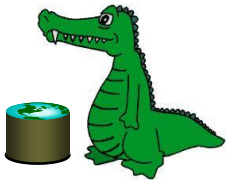
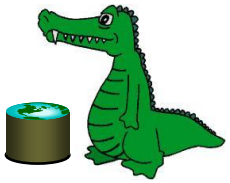


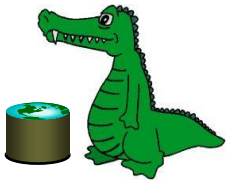
Logistics

- Lab 3 grades will be released today
- Lab 4 released
  - Monday Lab, Quiz in class
  - Homework due Tuesday 11:59pm
- Midterm 10/12



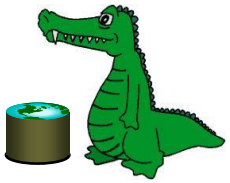
Outline

- Machine Learning
  - Supervised
  - Unsupervised
- Supervised Learning
  - Classification and Regression
  - K-NN
  - Naïve Bayes
  - Linear/Logistic Regression
  - SVM
  - Decision Tree and Random Forest



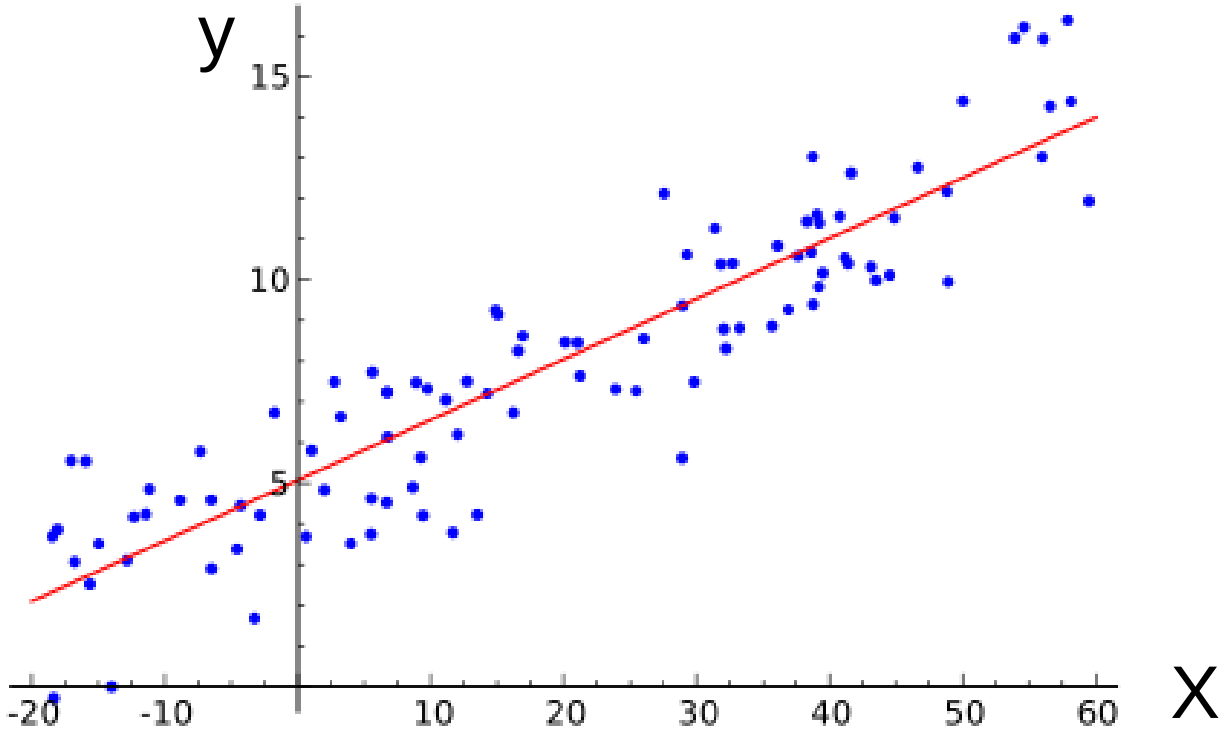
Outline

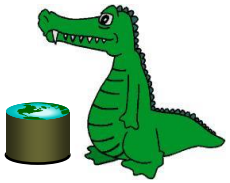
- Machine Learning
  - Supervised
  - Unsupervised
- Supervised Learning
  - Classification and Regression
  - K-NN
  - Naïve Bayes
  - Linear/Logistic Regression
  - SVM
  - Decision Tree and Random Forest



Linear Regression

We want to find the “best” line (linear function y=f(X)) to explain the data.



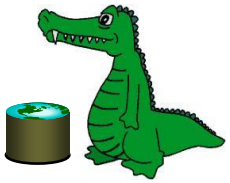


Linear Regression

The predicted value of y is given by:

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

The vector of coefficients $\hat{\beta}$ is the regression model.



Linear Regression

The regression formula

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

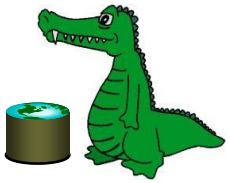
if $X_0 = 1$, can be written as a matrix product with X a row vector:

$$\hat{y} = \mathrm{X}\,\hat{\beta}$$

We get this by writing all of the input samples in a single matrix **X**:

i.e. **rows of X** $= \begin{pmatrix} X_{11} & \cdots & X_{1n} \\ \vdots & \ddots & \vdots \\ X_{m1} & \cdots & X_{mn} \end{pmatrix}$

are **distinct observations**, **columns of X** are **input features**.



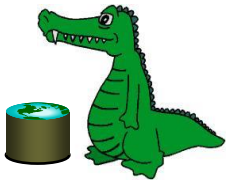
Training – Least Squares Solution

If the points are generated by an ideal line with additive Gaussian noise, the probability of a point y_j is $\Pr(y_j) = \exp\left(\frac{-(y_j - X_j\beta)^2}{2\sigma^2}\right)$ and the probability for all points is the product over j of $\Pr(y_j)$. To maximize it,

We can **easily maximize the log** of this expression $\frac{-(y_j - X_j\beta)^2}{2\sigma^2}$ for one point, or the sum of this expression at all points.

To determine the model parameters $\hat{\beta}$ from some data, we write down the Sum of Squares:

$$SS(\beta) = \sum_{i=1}^{N} (y_i - X_i\beta)^2$$

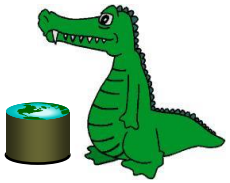


Least Squares Solution

The most common measure of fit between the line and the data is the **least-squares fit**.

i.e., minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided by a model.

There is a good reason for this: If the points are generated by an ideal line with additive Gaussian noise, the least squares solution is the ***maximum likelihood solution***.



Least Squares Solution

To minimize sum of squares:

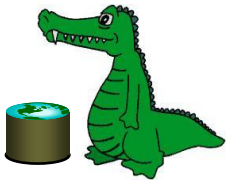
$$SS(\beta) = \sum_{i=1}^{N} (y_i - X_i\beta)^2$$

or symbolically $SS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$. We can take the derivative w.r.t. β which gives:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

And if $\mathbf{X}^T\mathbf{X}$ is non-singular, the unique solution is:

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



Iterative Least Squares

The exact method requires us to invert a matrix ($\mathbf{X}^T\mathbf{X}$) whose size is M^2 for M features and takes time $O(M^3)$. This is **too big** for large feature spaces like text or event data.

Gradient methods reduce the RSS error iteratively using the **derivative wrt** β

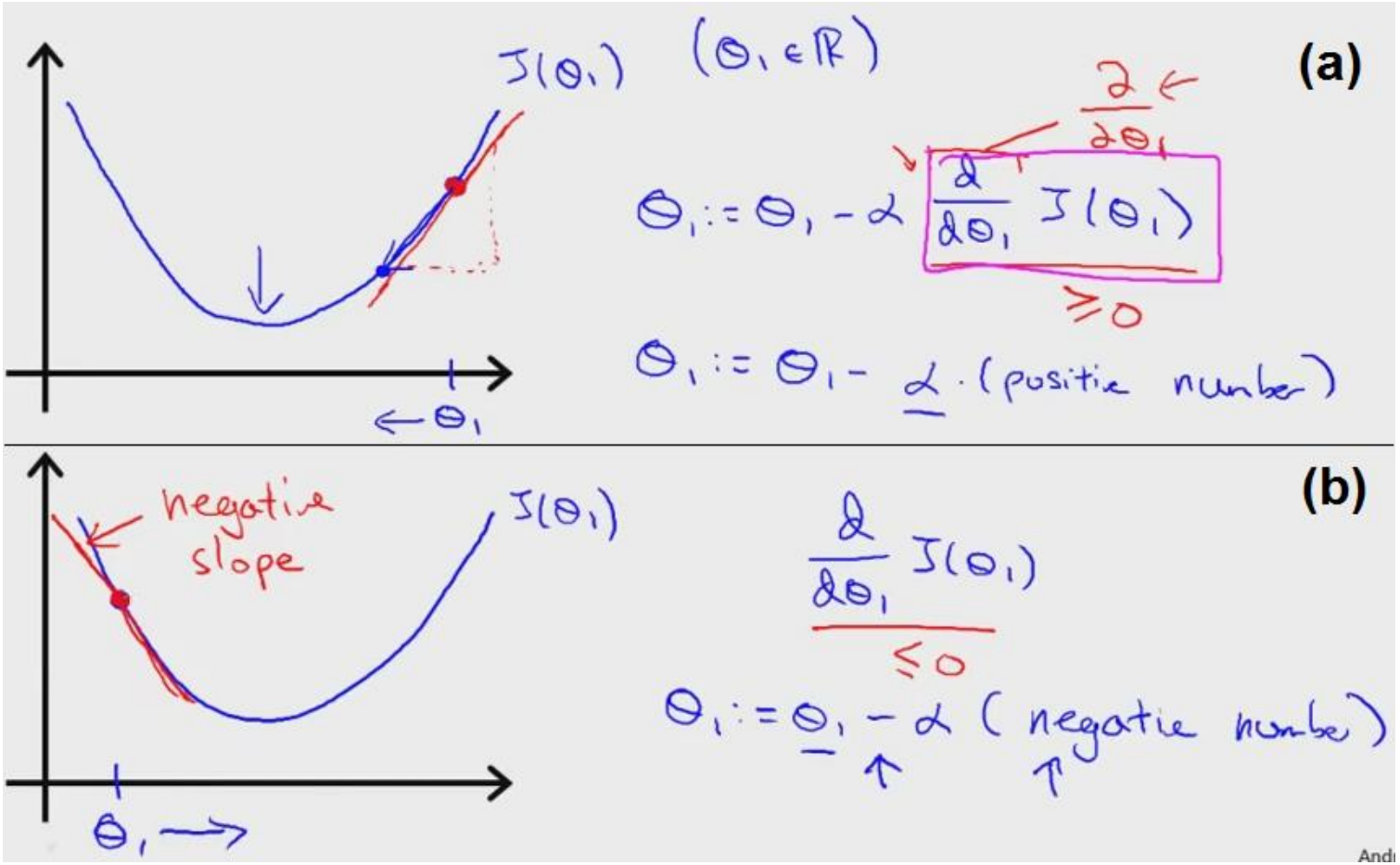
$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \beta x_i)^2$$

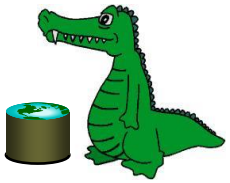
which is

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

Gradient Descent

- Finding local minimum:





Stochastic Gradient Descent (SGD)

A very important set of iterative algorithms use **stochastic gradient** updates.

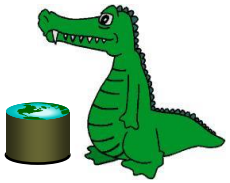
They use a **small subset or mini-batch X** of the data, and use it to compute a gradient which is added to the model

$$\beta' = \beta - \alpha \nabla$$

Where α is called the **learning rate**.

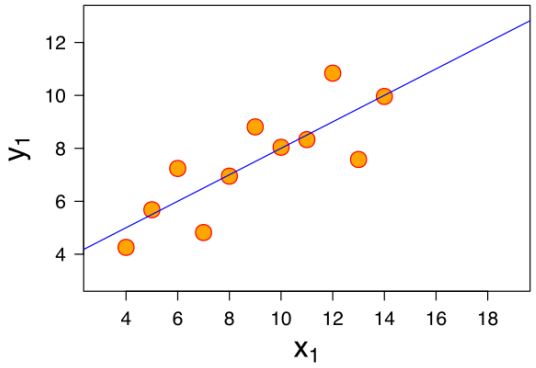
These updates happen **many times** in one pass over the dataset.

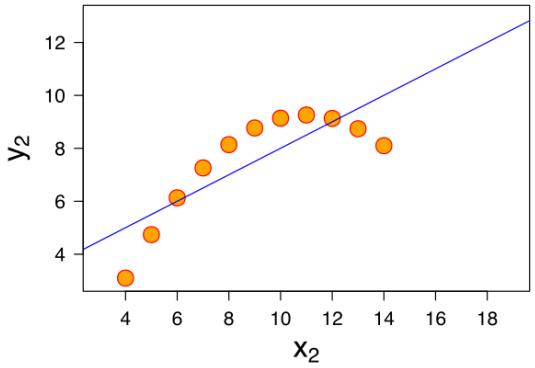
Its possible to compute high-quality models with very few passes, sometime with less than one pass over a large dataset.

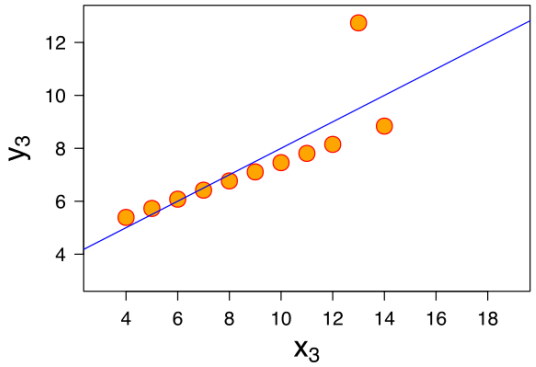


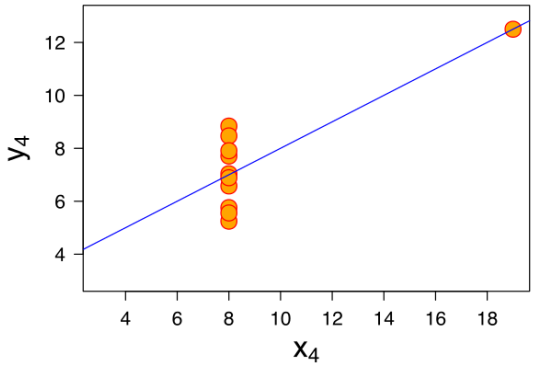
Fitness of the Model

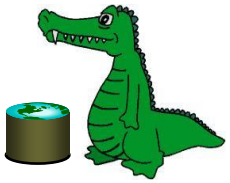
We can **always** fit a linear model to any dataset, but how do we know if there is a **real linear relationship**?











R^2-values

Approach: Measure how much the total "noise" (variance) is reduced when we include the line as an offset.

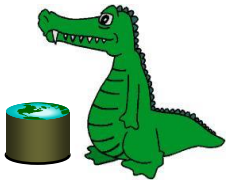
R-squared: a suitable measure. Let $\hat{y} = \mathrm{X}\,\hat{\beta}$ be a predicted value, and $\bar{y}$ be the sample mean. Then the R-squared value is

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

And can be described as the fraction of the total variance not explained by the model.

$R^2 = 0$: bad model. No evidence of a linear relationship.

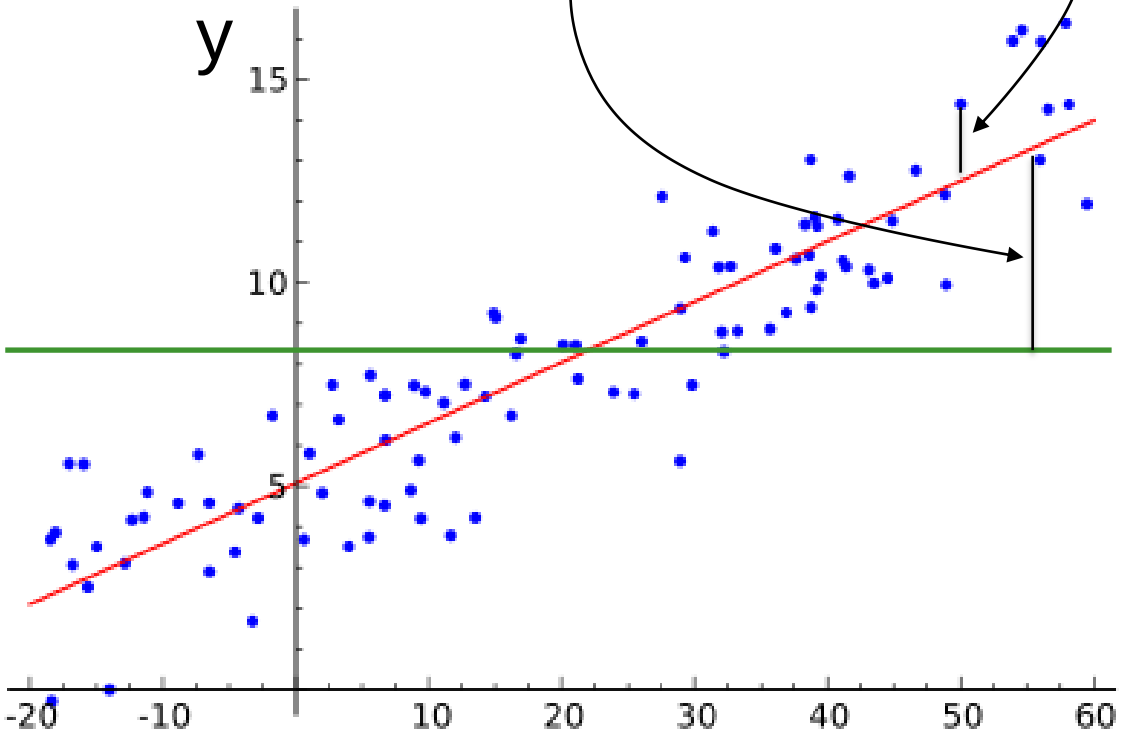
$R^2 = 1$: good model. The line perfectly fits the data.



R-squared

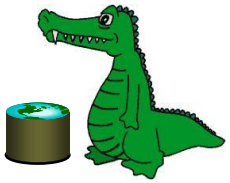
Small if good fit

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$



Line of $\hat{y}$

Line of $\bar{y}$



Logistic Regression

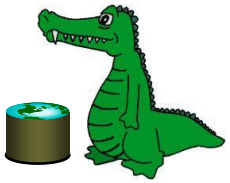
- We made a distinction earlier between regression (predicting a real value) and classification (predicting a discrete value).

- Logistic regression is designed as a **binary classifier** (output say {0,1}) but actually **outputs the probability** that the input instance is in the "1" class.

- A logistic classifier has the form:

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

where $X = (X_1, \ldots, X_n)$ is a vector of features.

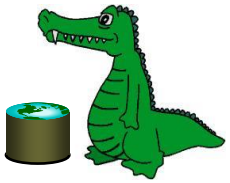


Logistic Regression

- Logistic regression is probably the **most widely used general-purpose classifier**.

- Its **very scalable** and can be **very fast** to train. It's used for
  - Spam filtering
  - News message classification
  - Web site classification
  - Product classification
  - Most classification problems with large, sparse feature sets.

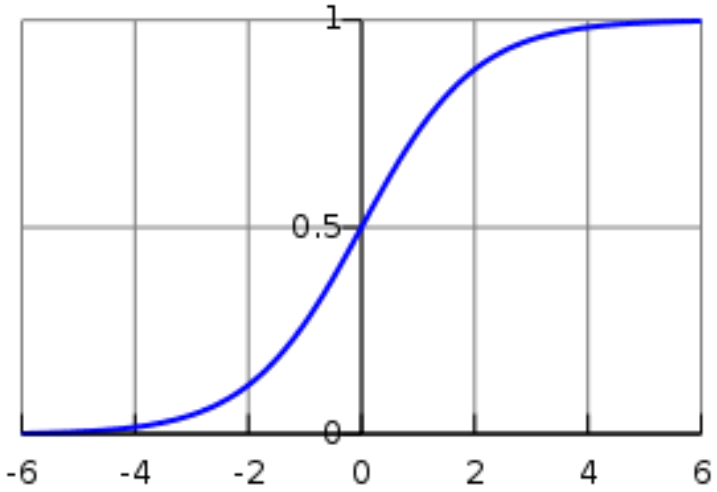
- The only caveat is that **it can overfit** on very sparse data, so its often used with Regularization



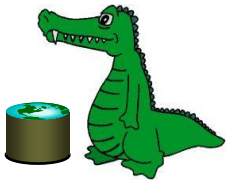
Logistic Regression

- Logistic regression maps the "regression" value $-X\beta$ in (-∞,∞) to the range [0,1] using a "logistic" function:

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

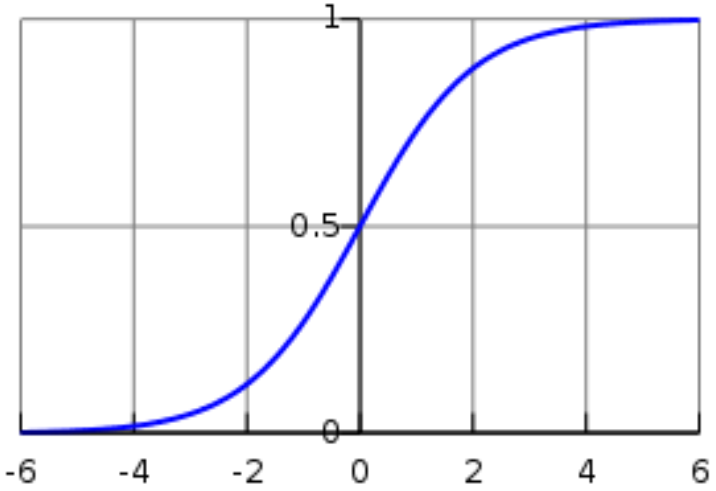


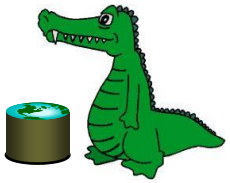
- i.e. the logistic function maps any value on the real line to a probability in the range [0,1]



Logistic Regression

- Where did the logistic function come from?





Logistic Training

For training, we start with a collection of **input values** X^i and corresponding **output labels** $y^i \in \{0,1\}$. Let p^i be the **predicted output** on input X^i, so

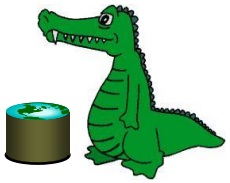
$$p^i = \frac{1}{1 + \exp(-X^i \beta)}$$

The **accuracy** on an input X^i is

$$A^i = y^i p^i + (1 - y^i)(1 - p^i)$$

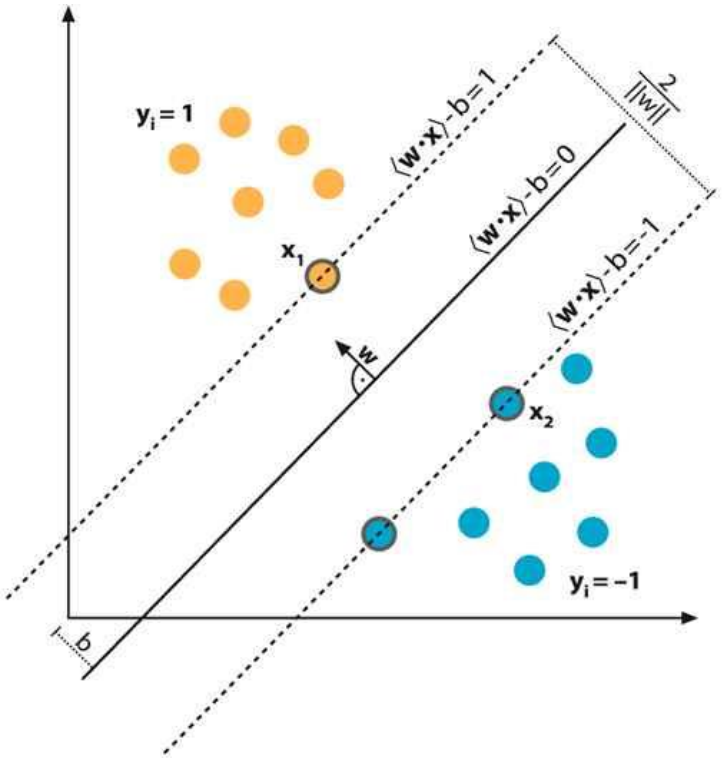
Logistic regression can use SGD to maximize either the sum of the log accuracy, or the total accuracy, e.g.

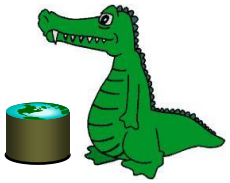
$$A = \sum_{i=1}^{N} \log A^i$$



Support Vector Machines

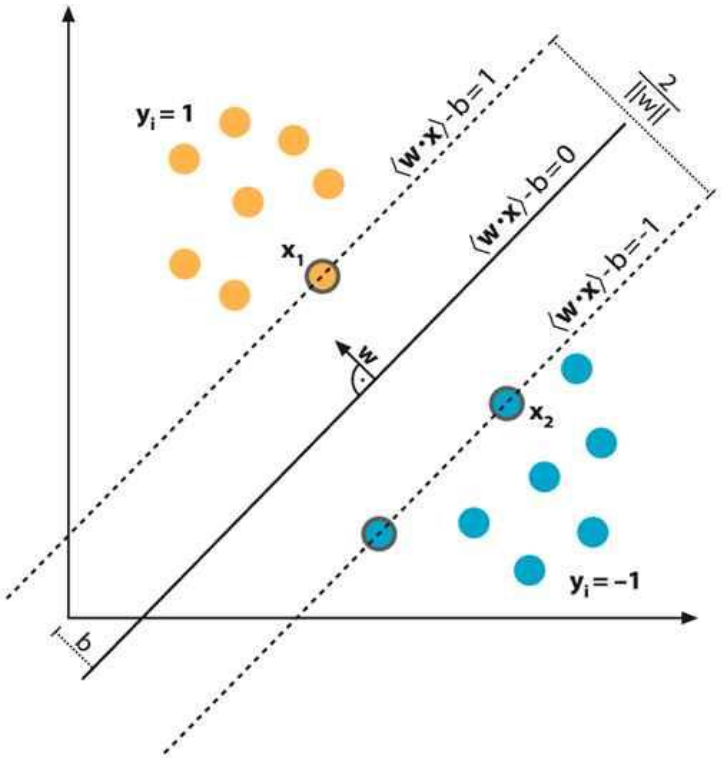
- A Support Vector Machine (SVM) is a classifier that tries to **maximize the margin** between training data and the classification boundary (the plane defined by $X\beta = 0$)

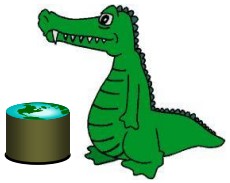




Support Vector Machines

- The idea is that maximizing the margin **maximizes the chance that classification will be correct on new data**. We assume the new data of each class is near the training data of that type.





SVM Training

SVMs can also be trained using SGD.

The objective function to be minimized here is ||w||, which would in term maximize the margin of the two hyperplanes

Subject to the constraint that $(w \bullet x^i + b)\, y^i \geq 1$

The constraint make sure that each data point must lie on the correct side of the margin