CS5841/EE5841 Machine Learning

Lecture 4: Classification Part 1

Evan Lucas

Overview

- Course updates
- Classification Intro

Class updates

- HW1 updated
- Error in Linear Algebra Quiz

Reading for KNN

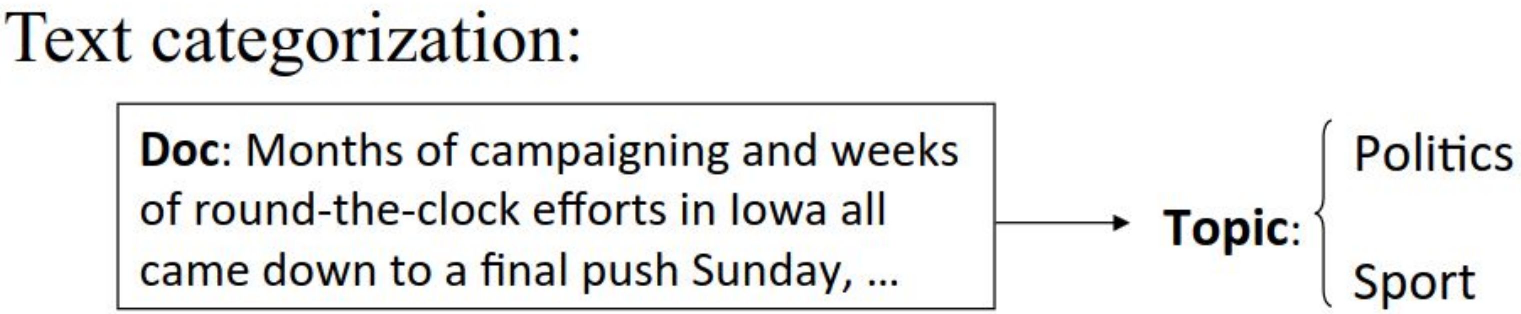
Bishop 2.5

Mitchell 8

Classification definition

- Given input x = $(x_1, x_2,\ldots,x_d)$
- Predict output class label $y \in Y$
  - Binary classification y = {0,1} or {-1,1}
  - Multi-class classification = {1,2,…,C}
- Learn a classification function f(x) : $\mathbb{R}^d \mapsto \mathcal{Y}$

Examples of classification problems



Input features: X

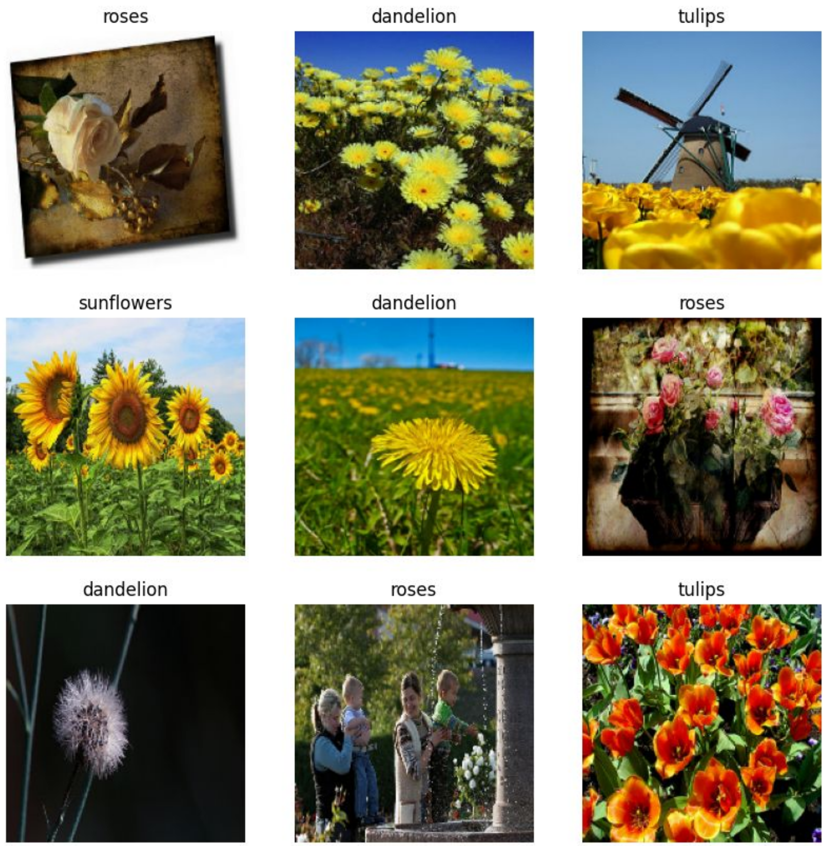
- Classic method: word frequency

Class label: y

- 'Politics': y = 0
- 'Sport': y = 1
- …

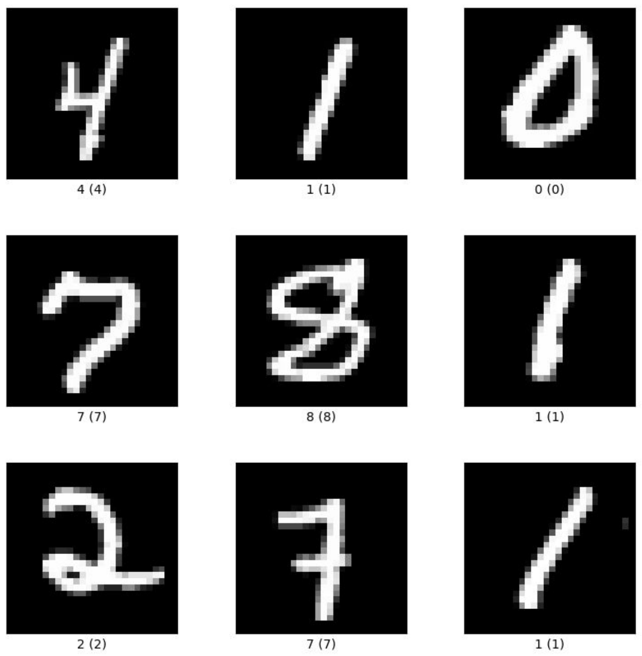
Examples of classification problems

- What is the flower in the picture?
- Features X
  - Classic - histogram or other computed image statistics
  - Modern - pixels
- Class label y
  - 0: ‘roses’
  - 1: ‘dandelion’
  - etc.



Examples of classification problems

- MNIST!
- Features X
  - Classic - Classic - histogram or other computed image statistics
  - Modern - pixels
- Class label y
  - 0: ‘0’
  - 1: ‘1’
  - etc.



Supervised learning

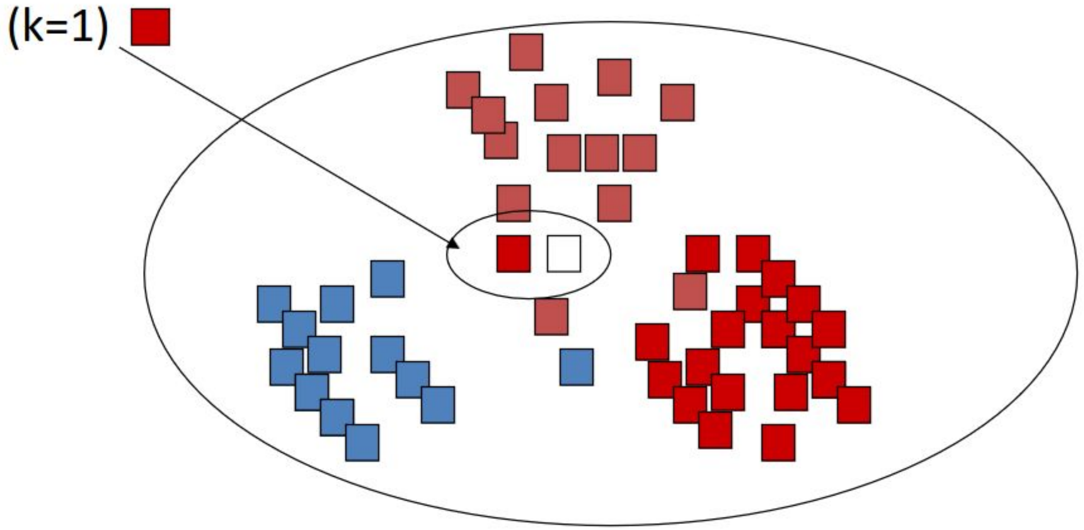
- Training examples:
  - $D = \{(x_i, y_i), i = 1, \ldots, n\}$
- Assume independent identically distributed (IID)
  - Critical assumption for many ML methods
  - Independent - all events are independent and not connected to other events
  - Identically distributed - the distribution does not fluctuate and all items are sampled from the same distribution
  - Is this a good assumption?

Regression and Classification

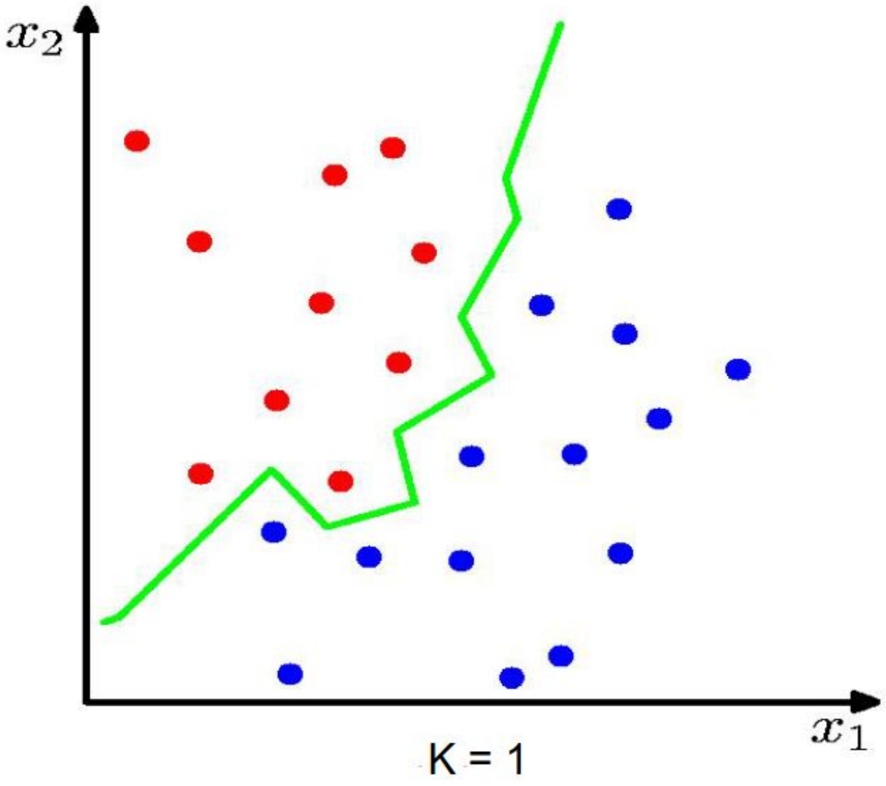
- A common approach is to turn binary classification into a regression problem
  - Just pretend that the binary label is actually a variable output
  - This can be extended to multi-class ordinal classification, but not non-ordinal classification (what is the relationship between 'dandelion' and 'rose'?)
  - Computationally efficient, but ignores discrete nature of class label
- Regression can be made into classification through thresholding
  - Ex: a regressed value greater than x is assigned 1 and lower than x is assigned 0

KNN - the data is the model

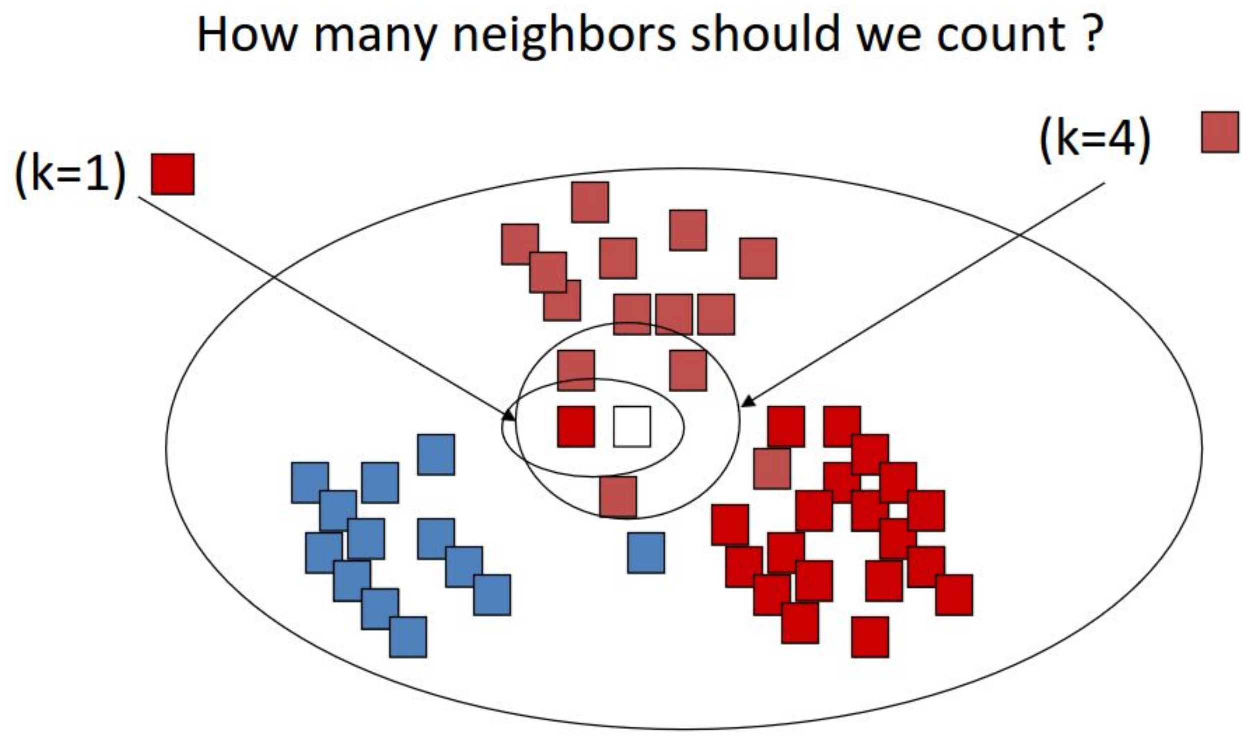
- “You are the average of your friends”
  - This, but applied to data



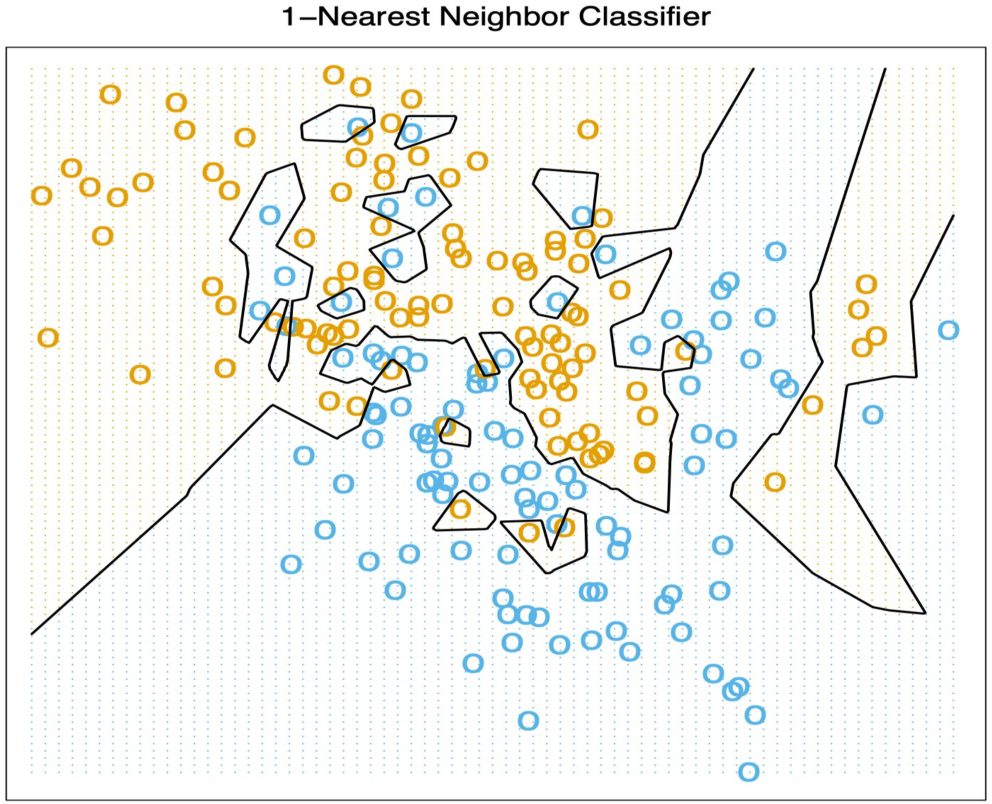
Decision boundaries

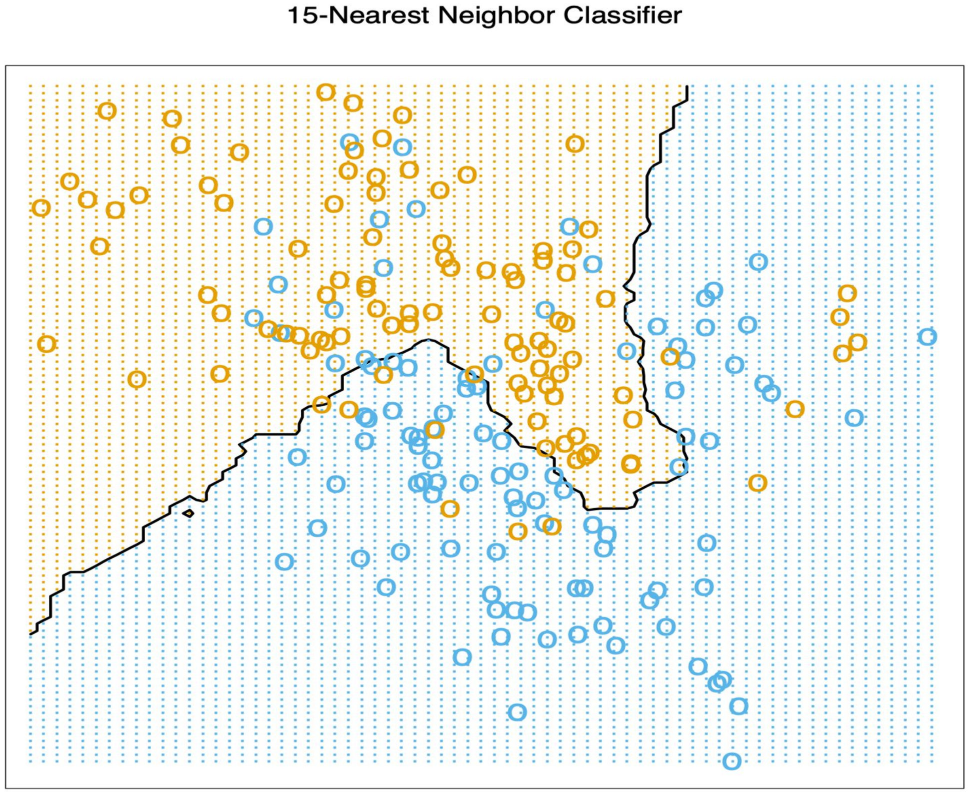


Adjusting K in KNN



Adjusting K in KNN





Cross validation with KNN

- Predict class labels for validation set using the training set examples
- Choose *K* that maximizes classification accuracy
- Possibly choose distance metric as well

Probabilistic interpretation of KNN

- Estimating conditional probability P(y|x)
  - Class y given neighborhood of x
- Bias and variance tradeoff
  - Small k - larger variance (less stable)
  - Big k - larger bias (less true)

When is KNN useful?

- Abundance of training data
- Few attributes/features per point
- Pros
  - No training time
  - Learns complex target functions
  - No information loss
- Cons
  - Slow
  - Easily overfits

Curse of dimensionality

- Motivating example:
  - Consider a case with 20 attributes, but only 2 are relevant to target function.
- Formal definition:
  - These data points are uniformly distributed in a p-dimensional unit ball centered at the origin.
  - Consider a KNN estimate from the origin.
  - Mean distance from origin to closest data point is

$$d(p, N) = \left(1 - 2^{-1/N}\right)^{1/p} \approx 1 - \frac{\log N}{p}$$

Weighted KNN

- Weight contribution of each neighbor based on distance

$$w(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\lambda |\mathbf{x} - \mathbf{x}_i|_2^2\right)$$

$$\Pr(y|\mathbf{x}) = \frac{\sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)\delta(y, y_i)}{\sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)}$$

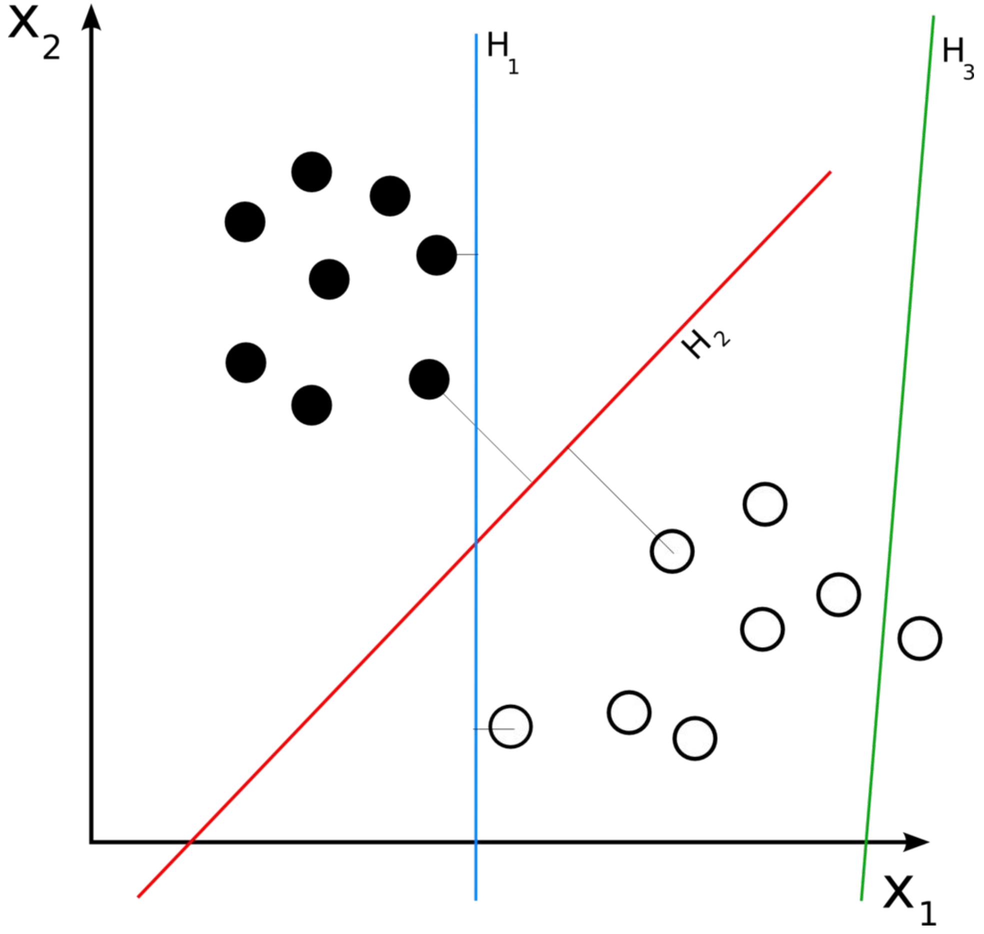
$$\delta(y, y_i) = \begin{cases} 1 & y = y_i \\ 0 & y \neq y_i \end{cases}$$

Reading for Logistic Regression

- Bishop 4.3
- ESL 4

Linear classifiers

- Can we separate our classes with a line?
- What makes a good linear classifier?



https://en.wikipedia.org/wiki/Linear_classifier#/media/File:Svm_separating_hyperplanes.png

Linear classifiers

- Map input features into some score space

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right)$$

- Often we use a threshold function

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T \cdot \mathbf{x} > \theta, \\ 0 & \text{otherwise} \end{cases}$$

https://en.wikipedia.org/wiki/Linear_classifier#/media/File:Svm_separating_hyperplanes.png

Generative and Discriminative models

Discriminative Models

- Model P(y|x)
- Pros
  - Usually good performance
- Cons
  - Slow convergence
  - Expensive computation
  - Sensitive to noise data

Generative Models

- Model P(x|y)
- Pros
  - Usually fast converge
  - Cheap computation
  - Robust to noise data
- Cons
  - Usually performs worse

Generative Models

- Given a class, what is the distribution of features?
- Examples:
  - Naive Bayes
  - Linear Discriminant Analysis
- Model the joint probability distribution

Discriminative models

- Where can we set decision boundaries on observed features?
- Examples:
  - Support vector machine (SVM)
  - Logistic regression
  - Perceptrons
- Model conditional density functions
- Slightly more challenging to train

Training discriminative models

- General form
  - w - model weights
  - R - regularization function
  - C - balance between loss and regularization
  - L( ) - chosen loss between prediction and true label

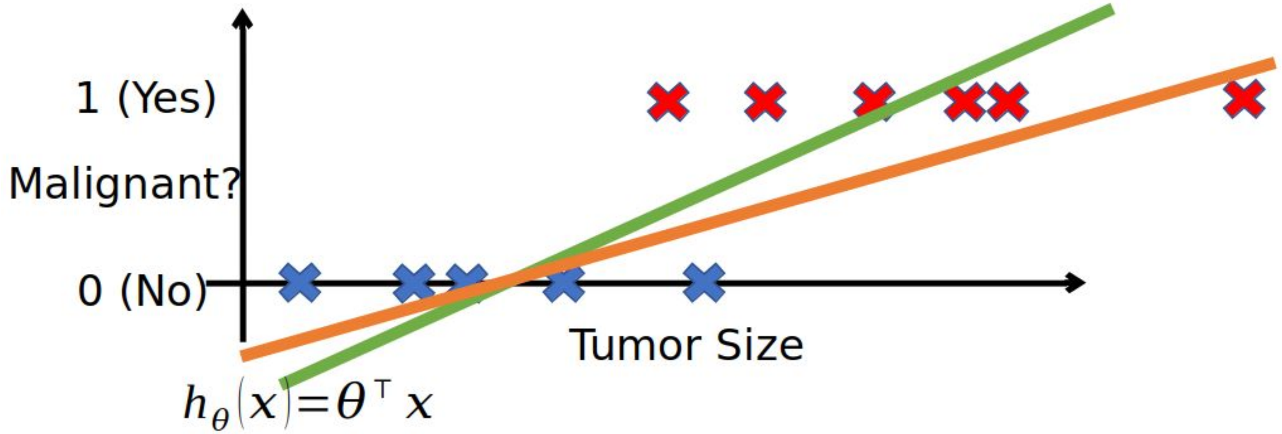
$$\arg\min_{\mathbf{w}} R(\mathbf{w}) + C \sum_{i=1}^{N} L(y_i, \mathbf{w}^\top \mathbf{x}_i)$$

Note about following slides

- I'm using materials from the last ML course
  - Some variables are different
  - Cost == Loss
  - θ == w
  -

Binary classification

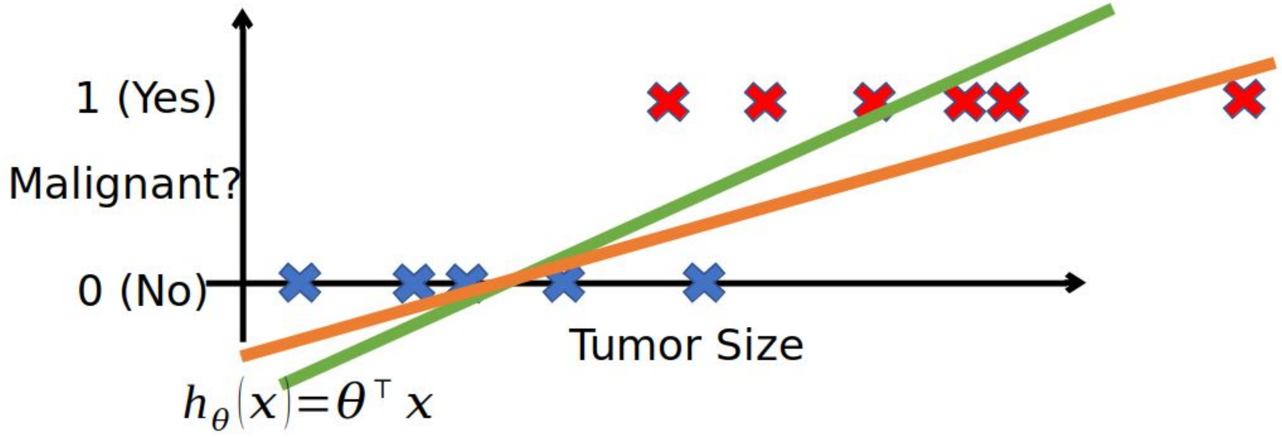
- Based on linear regression
- Threshold classifier output at 0.5
  - If < 0.5, predict "0"
  - If > 0.5, predict "1"
- What are potential issues with this?



Binary classification, continued

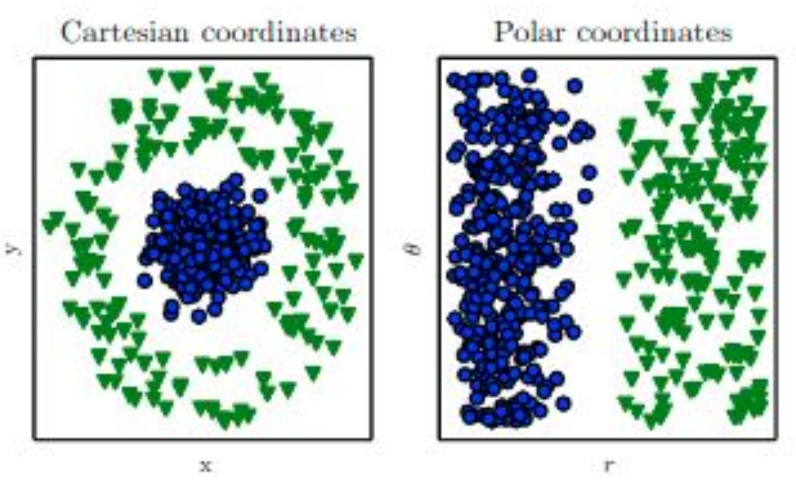
- Problem 1: if we had a single Yes with a very large tumour.

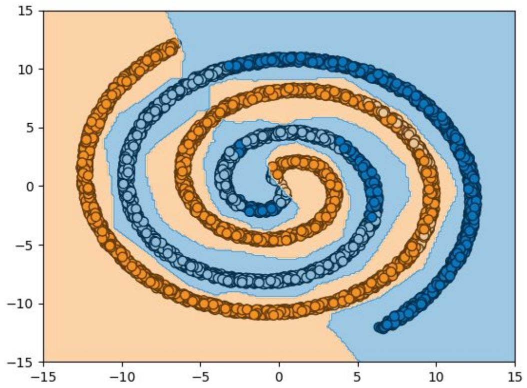
- Problem 2: Hypothesis can give values large than 1 or less than 0



Non-linear decision boundary (preview)

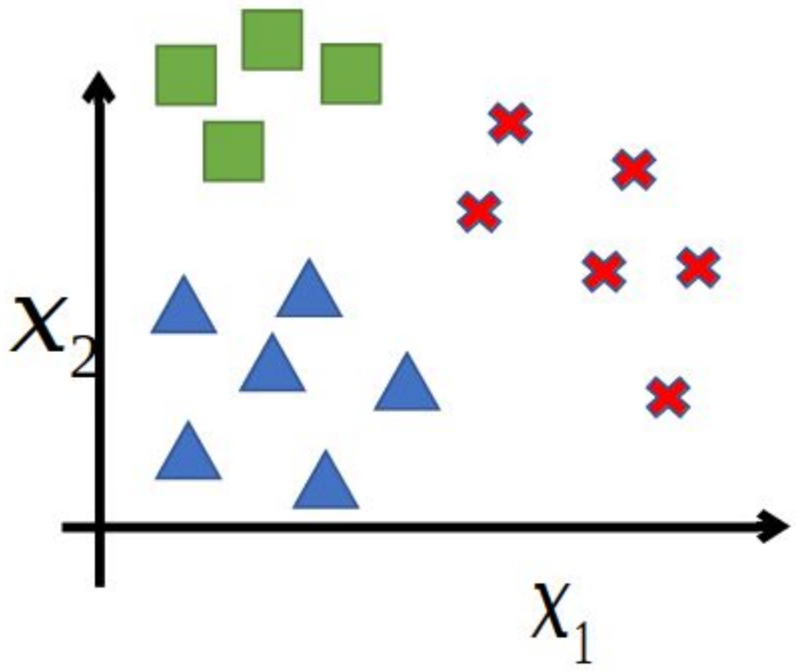
- Option 1: Use a different data representation
- Option 2: Apply a kernel



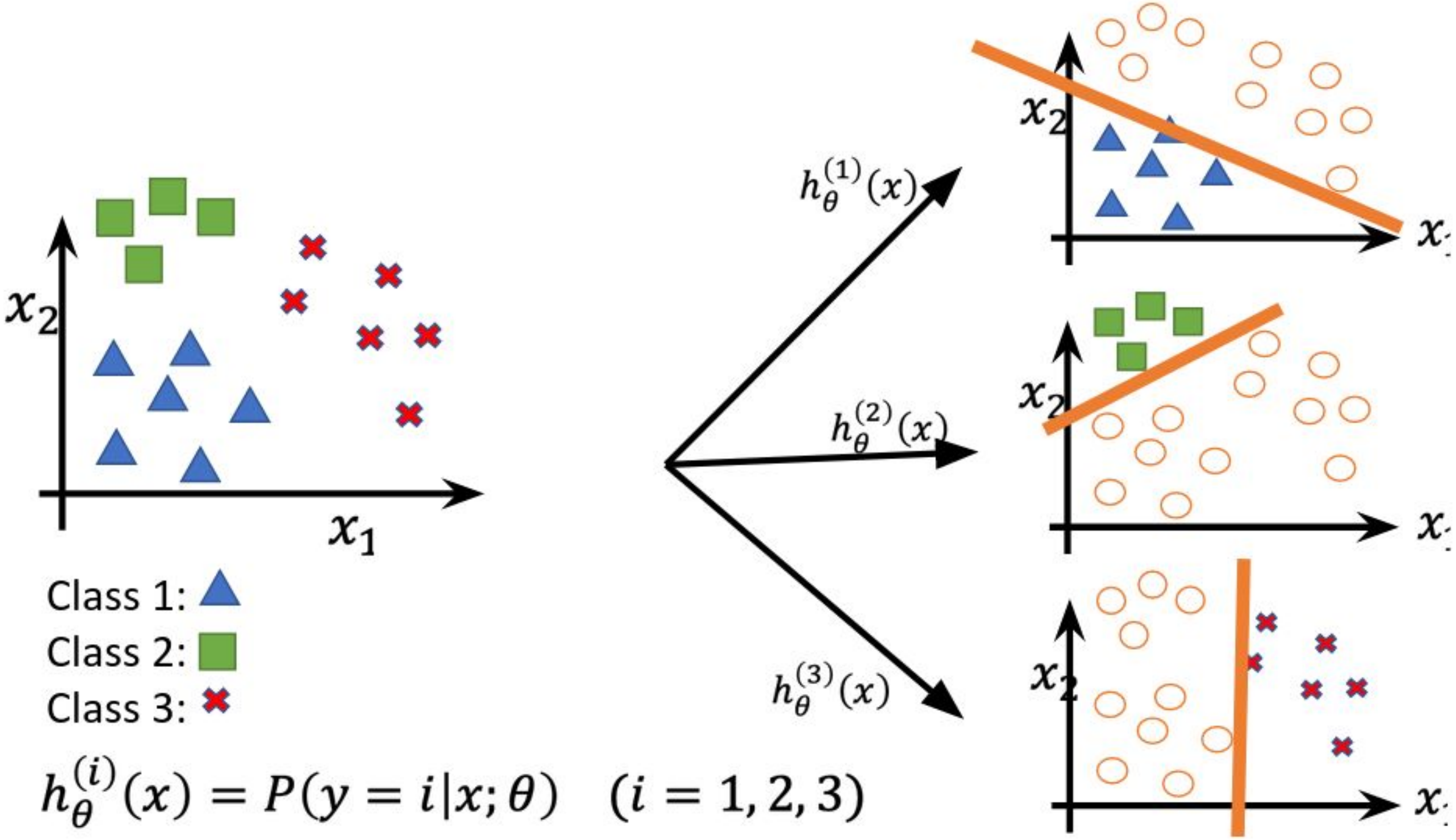


Multi-class classification

- How can we extend a binary classifier to handle multiple classes?



One-vs-all classification

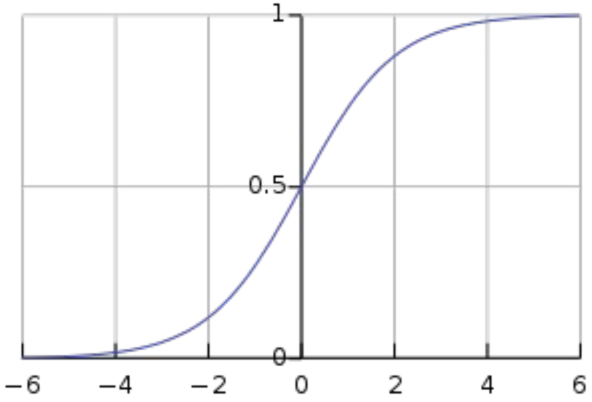


$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \quad (i = 1, 2, 3)$$

One solution to these problems

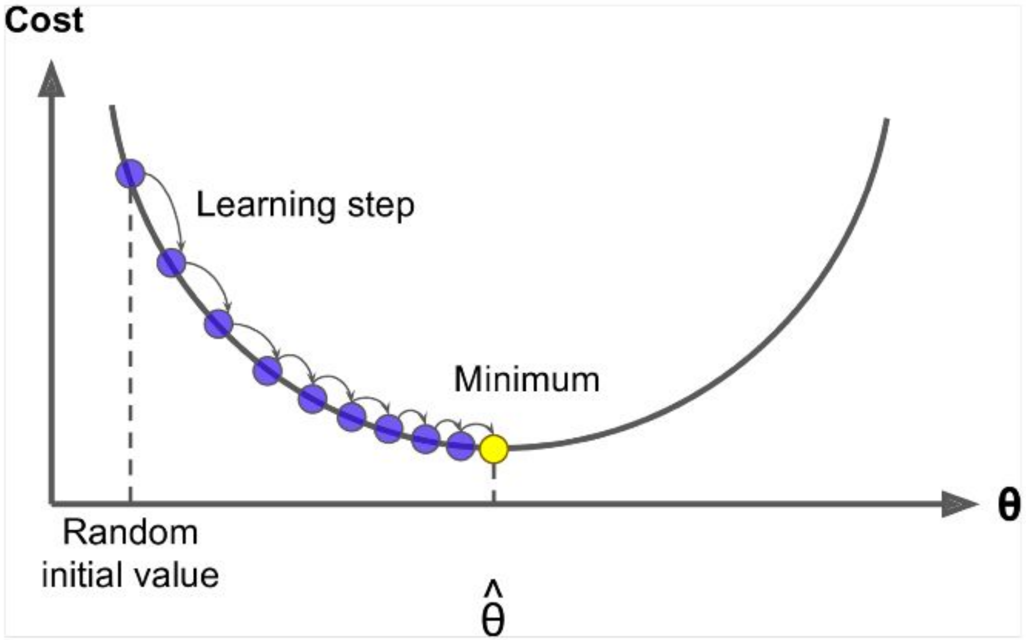
- Use a function that maps into {0,1}
- One option: The Sigmoid/Logistic function

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



Gradient descent

- Iteratively update weights
- How do we determine step size?
  - Should step size be constant?

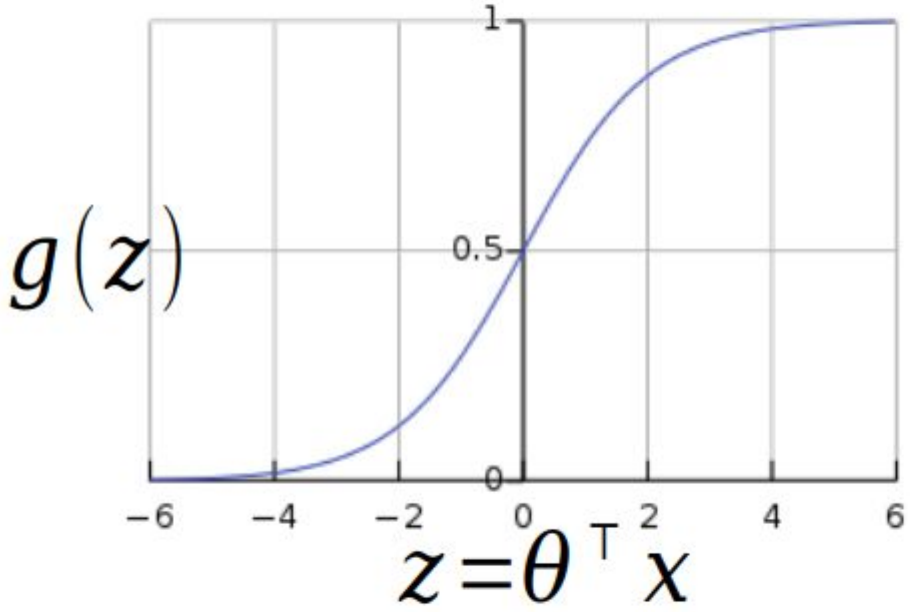


$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \mathcal{L}(\mathbf{w})$$

$$\eta_t \propto 1/\sqrt{t}$$

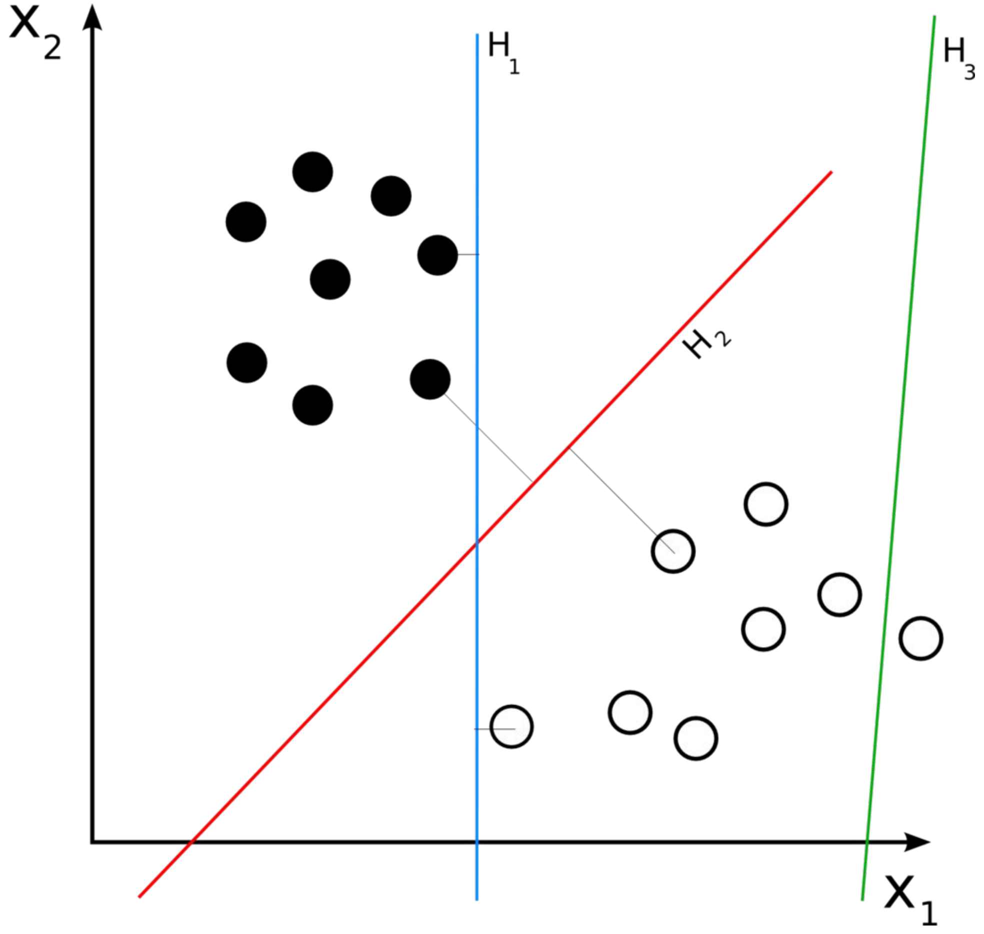
Logistic regression

- Predict “y=1” if z > 0
- Predict “y=0” if z < 0



Decision boundaries

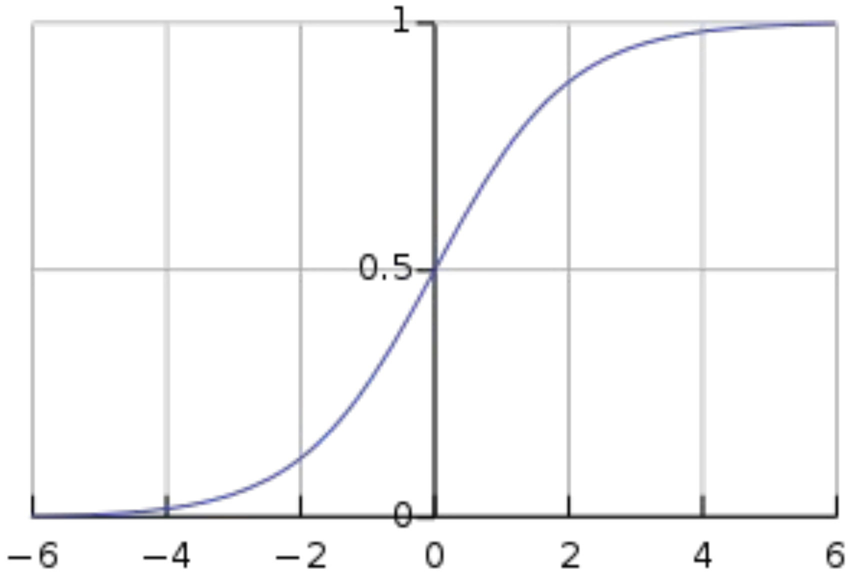
- Logistic regression can be represented as a decision boundary
- Boundary is set of points where $h_w(x) = 0.5$



Interpreting Logistic Regression

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$



Let’s think through some examples:

- P(y=1| h(x) = 2)
- P(y=0| h(x) = -4)

Let’s look into the logistic function

$$g(z) = \frac{1}{1+e^{-z}}$$

$$\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}$$

Likelihood of parameters

- Assume *m* independent training examples

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X; \theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

Log likelihood

- Maximize the likelihood

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))
\end{aligned}
$$

Gradient ascent of likelihood

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x) \frac{\partial}{\partial \theta_j} \theta^T x \\
&= \left(y(1 - g(\theta^T x)) - (1-y) g(\theta^T x) \right) x_j \\
&= (y - h_\theta(x))\, x_j
\end{aligned}
$$

$$g'(z) = g(z)(1 - g(z))$$

Gradient ascent rule

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

Sidebar: how often do we update weights?

- Gradient descent (GD) - consider all training data for each weight update
  - Usually better minimum
  - In complex models, sometimes doesn't learn features unique to subsets
- Stochastic gradient descent (SGD) - consider one point at a time
  - Usually faster
  - Usually not as well minimized
- Mini-batch gradient descent - use a small batch

Questions + Comments?