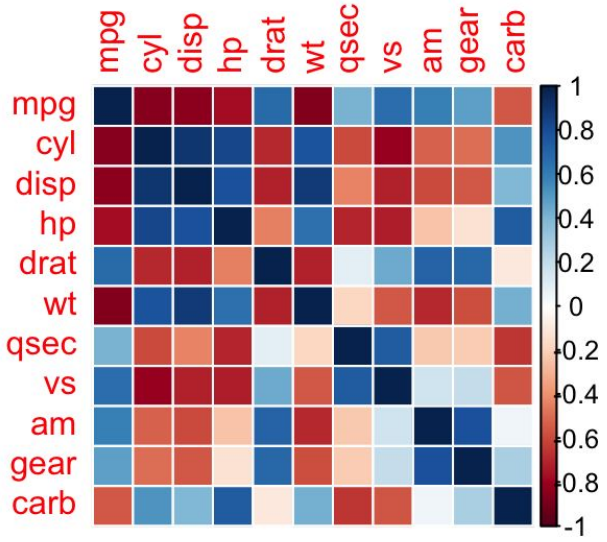
$$y = \alpha x_1 + \beta x_2$$

But our real rule is: $y = 5x_1$

Then $y = -1000x_1 + 10005x_2$ is also a solution

Why is this a problem?

- Creates problems when finding the minimum (next).
- Causes more uncertainty in coefficient estimates (after gradient descent).



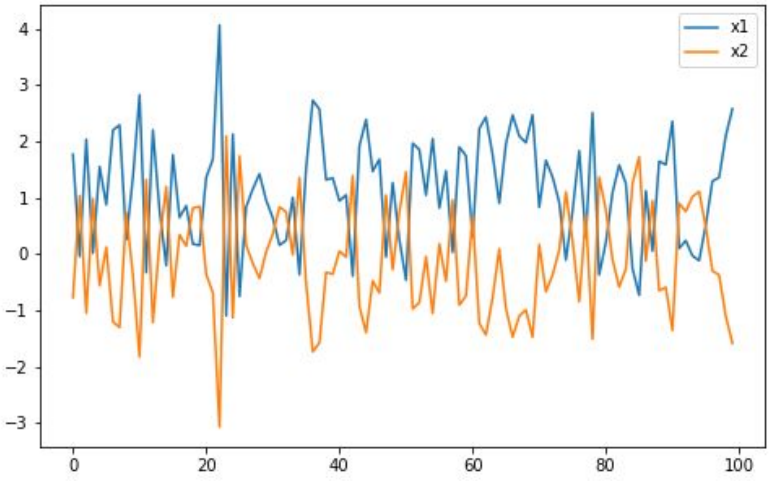
A simple illustration - correlated

$$y = x_1 + \epsilon$$

$$x_2 = 100 * x_1 + \epsilon$$

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

```
n=10000
x1 = np.linspace(0,0.01,n)


s = np.random.normal(0, 0.001, n)

x2 = 100*x1 + s

df=pd.DataFrame({'x1':x1,'x2':x2})

y = x1 + np.random.normal(0, 0.01, n)
coefs1=[]
coefs2=[]
scores_perp=[]
for i in range(0,100):
    y = x1 + np.random.normal(0, 0.001, n)
    regr = linear_model.LinearRegression()
    x=df
    # Train the model
    regr.fit(x,y)
    coefs1.append(regr.coef_[0])
    coefs2.append(regr.coef_[1]*100)
    scores_perp.append(regr.score(x,y))

plt.figure(figsize=(8,5))
plt.plot(coefs1,label='x1')
plt.plot(coefs2,label='x2')
plt.legend()
```



Number of iterations solving the same problem.

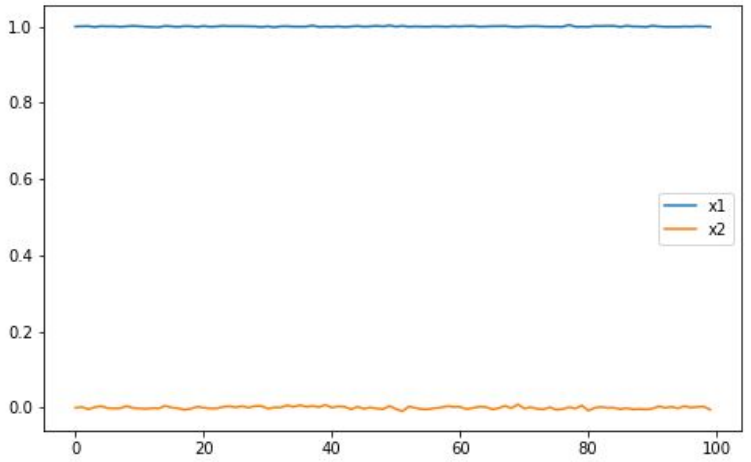
A simple illustration - orthogonal features

$$y = x_1 + \epsilon$$

$$x_1 \perp x_2$$

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2$$

```
k = np.random.normal(0,0.01,n)/np.linalg.norm(k)**2
s = np.random.normal(0, 0.001, n)

x2 = 100*np.linspace(0,0.01,n)
x2 -= x1.dot(k) * k / np.linalg.norm(k)**2
x1=k
df=pd.DataFrame({'x1':x1,'x2':x2})

y = x1 + np.random.normal(0, 0.01, n)
coefs1=[]
coefs2=[]
scores_perp=[]
for i in range(0,100):
    y = x1 + np.random.normal(0, 0.001, n)
    regr = linear_model.LinearRegression()
    x=df
    # Train the model
    regr.fit(x,y)
    coefs1.append(regr.coef_[0])
    coefs2.append(regr.coef_[1]*100)
    scores_perp.append(regr.score(x,y))

plt.figure(figsize=(8,5))
plt.plot(coefs1,label='x1')
plt.plot(coefs2,label='x2')
plt.legend()
plt.show()
```



Number of iterations solving the same problem.

Correlation and Stability

$$F(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|^2,$$

where $\beta = (\beta_1, \beta_2)$. $X \in \mathbb{R}^{n\times 2}$, $y \in \mathbb{R}^n$.

$$F(\beta) = F(\beta_0) + \nabla F(\beta_0) \cdot (\beta - \beta_0) + (\beta - \beta_0)^T D^2 F(\beta_0)(\beta - \beta_0) + O\left(|\beta - \beta_0|^3\right)$$

Let's assume our true rule is

$$y = 5X_0 + +10X_1 + \epsilon,$$

where $\mathbf{X} = [X_0, X_1]$ and $X_0, X_1 \in \mathbb{R}^n$. Let's see how convergence is affected when X_0 and X_1 are correlated.

Correlation and Stability

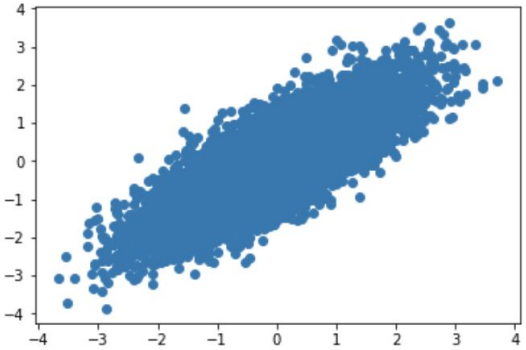
Let's assume our true rule is

$$y = 5X_0 + +10X_1 + \epsilon,$$

where $\mathbf{X} = [X_0, X_1]$ and $X_0, X_1 \in \mathbb{R}^n$. Let's see how convergence is affected when X_0 and X_1 are correlated.

```
In [4]: mean = (0,0)
        corr = 0.8
        cov = [[1, corr],[corr,1 ]]
        X = np.random.multivariate_normal(mean, cov, 10000)
        plt.scatter(X[:,0],X[:,1])
```

```
Out[4]: <matplotlib.collections.PathCollection at 0x1a22880950>
```

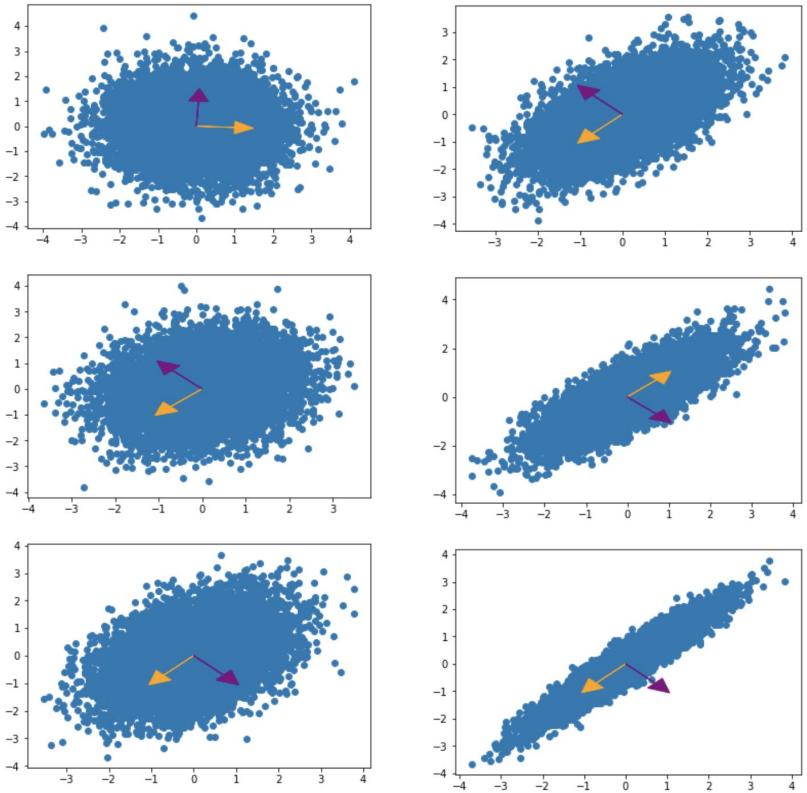


$$F(\beta) = F(\beta_0) + \nabla F(\beta_0) \cdot (\beta - \beta_0) + (\beta - \beta_0)^T D^2 F(\beta_0)(\beta - \beta_0) + O\left(|\beta - \beta_0|^3\right)$$

Correlation and Stability

```
:
import numpy as np
from sklearn.decomposition import PCA

mean = (0,0)
for corr in [0.0,0.2,0.4,0.6,0.8,0.95]:

    cov = [[1, corr],[corr,1 ]]
    X = np.random.multivariate_normal(mean, cov, 10000)

    pca = PCA(n_components=2)
    pca.fit(X)
    pca.components_
    v1,v2 = pca.components_
    plt.scatter(X[:,0],X[:,1])
    plt.arrow(0, 0, *v1, head_width=0.5, head_length=0.5,color='orange')
    plt.arrow(0, 0, *v2, head_width=0.5, head_length=0.5,color='purple')
    plt.show()
```

Correlation and Stability



Gradient Flow with Zero Correlation

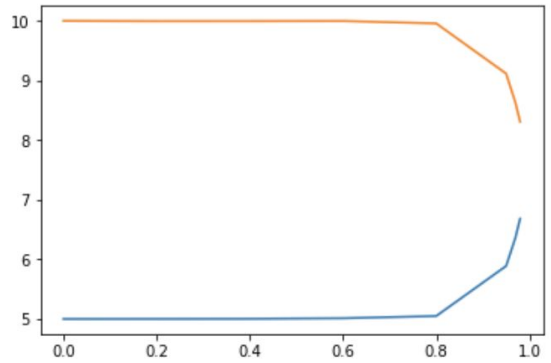
In [110]:
```
corr = 0.0
cov = [[1, corr],[corr,1 ]]
X = np.random.multivariate_normal(mean, cov, 10000)
y = 5*X[:,0] + 10*X[:,1] + np.random.normal(0,0.5,10000)

def grad(X, y, beta):
    df = 2*(-X.T.dot(y) + X.T.dot(X).dot(beta))
    return df
def gradflow(X,y,beta0,nu=0.00001):
    beta = beta0
    for k in range(100):
        beta = beta - nu*grad(X,y,beta)
    return beta
def gradflow_full(X,y,beta0,nu=0.00001):
    beta = beta0
    gradflow=[]
    for k in range(100):
        beta = beta - nu*grad(X,y,beta)
        gradflow.append(beta)
    return gradflow
gradflow(X,y,[3,3])
```

Out[110]:
```
array([5.00420042, 9.9867889 ])
```

Gradient Flow with Positive Correlation

In [112]:
```
betas = []
corrs=[0.0,0.2,0.4,0.6,0.8,0.95,0.97,0.98]
for corr in corrs:
    cov = [[1, corr],[corr,1 ]]
    X = np.random.multivariate_normal(mean, cov, 10000)
    y = 5*X[:,0] + 10*X[:,1] + np.random.normal(0,0.5,10000)
    beta_solution = gradflow(X,y,[3,3])
    betas.append(beta_solution)
plt.plot(corrs,betas)
```

Out[112]:
```
[<matplotlib.lines.Line2D at 0x1a30ebbb10>,
 <matplotlib.lines.Line2D at 0x1a30ebbc90>]
```

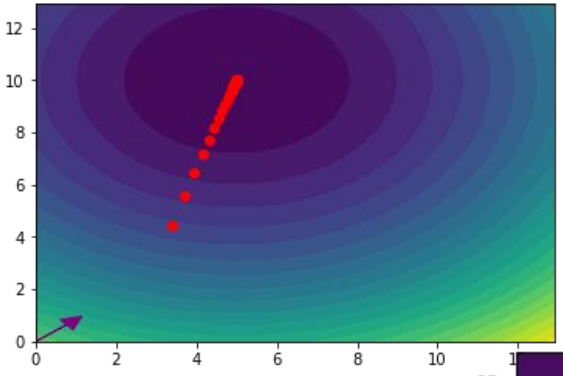


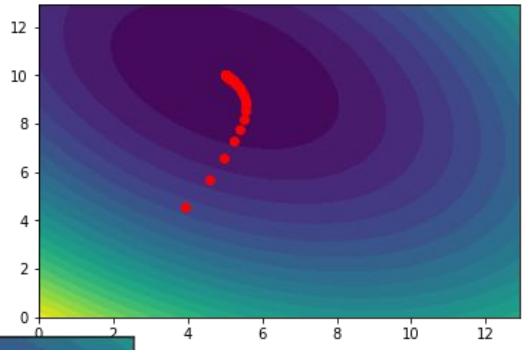
Gradient Flow with Positive Correlation

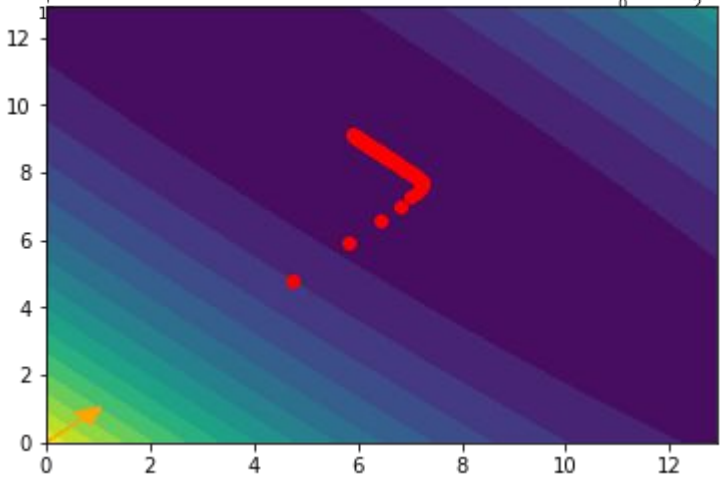
In [122]:

```
import numpy as np
from sklearn.decomposition import PCA

mean = (0,0)
for corr in [0.0,0.2,0.4,0.6,0.8,0.95,0.97]:

    cov = [[1, corr],[corr,1 ]]
    X = np.random.multivariate_normal(mean, cov, 10000)
    y = 5*X[:,0] + 10*X[:,1] + np.random.normal(0,0.5,10000)

    pca = PCA(n_components=2)
    pca.fit(X)
    pca.components_
    v1,v2 = pca.components_
    #plt.scatter(X[:,0],X[:,1])
    beta_solution = np.squeeze(gradflow_full(X,y,[3,3]))

    a = np.arange(0,13, 0.1)
    b = np.arange(0,13, 0.1)
    xx, yy = np.meshgrid(a,b, sparse=True)
    z =  ((xx-5)**2 +2*corr*(xx-5)*(yy-10)+ (yy-10)**2)
    h = plt.contourf(a,b,z,20)
    plt.scatter(beta_solution[:,0],beta_solution[:,1],color='r')
```

Gradient Flow with Positive Correlation







Eventually it gets stuck in the flat region!

End

Appendix follows (rough notes)

How to make predictions with K-Fold?

1. If we do K-fold cross validation, which coefficient do we choose? The mean, max? (Answer: None of them! That's not the point).

The **purpose k-fold cross validation is to measure the confidence of your model "procedure"**- ie. feature selection, model selection (linear regression, random forest, etc), regularization (hyperparameter selection).

It can also give you a sense of how wildly parameters change and thus a sense of **'confidence'** about your variables, with regards to generalizing to unseen data.

But ultimately you **verify your model with k-fold cross validation,** then **train on the entire data set** once you're ensured it generalizes to new examples!

More than one model? Do your best on the K folds with each model, and compare the total error over each fold:
More explanations:
https://stats.stackexchange.com/questions/52274/how-to-choose-a-predictive-model-after-k-fold-cross-validation

Should you always normalize your data?

- It depends a lot on your data and your model, but generally for linear models, **normalizing is not only safer, it is usually needed**

- When we penalize the size of error terms (regularization), we will need to normalize for this to make any sense.

- For tree-based models, it doesn't matter (we will see why later).

- **Long story short:** if your model is linear, you probably want to normalize unless you have a reason not to. Ie. is a 1cm change in height the same as a 1kg change in weight?

Types of standardization/normalization

Normalizing to mean zero and unit variance

- **Regression/Classification**
- **Clustering**, standardization may be especially crucial in order to compare similarities between features based on certain distance measures.
- **Principal Component Analysis**, where we usually prefer standardization over Min-Max scaling.
- Coefficient comparison when variables represent different categories.
- Helps with gradient descent convergence - why?

Min-Max Scaling

- Image processing, where pixel intensities have to be normalized to fit within a certain range (i.e., 0 to 255 for the RGB color range).
- I've honestly never seen real world examples other than this where this is done.

Conclusion: Always use standardization (mean zero and unit variance) unless you believe there is a strong reason not to.

Standardization or Normalization?

There are some definitions (which I may have mixed up in lecture potentially):

Normalization transforms your data into a range between 0 and 1.

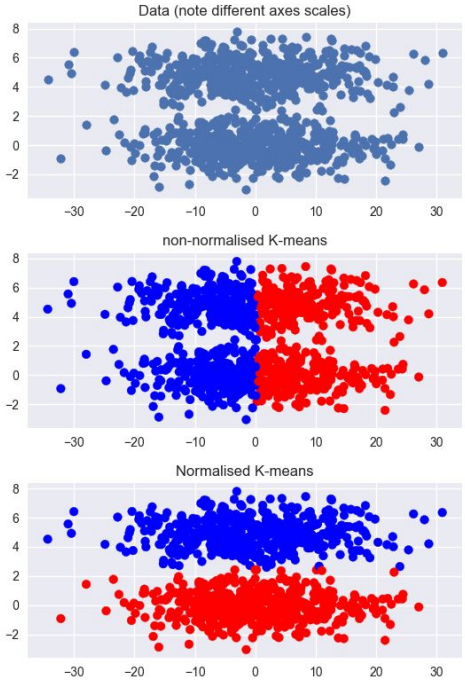
Standardization transforms your data such that the resulting distribution has a mean of 0 and a standard deviation of 1.

It's quite rare for people to use min/max normalization in regression problems in practice. In general people want to standardize so that the mean is 0 and the variance is 1. So you would have:

In general you care less about having your data all be within the same range, **but instead want measurements of change to be the same across all variables**. This is important when comparing coefficients, and when we will cover regularization, which puts bounds on the coefficients. There are indeed some exceptions though, such as when your data doesn't vary much but you want it to have the same range.

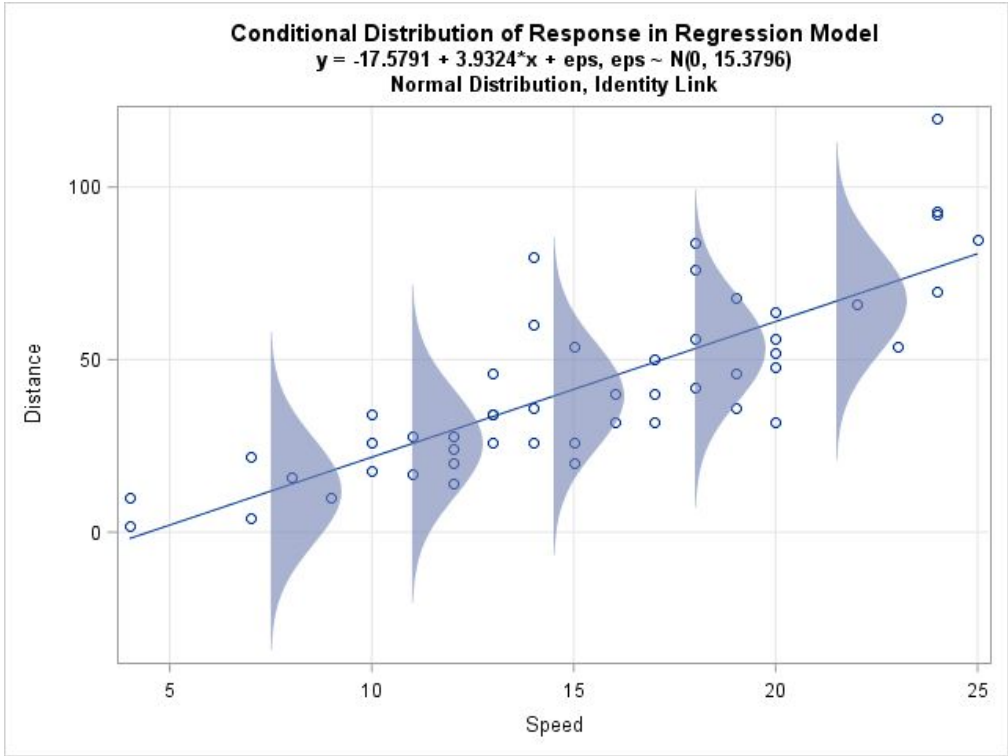
Taking max or min also isn't really stable for outliers (can you see why? Just one large point will mess things up).

Clustering Example



- Here we see that the plot above has a different range than the data below.
- When we normalize our data, we don't get the natural split we want.
- Although this is classification, the idea generalizes as well.

Errors are normally distributed



$$y_i = \beta \cdot x_i + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- We assume the errors for ordinary least squares are assumed to be normally distributed (we will see why later).
- This assumption will be justified by the central limit theorem.

Decision Tree Regression

First: A 1d example

$$y = \sin(x) + \text{noise}$$

- Decision trees make a recursive series of decisions which lead to an outcome, real valued or labeled (for regression, real valued)
- The goal is to make splitting decisions on the data to minimize a norm, in particular, the L2 distance to the mean on that subset of the data, also known as the **variance.**

First: A 1d example

$$y = \sin(x) + \text{noise}$$

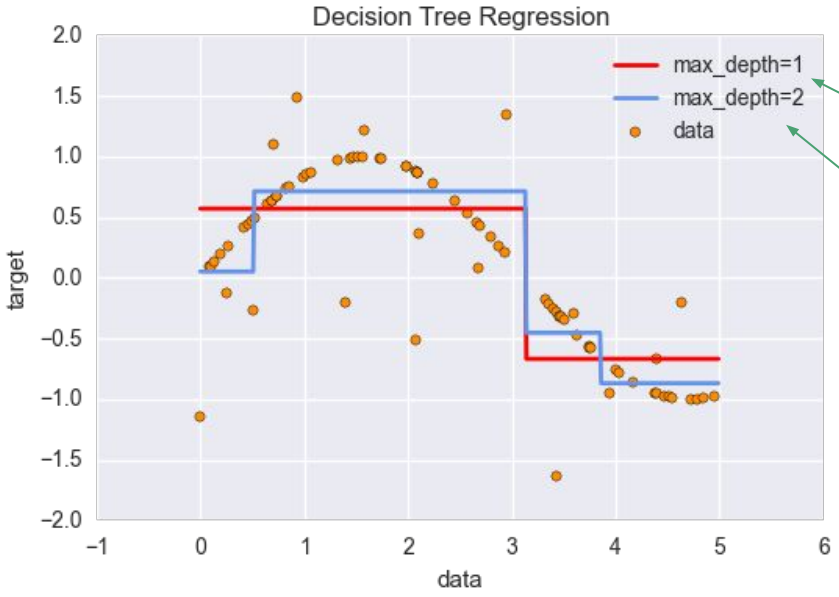
- For a depth of **zero**, one chooses the **mean**.
- How does one choose the splitting point when the depth is larger than zero?
- For a depth of one, the algorithm searches through all values between (-1,6) (in this example), and chooses the split which **minimizes the variance on the two segments which it creates.**

Increased depth can be bad

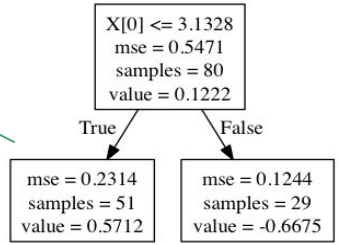
$$y = \sin(x) + \text{noise}$$

- In general, if one adds more depth to the tree, there is a risk of overfitting. Look at the unwanted variance we have picked up in this example.
- We will learn next lecture how to choose the perfect depth number!

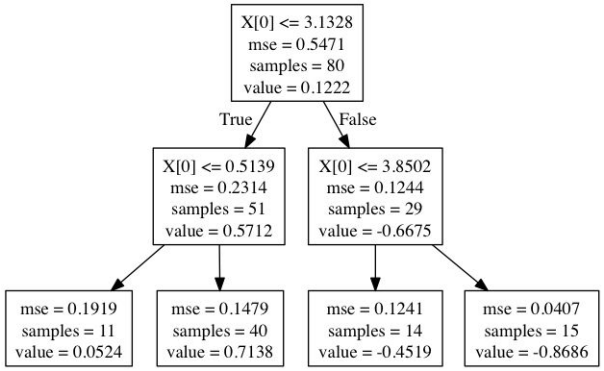
How do the decisions work?



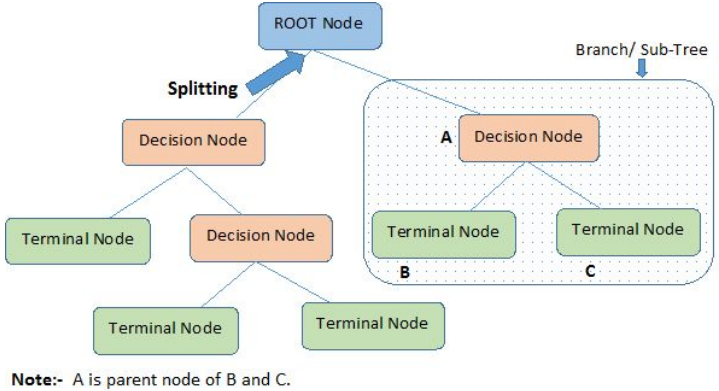
max_depth=1



max_depth=2

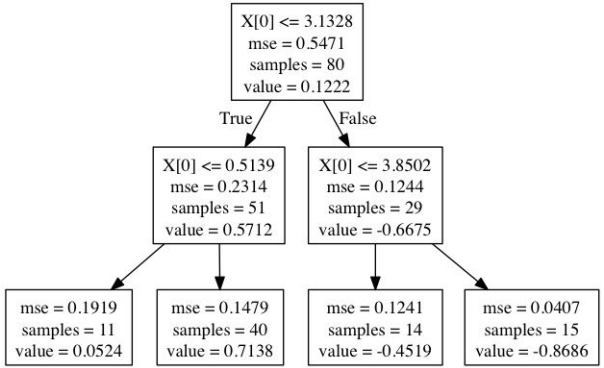


Some terminology

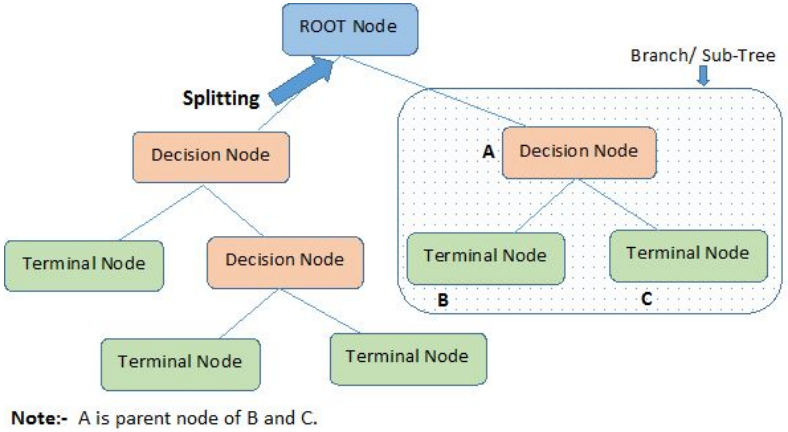


Note:- A is parent node of B and C.

max_depth=2



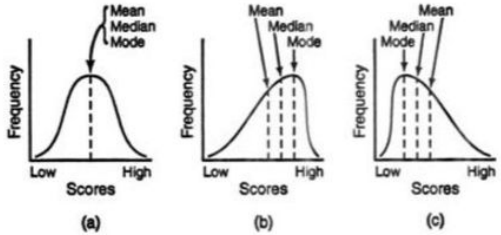
Some terminology



1. **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
6. **Branch / Sub-Tree:** A sub-section of entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.

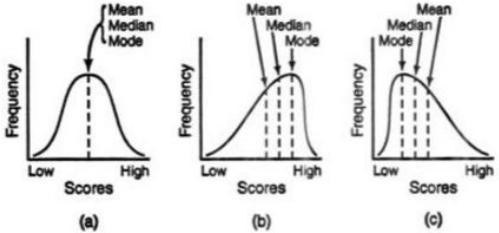
What are the advantages of decision trees?

- Very easy to interpret.
- Makes **no a-priori assumptions about the structural form of the data** (as linear models do). For instance, works well with non-linear data.
- More “robust” than linear models, meaning **less sensitive to outliers**. This is for the same reason that the median is less sensitive to outliers than the median.
- Well defined notion of ‘**most significant variables**’, so good for data exploration (will explain).
- Handles categorical and real-valued variables (doesn’t require one-hot encoding as linear models always do - Exercise: Why?)



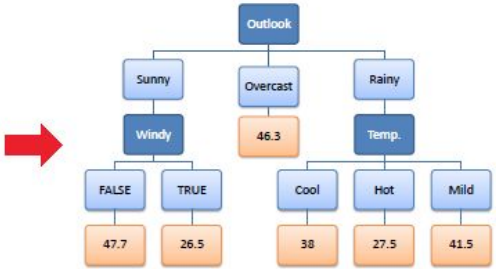
What are the disadvantages?

- Prone to **overfitting** (more on this next lecture).
- Can lose information when dealing with continuous variables, since it needs to make a finite number of splits.



More than one variable?

Predictors				Target
Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

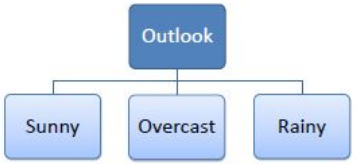


- A decision tree is simply a recursive set of decisions which leads to a result.
- It can be a real value or a class.
- Decision trees are very simple to understand, but are somewhat complex to explain when multiple features are involved.

Machine Learning Objective: Let's try to predict the number of hours played on a golf course in a single day by all of the golfers.

Example taken from: http://chem-eng.utoronto.ca/~datamining/dmc/decision_tree_reg.htm

How do we choose the root node?



Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

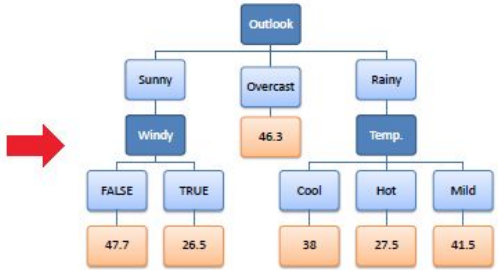
		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

- Here we've tried splitting first by the variable "Outlook".
- From here, we can compute the conditional standard deviations in the three subsets created.
- Our goal is to find the variable that, when split, reduces the stdev the most.
- Original stdev of "Hours Played" is 9.33.
- We can try this for every variable, and then choose the one which reduces the stdev the most.
- Subsequent decisions are made by recursively iterating this procedure.

More precise description

Predictors				Target
Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



For zero depth we minimize:

$$\mathrm{Var}(y) := \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

Solution is: $\hat{y} = \bar{y}$

$$\mathrm{Var}(Y|X = x_j) = \sum_{i=1}^{N}(y_i - \bar{y}_j)^2 p(y_i|x_j)$$

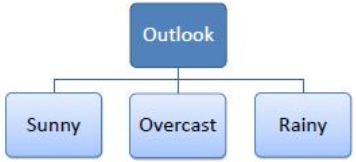
$$= \frac{1}{N}\sum_{i,x_i=x_j}(y_i - \bar{y}_j)^2 |X = x_j|$$

For depth one we minimize the conditional variance for each variable:

$$\mathrm{VarRed}(Y, X) = \mathrm{Var}(Y) - \sum_{j}\mathrm{Var}(Y|X = x_j)$$

Variance Reduction

Let's try Outlook first



Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

$$\mathrm{Var}(y) := \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 = 9.32^2$$

$$\mathrm{Var}(Y|X = x_j) = \sum_{i=1}^{N}(y_i - \bar{y}_j)^2 p(y_i|x_j)$$

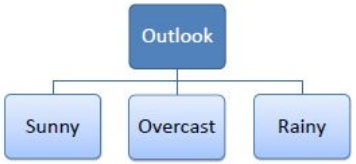
$$= \frac{1}{N}\sum_{i, x_i = x_j}(y_i - \bar{y}_j)^2 |X = x_j|$$

$$\sum_j \mathrm{Var}(Y|\mathrm{Outlook}) = \frac{4}{14}(3.49)^2 + \frac{5}{14}(7.78)^2 + \frac{5}{14}(10.87)^2$$

$$\mathrm{VarRed}(Y, X) = \mathrm{Var}(Y) - \sum_j \mathrm{Var}(Y|X = x_j)$$

- Therefore there is a variance reduction of **19.56 for the outlook variable.**

Now we've split based on Outlook



Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

- These groups now have less variance in the data. And Outlook reduced variance the most. Now we continue recursively.
- A branch set with standard deviation more than 0 needs further splitting - we need pure classes on the final leaf nodes.

Common Questions

- **What about real valued inputs?** Create a histogram of the variable, then do binary splits.

- **Are splits always binary?** In Python, yes, but not for a general algorithm. Any 3 way split can be seen as a one versus all split (ie. A&B or C versus A, B or C). It depends on the algorithm.

Follow the notebook here:

https://github.com/doriang102/Columbia_Data_Science/blob/master/notebooks/Lecture1%20-%20Introduction-to-Regression.ipynb

Great questions from last time

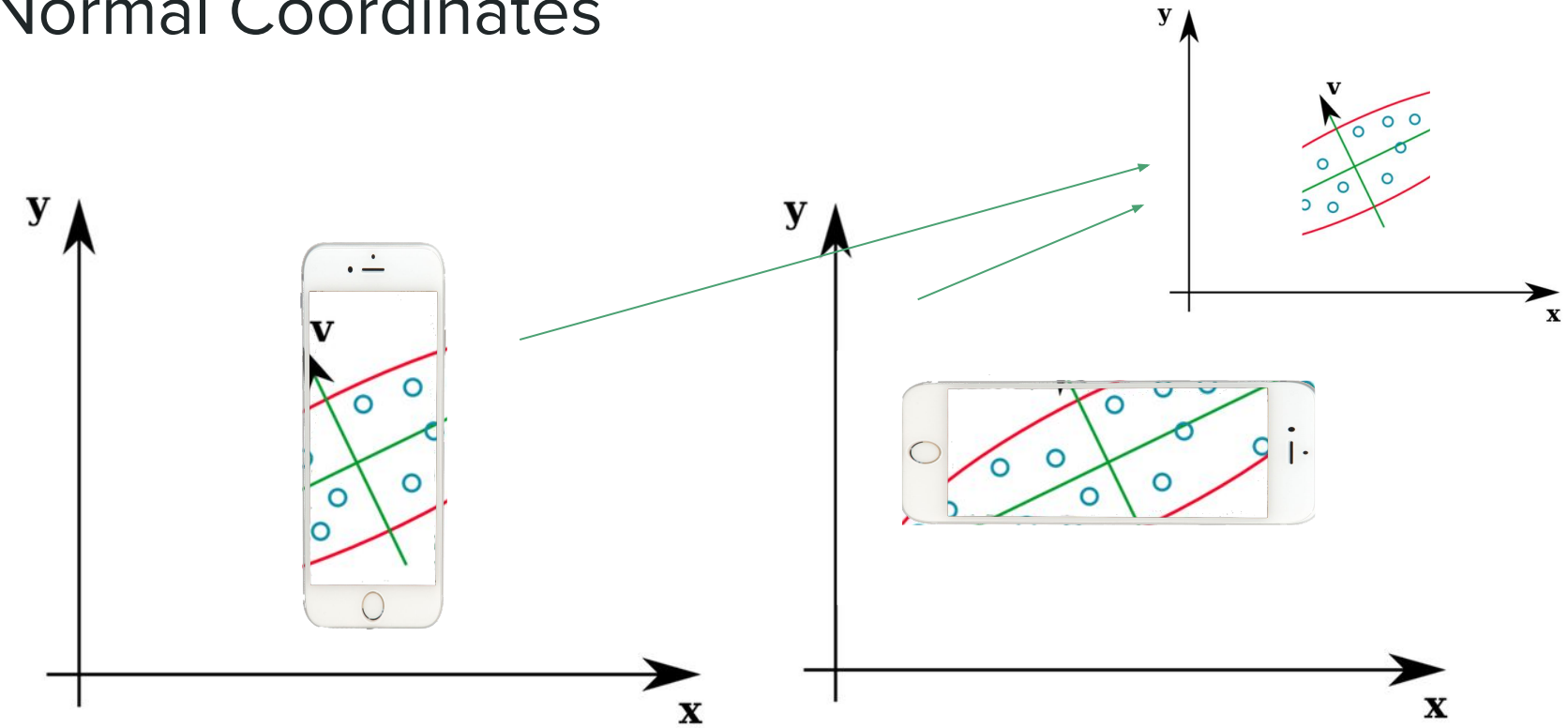
1. PCA - why are eigenvectors better than our original axes? What are they?

2. If we do K-fold cross validation, which coefficient do we choose? The mean, max? (**Answer: None of them! That's not the point**).

3. Does standardizing data result in better or worse performance?
   (**No: But not doing it prevents you from doing regularization, so it can in that way**)

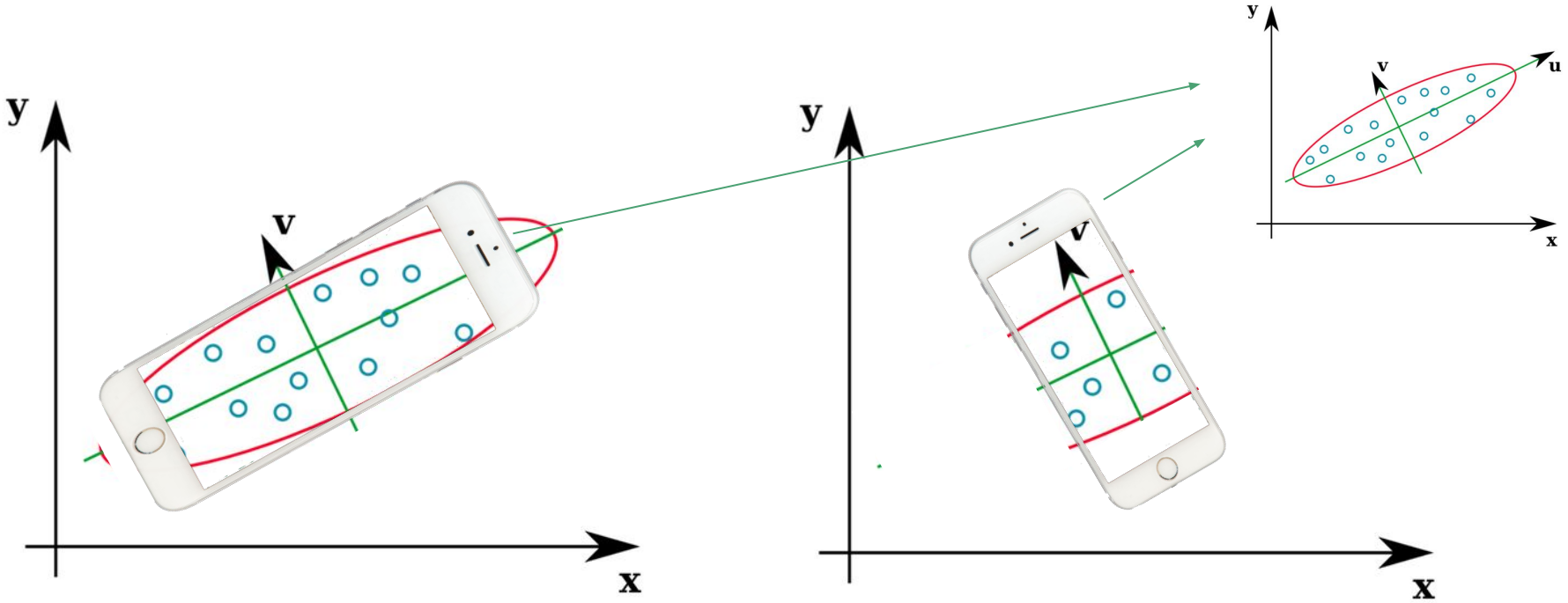
4. When to use **standardization** vs **normalization**? (these words are used in different ways by different people in the literature)

Normal Coordinates



Imagine you can only take a photo of these points in two directions - which ones would you choose?

Eigenvectors



By rotating our camera to “good” angles, we capture more about the distribution of the data! These are the eigenvectors (ie. PCA).