Computational Data Analysis

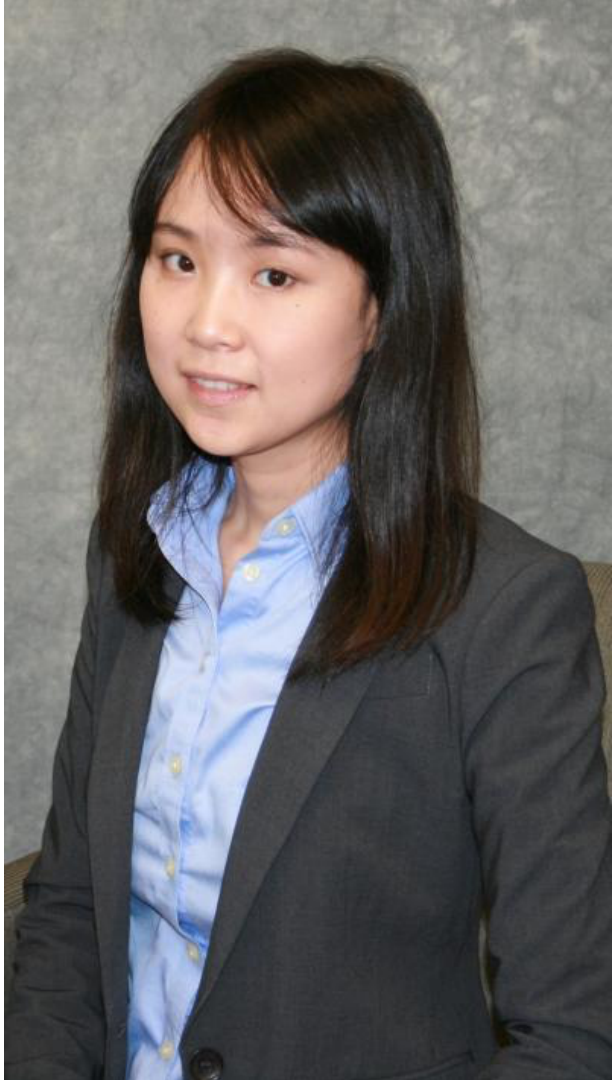
Machine Learning

Yao Xie, Ph.D.

Associate Professor

Harold R. and Mary Anne Nash Early Career Professor

H. Milton Stewart School of Industrial and Systems Engineering

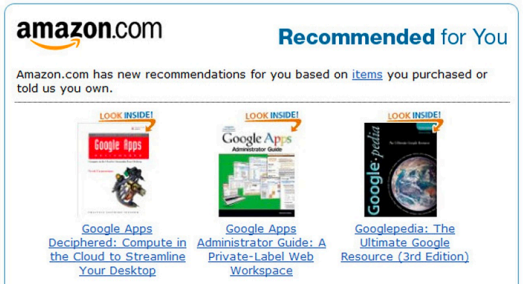


Classification

Classification

- Classification is a predictive task: given input, predict categorical output (in the simplest case, binary category)
- Supervised learning: training data include both features (input) and the response (output)
- Examples: healthcare, user preference (recommender systems), email spam filtering

'Doctor in clinic':
https://www.leafly.com/news/health/how-to-find-a-doctor-or-clinic-that-specializes-in-medical-mariju

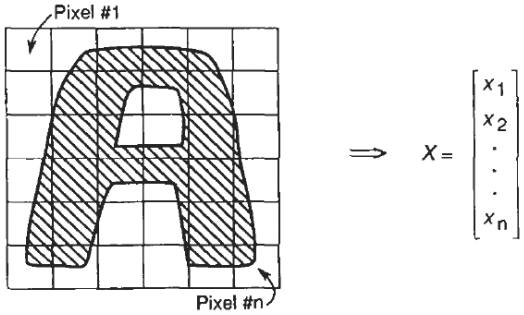


'amazon recommendation':
https://www.mageplaza.com/blog/product-recomendation-how-amazon-succeeds-with-it.html

Georgia Tech

Classification

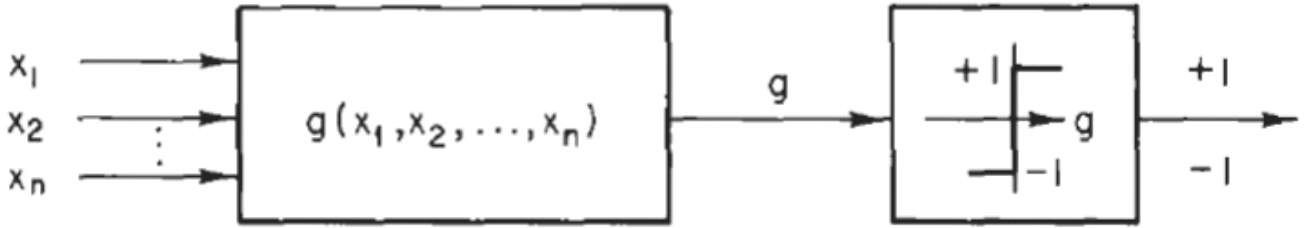
Represent the data as a vector (features)



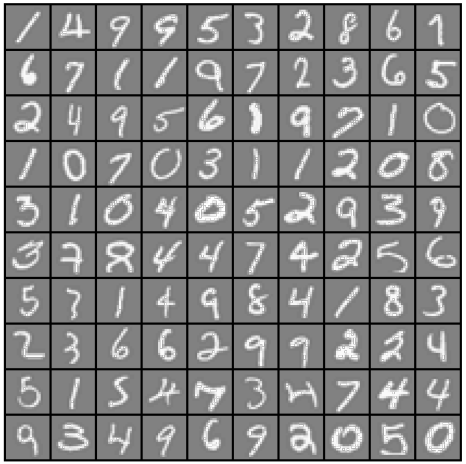
A label is provided for each data point, eg., $y \in \{-1, +1\}$,

sometimes $y = \{0,1\}$

Classifier



Classification versus clustering



Classification (training with label)

?

5

?

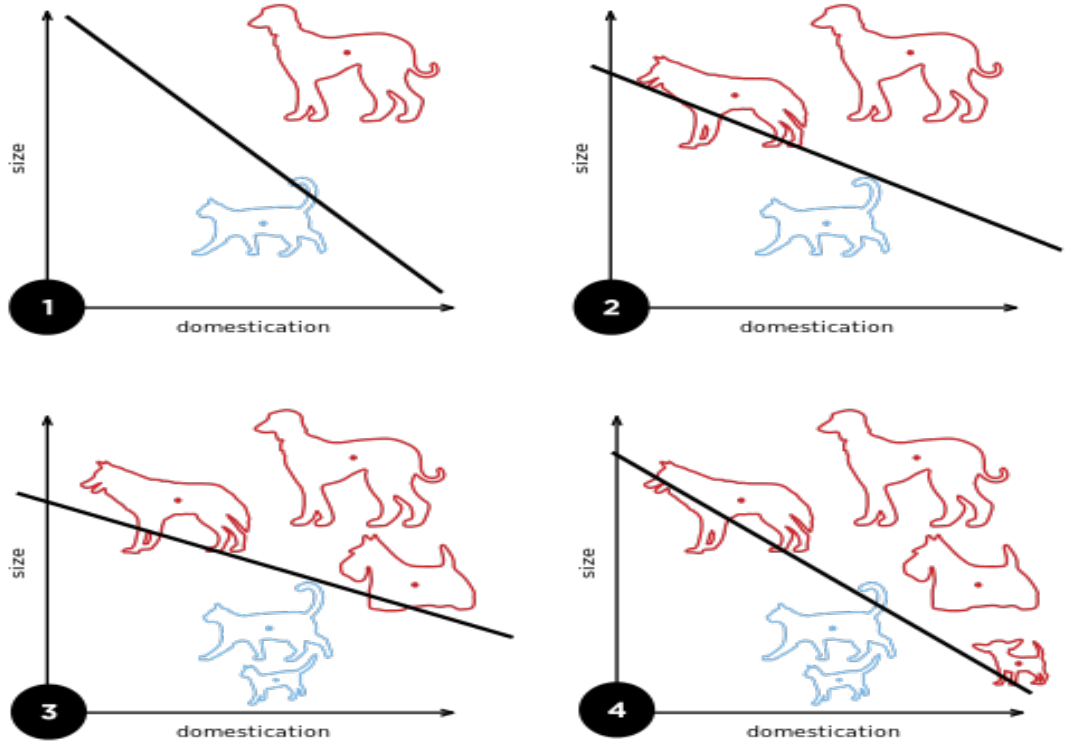
7

Supervised learning

Clustering (no label)

Unsupervised learning

Decision boundary



Classify cats and dogs?

Features:
“size”
“domestication”

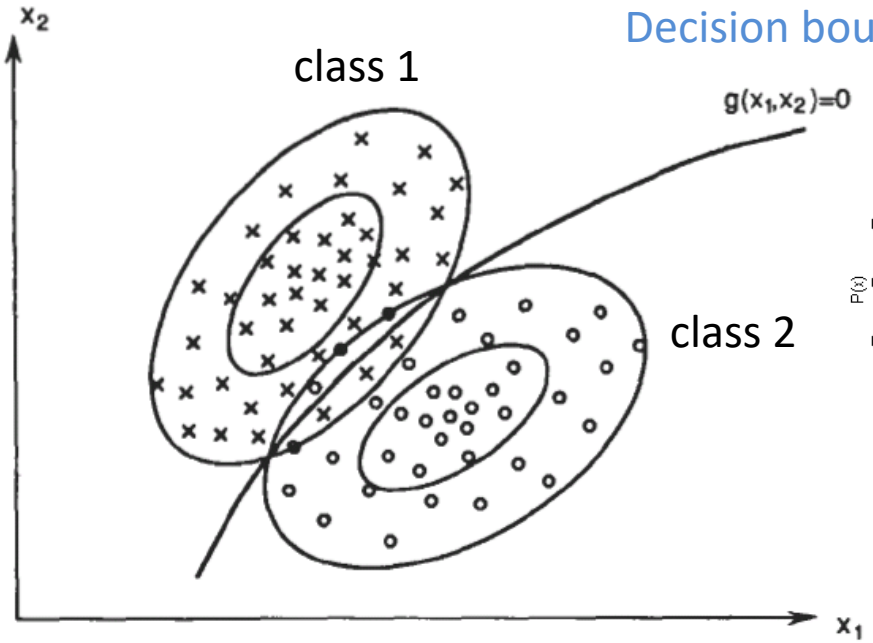
Classification algorithms

- Bayes classifier
  - parametric model for each class
- K-nearest neighbors
  - non-parametric method
- Logistic regression
  - modeling how features map to labels

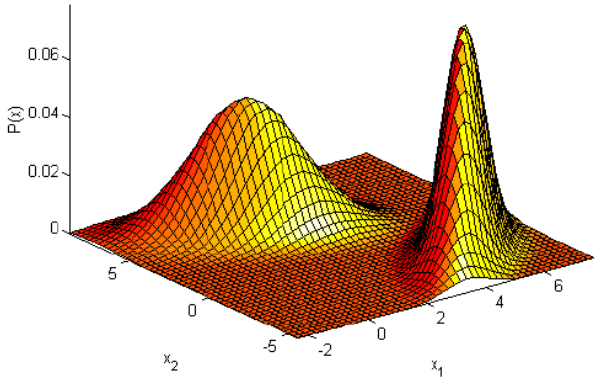
(more to come)

- SVM
  - Geometry based method
- Neural networks
  - capture complex non-linear decision boundary
- Ensemble methods, boosting, decision tree

Decision-making: divide high-dimensional space



Decision boundary

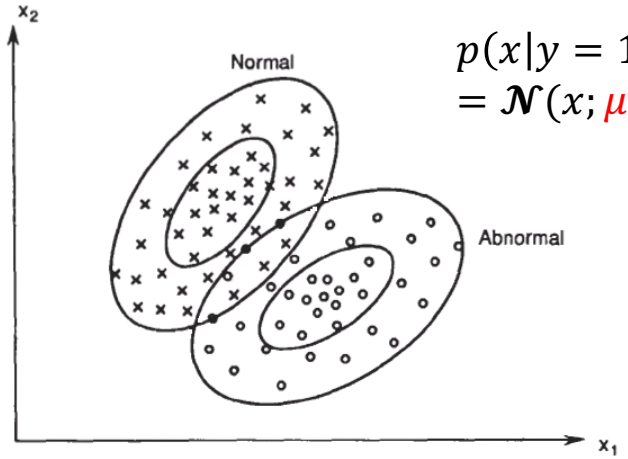


Bayes classifier

- Focus on binary classification, can be extend to multiple class classification
- Parametric approach: Assume distribution for features for each class (likelihood)

$$P(x|y=1), P(x|y=-1)$$

- For example



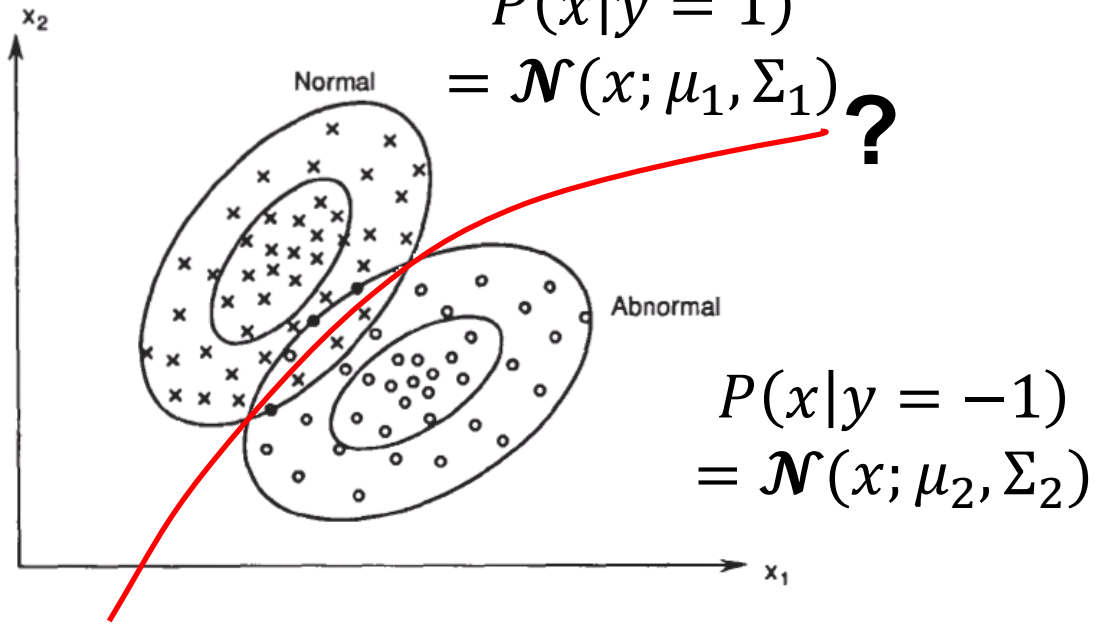
$$p(x|y=1) = \mathcal{N}(x; \mu_1, \Sigma_1)$$

$$p(x|y=-1) = \mathcal{N}(x; \mu_2, \Sigma_2)$$

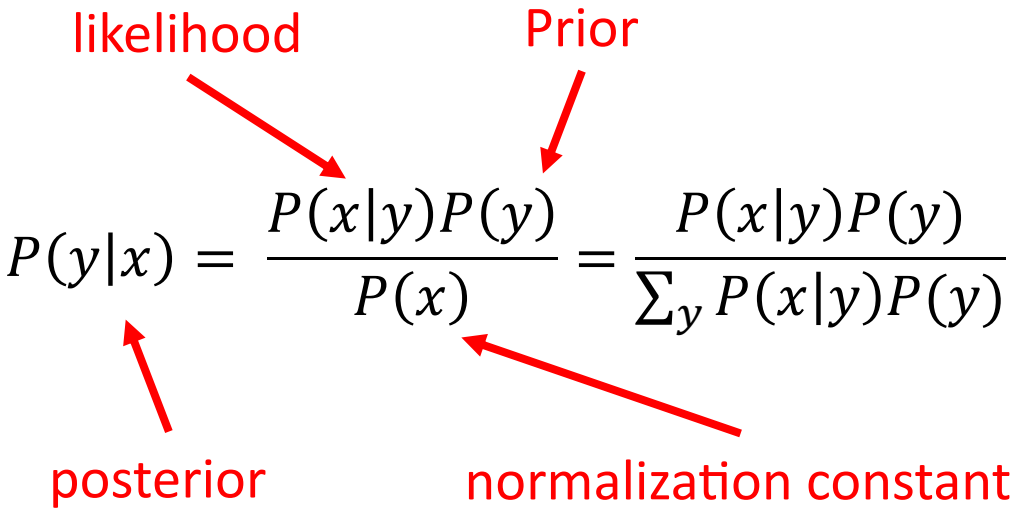
- Class prior: $P(y=1), P(y=-1)$

How to come up with decision boundary

Given class conditional distribution (likelihood): $P(x|y=1), P(x|y=-1)$, and class prior: $P(y=1), P(y=-1)$



Use Bayes rule



Bayes Decision Rule

- Learning/specify class **prior**: $p(y)$, **likelihood**: $p(x|y)$

- Calculate **posterior** probability of a test sample x belong to class i

$$q_i(x) \coloneqq P(y = i|x) = \frac{P(x|y = i)P(y = i)}{P(x)}$$

- Bayes decision rule:

  If $q_i(x) > q_j(x)$, then $y = i$, otherwise $y = j$

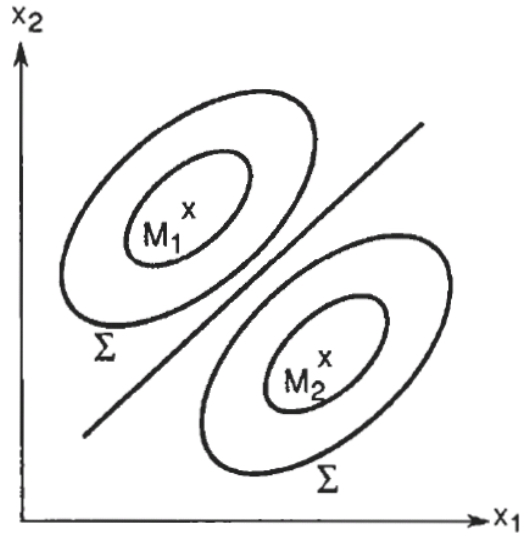
- We can also use $h(\mathrm{x}) = -\ln \frac{q_i(x)}{q_j(x)}$ and compare it with 0

- This is equivalent to the following decision boundary

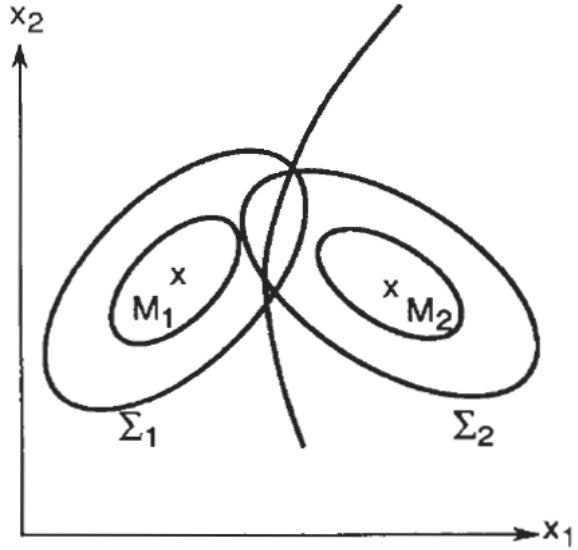
  If likelihood ratio $l(x) = \frac{P(x|y=i)}{P(x|y=j)} > \frac{P(y=j)}{P(y=i)}$, then $y = i$, otherwise $y = j$

Example: Gaussian class conditional distribution

Depending on assumptions made for the likelihood, the decision boundary can be very different



LDA



QDA

Decision boundary: $\mathrm{h}(\mathrm{x}) = -\ln \frac{q_i(x)}{q_j(x)} = 0$

Linear discriminant analysis (LDA)

$$P(x|y=1) = N(x;\mu_1,\Sigma_1), \qquad p(y=1) = \pi_1$$
$$P(x|y=0) = N(x;\mu_2,\Sigma_2), \qquad p(y=0) = \pi_2 = 1-\pi_1$$

Assume $\Sigma_i = \Sigma, i = 1,2$

$$h(x) = -\ln\frac{q_1(x)}{q_2(x)}$$
$$= -(\mu_1-\mu_2)^T\Sigma^{-1}x + \left(\frac{1}{2}\mu_1\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_2\Sigma^{-1}\mu_2\right) - \log\pi_1 + \log\pi_2$$

If $h(x) < 0$, decide class 1

Quadratic discriminant analysis (QDA)

$$P(x|y=1) = N(x;\mu_1,\Sigma_1), \qquad p(y=1) = \pi_1$$
$$P(x|y=0) = N(x;\mu_2,\Sigma_2), \qquad p(y=0) = \pi_2 = 1-\pi_1$$

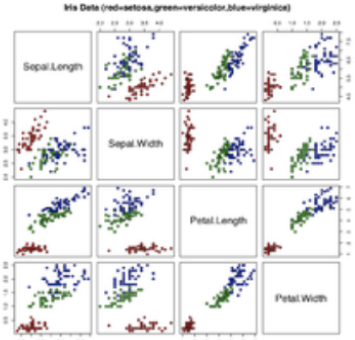
Allow the covariance for each class to be different $\Sigma_i, i = 1,2$

$$h(x) = -\ln\frac{q_1(x)}{q_2(x)}$$
$$= \frac{1}{2}x^T(\Sigma_1^{-1} - \Sigma_2^{-1})x - (\mu_1^T\Sigma_1^{-1} - \mu_2^T\Sigma_2^{-1})x + \left(\frac{1}{2}\mu_1\Sigma_1^{-1}\mu_1 - \frac{1}{2}\mu_2\Sigma_2^{-1}\mu_2\right)$$
$$-\log\pi_1 + \log\pi_2$$

If $h(x) < 0$, decide class 1

Fisher's Iris Example

- The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher (1936) as an example of discriminant analysis.
- The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.
- Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other



- Demo:

Flower: https://wnellie.tumblr.com/post/143155937907/analysis-of-the-famous-iris-flower-dataset-part
Fisher: https://www.umass.edu/wsp/resources/tales/fisher.html
Data set image : https://en.wikipedia.org/wiki/Iris_flower_data_set

https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html#sphx-glr-auto-examples-decomposition-plot-pca-vs-lda-py

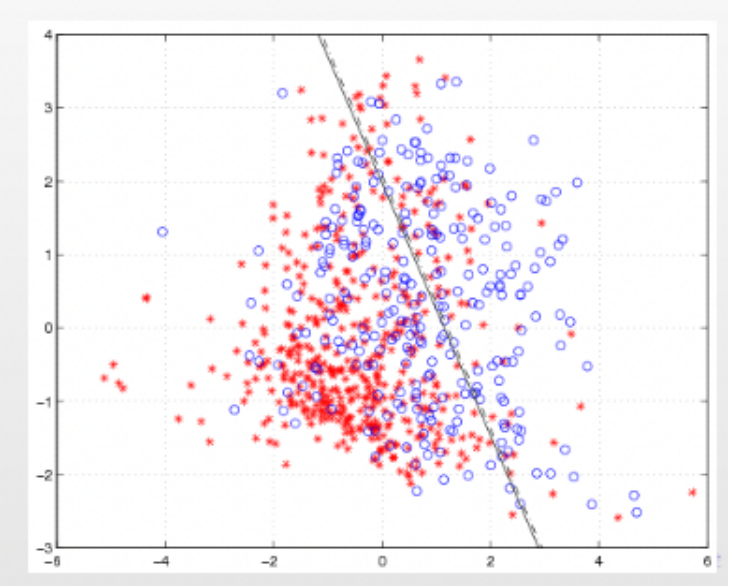
Georgia Tech

Example: diabetes dataset

Without diabetes: stars (class 1), with diabetes: circles (class 0).

Solid line: classification boundary obtained by LDA.

Dash dot line: boundary obtained by linear regression of indicator matrix.



Diabetes dataset (cont)

Two input variables computed from the principle components of the original 8 variables

Prior probabilities: $\hat{\pi}_1 = 0.651, \hat{\pi}_2 = 0.349$

Mean vectors $\hat{\mu}_1 = [-0.4035, -0.1935]^T, \hat{\mu}_2 = [0.7528, 0.3611]^T$

$$\hat{\Sigma} = \begin{bmatrix} 1.7925 & -0.1461 \\ -0.1461 & 1.6634 \end{bmatrix}$$

LDA decision rule

$$f^{LDA}(x) = \begin{cases} 1 & 1.1443 - x_1 - 0.5802 x_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

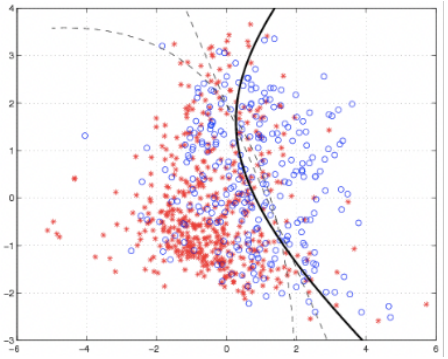
Within training data classification error rate 28.26%

Diabetes dataset (cont)

QDA: Everything else stay the same, except that

$$\hat{\Sigma}_1 = \begin{bmatrix} 1.6769 & -0.0461 \\ -0.0461 & 1.5964 \end{bmatrix}, \hat{\Sigma}_2 = \begin{bmatrix} 2.0087 & -0.3330 \\ -0.3330 & 1.7887 \end{bmatrix}$$

Within training data classification error rate: 29.04%



Georgia Tech

Naïve Bayes Classifier

- Use Bayes decision rule for classification
- Simplify by assuming all features (dimensions of the vector x) are independent given the label
- Thus likelihood $p(x|y=1)$ is fully factorized

$$p(x|y=1) = \prod_{i=1}^{n} p(x_i|y=1)$$

- Similarly for $p(x|y=0)$

Naïve Bayes Classifier

- Simplify by assuming all features (dimensions of the vector x) are independent given the label
- Thus likelihood $p(x|y=1)$ is fully factorized

$$p(x|y=1) = \prod_{i=1}^{n} p(x_i|y=1)$$

- Similarly for $p(x|y=0)$
- Example: email spam filtering

Example: Email spam filtering

- Suppose we want to train a classifier to identify an email as spam/non-spam
- Each email is represented using the bag-of-words model
- Volcabulary

  {secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza}.

- Training data

spam

- million dollar offer
- secret offer today
- secret is secret

non-spam

- low price for valued customer
- play secret sports today
- sports is healthy
- low price pizza

Email spam filtering (cont.)

- Assuming n keywords are independent of each other
- **Likelihood** of a sentence with its feature vector x given class $y = c$:

$$P(x|y = c) = \prod_{k=1}^{n} \theta_{c,k}^{x_k}, \quad c = 0, 1$$

- $\theta_{c,k} \in [0,1]$ is the probability of word k appearing in class c

Constraint

$$\theta_{0,k} + \theta_{1,k} = 1, \forall k$$

- Training: maximizing log likelihood function of the training data

$$\ell\left(\theta_{c,k}, k = 1, \dots, n\right) = \sum_{i=1}^{m} \sum_{k=1}^{n} x_k^i \log \theta_{y^i,k} \ , c = 0, 1$$

Email spam filtering (cont.)

- Solving a constrained maximization problem to find the estimator $\{\hat{\theta}_{c,k}\}$

$$\max_{\theta} \ \ell\left(\theta_{c,k}, c = 0,1, k = 1, \dots, n\right) = \sum_{i=1}^{m} \sum_{k=1}^{n} x_k^i \log \theta_{y^i,k}$$

subject to $\theta_{0,k} + \theta_{1,k} = 1, k = 1, \dots, n$

- Apply this to a new email: z $\in R^n$
- Evaluate the likelihood $P(z|y = c) = \prod_{k=1}^{n} \hat{\theta}_{c,k}^{z_k}, \ c = 0, 1$

Email spam filtering (cont.)

- Evaluate prior probability: $P(y=0)=\frac{4}{7}=1-P(y=1)$
- Evaluate the posterior

$$q_c(z) \coloneqq P(y=c|z)=\frac{P(z|y=c)P(y=c)}{P(z|y=0)P(y=0)+P(z|y=1)P(y=1)}$$

- Bayes decision rule:

  If $q_1(x) > q_0(x)$, then $y=1$, “spam”!

  otherwise $y=0$, “no-spam”.

Classification algorithms

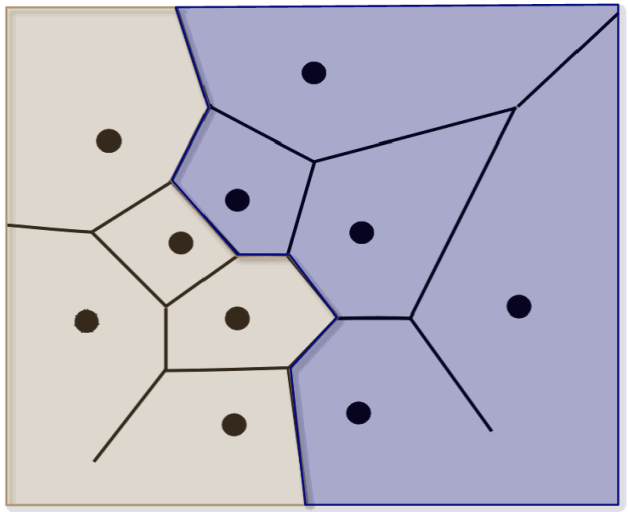
- Bayes classifier
- K-nearest neighbors
- Logistic regression

Nearest neighbor classifier

- The **nearest neighbor classifier**:

  assign x the same label as the closest training samples x^i

- The nearest neighbor rule defines a **Voronoi partition** of the feature space



- Nonparametric
- Nonlinear
- Easy to kernelize

K-nearest neighbors

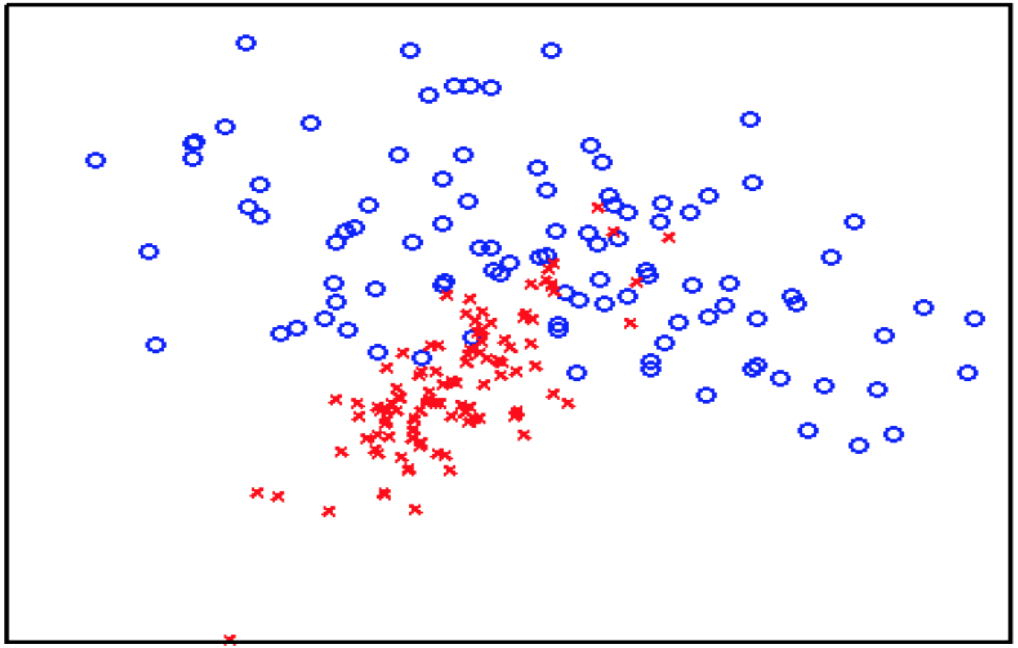
***K*-nearest neighbor classifier:** assign x a label by taking a **majority vote** over the K training points x^i closest to x

To define this more mathematically:

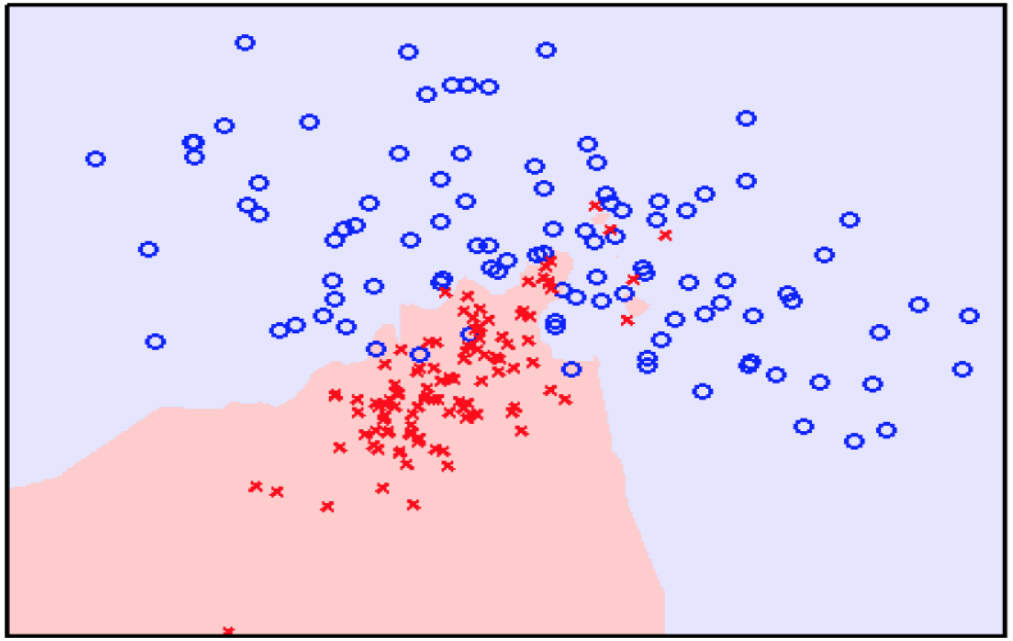
- $I_k(x) \coloneqq$ indices of the k training points closest to x.
- When labels $y^i = \pm 1$, then we can write the k-nearest neighbor classifier as:

$$\underbrace{f(x)}_{\text{predicted label for test point}} \coloneqq sign\left(\sum_{i \in I_k(x)} y^i\right)$$

Example

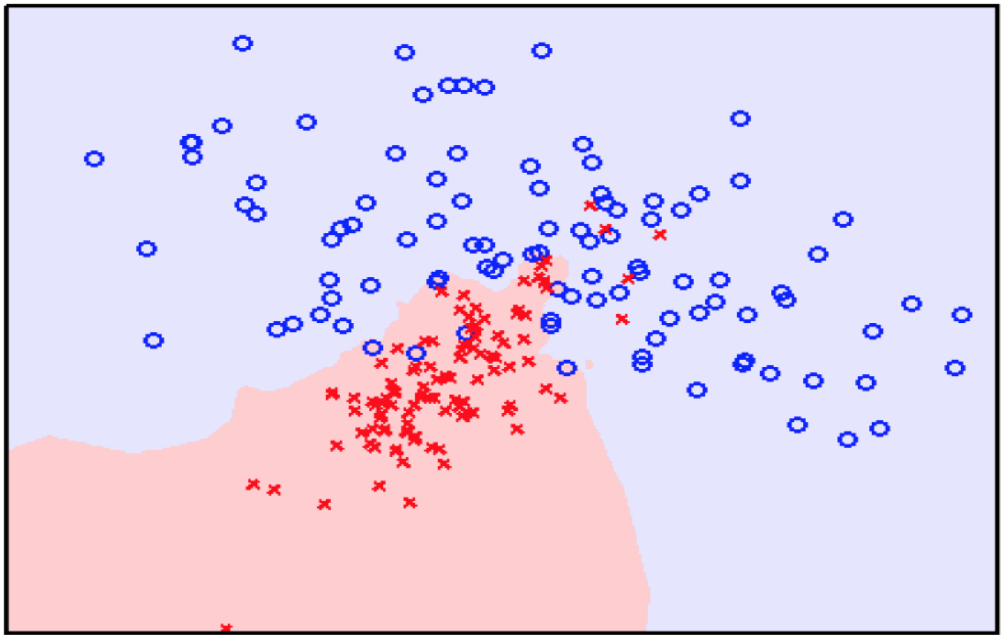


Example: decision boundary change as K changes



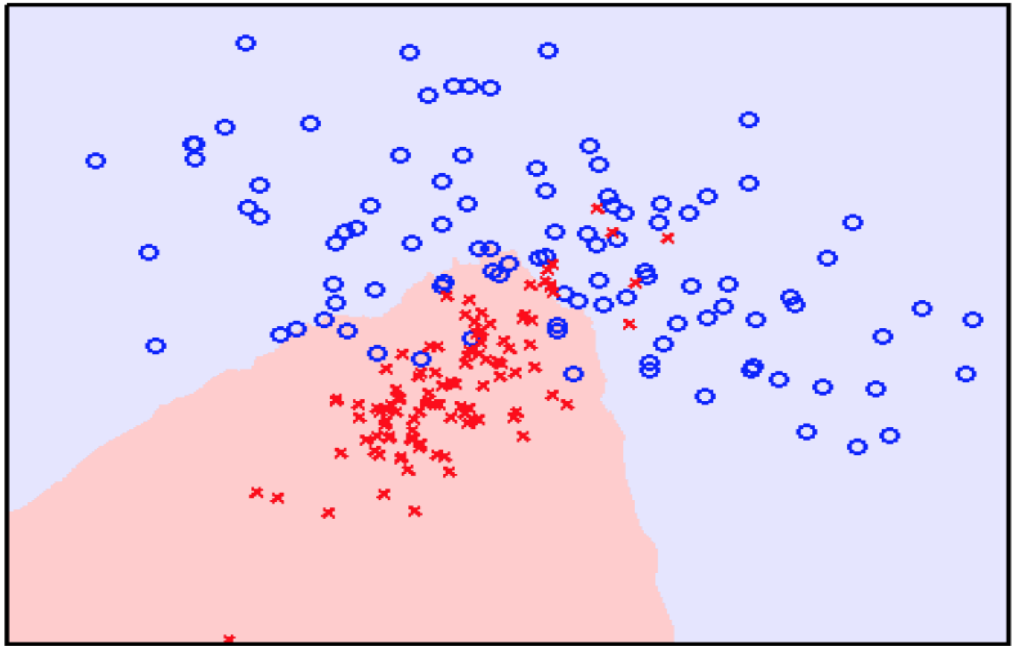
K = 3

Example



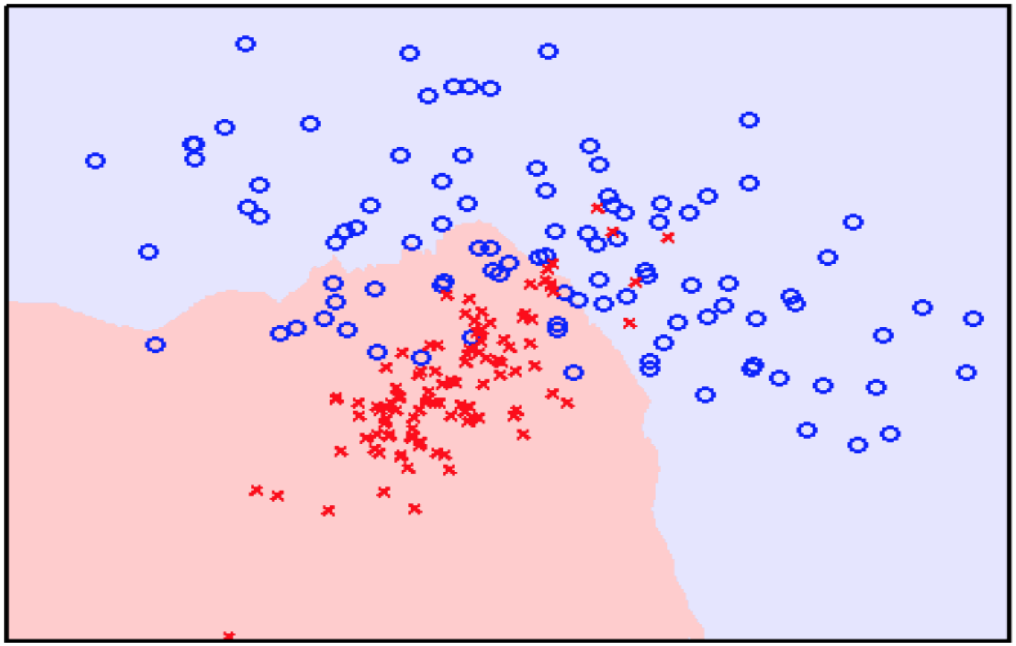
K = 5

Example



K = 25

Example



K = 51

Classification algorithms

- Bayes classifier
- K-nearest neighbors
- Logistic regression

Classification from a generative model

- Assume data are from a *statistical* generative model determined by
  - For each data point
  - Sample the value of x^i according to a prior $p(x)$
  - Sample a label y^i from the conditional probability $p\left(y\middle|x^i,\theta\right), y = 0, 1$
- Logistic regression: specify $p\left(y\middle|x^i,\theta\right)$ using logistic function
  - logistic regression model

$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^\top x)}$$

  - Note that

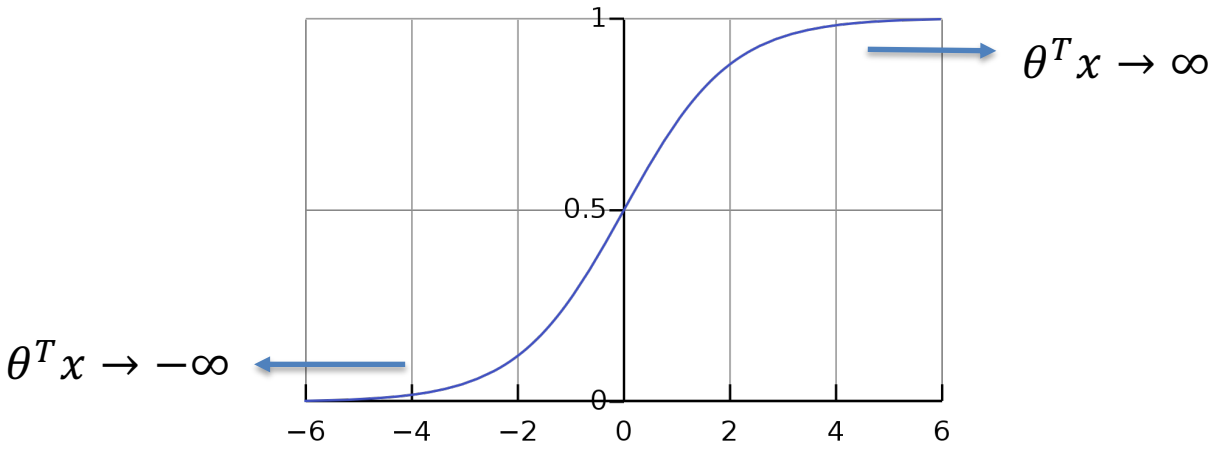
$$p(y = 0|x, \theta) = 1 - \frac{1}{1 + \exp(-\theta^\top x)} = \frac{\exp(-\theta^\top x)}{1 + \exp(-\theta^\top x)}$$

A closer look at logistic regression model

- Assume that the posterior distribution $p(y = 1|x)$ take a particular form (link function)

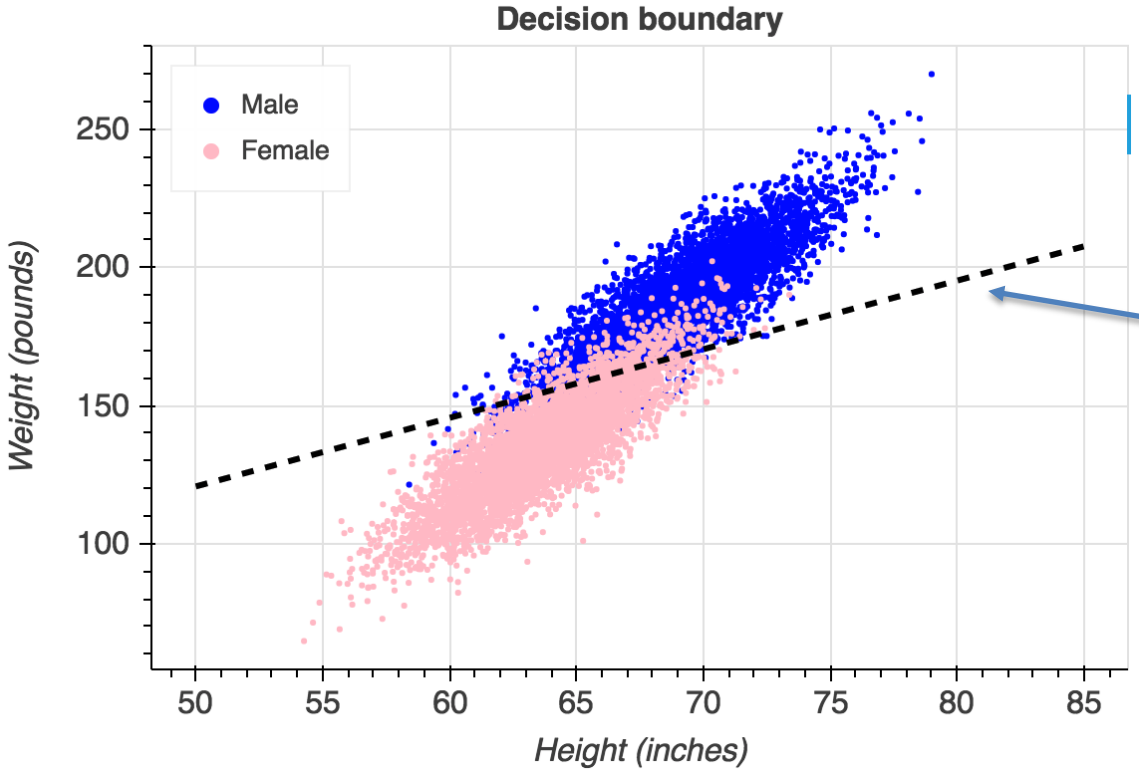
$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^{\mathsf{T}} x)}$$

- Logistic function (sigmoid function) $f(u) = \frac{1}{1+\exp(-u)}$



- Classification: given a new data z, $p\left(y = 1\middle|z, \hat{\theta}\right) > \eta$, class 1. Usually threshold $\eta = 0.5$

Decision boundary of logistic regression

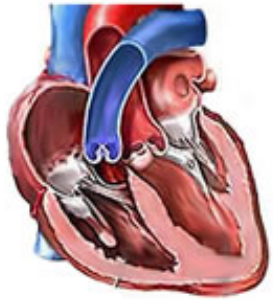


Change θ will change this decision boundary

Learning in logistic regression is to find θ to optimally separate the classes in training data

Demo: Statlog (Heart) Data Set

- 13 attributes (see heart.docx for details)
  - 2 demographic (age, gender)
  - 11 clinical measures of cardiovascular status and performance
- 2 classes: absence (1) or presence (2) of heart disease
- 270 samples
- Dataset taken from UC Irvine Machine Learning Repository
- Demo code



Comparing three methods

	Bayes classifier (Gaussian)	KNN	Logistic
Number of parameters	$O(n^2)$	1 (non-parametric)	$O(n)$
Robustness to model mismatch	No	Yes	relatively
Noisy prediction?	No	Yes	Yes (when logistic function value is close to 0.5)

The actual performance may vary case-by-case.

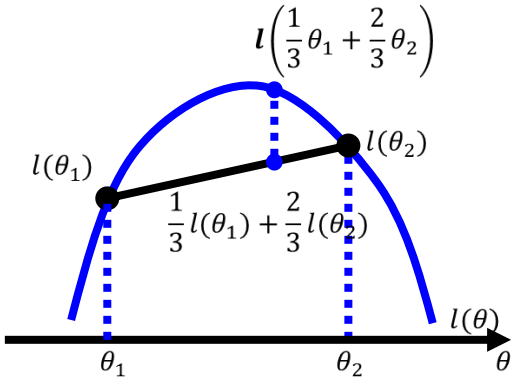
Learning parameters in logistic regression

Find θ, such that the conditional likelihood of the labels is maximized

$$\max_{\theta} l(\theta) := \log \prod_{i=1}^{m} P\left(y^i \middle| x^i, \theta\right)$$

Good news: $l(\theta)$ is concave function of θ, and there is a single global optimum.

Bad new: no closed form solution
(resort to numerical method)



Learning parameters in logistic regression

logistic regression model

$$p(y = 1|x, \theta) = \frac{1}{1 + \exp(-\theta^\top x)}$$

Note that

$$p(y = 0|x, \theta) = 1 - \frac{1}{1 + \exp(-\theta^\top x)} = \frac{\exp(-\theta^\top x)}{1 + \exp(-\theta^\top x)}$$

Plug in

$$l(\theta) := \log \prod_{i=1}^{m} P\left(y^i \middle| x^i, \theta\right)$$

$$= \sum_i (y^i - 1)\, \theta^\top x^i - \log(1 + \exp\left(-\theta^\top x^i\right))$$

Gradient of $l(\theta)$

$$l(\theta) := \log \prod_{i=1}^{m} P\left(y^i \middle| x^i, \theta\right)$$

$$= \sum_i (y^i - 1)\, \theta^\top x^i - \log\left(1 + \exp\left(-\theta^\top x^i\right)\right)$$

Gradient

$$\frac{\partial l(\theta)}{\partial \theta} = \sum_i (y^i - 1)\, x^i + \frac{\exp\left(-\theta^\top x^i\right) x^i}{1 + \exp(-\theta^\top x^i)}$$

Setting it to 0 does not lead to closed form solution

A big digression on optimization:

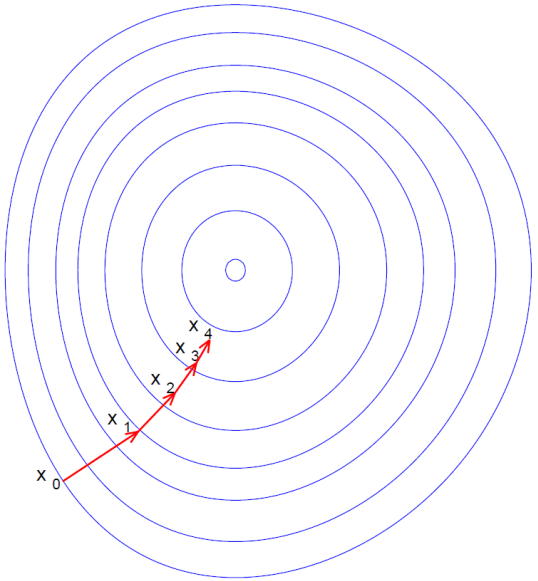
Stochastic gradient descent (SGD)

Gradient descent to minimize $f(x)$

- One way to solve an *unconstrained* optimization problem is gradient descent
- Given an initial guess, we *iteratively* refine the guess by taking the direction of the negative gradient
- Think about going down a hill by taking the steepest direction at each step
- Update rule

  $x^{t+1} = x^t - \gamma_t \nabla f(x^t)$

  $\gamma_t > 0$ is called the step size or learning rate



Gradient Ascent/Descent Algorithm

Initialize parameter θ^0

Do

gradient of $\nabla l(\theta)$ since we are solving maximization

$$\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_i (y^i - 1)\, x^i + \frac{\exp\left(-\theta^{t\top} x^i\right) x^i}{1 + \exp\left(-\theta^{t\top} x^i\right)}$$

While the $||\theta^{t+1} - \theta^t|| > \epsilon$

Batch gradient vs stochastic gradient

- The algorithm on the previous page is called the batch gradient descent
- To compute the gradient at each iteration, we need to sum over *all* data points in the dataset
- What if we have a huge dataset? 1million data point?
- Note that gradient de-couples: it consists of sum of evaluations over individual samples

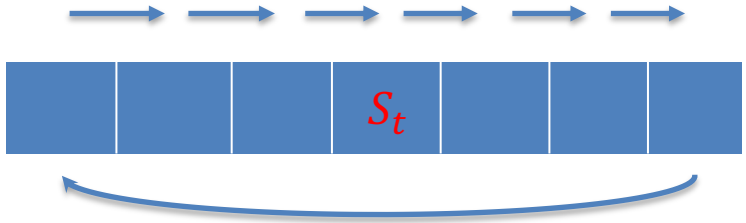
$$\nabla l(\theta) = \sum_i (y^i - 1)\, x^i + \frac{\exp(-\theta^\top x^i) x^i}{1+\exp(-\theta^\top x^i)}$$

Stochastic gradient descent

- At each iteration, we randomly sample a small subset S_t data point $(x_i, y_i), i \in S_t$ (in the extreme case, there is just one sample in S_t)
- Use gradient estimated using a small subset of data (instead of full data)

$$\nabla \hat{l}(\theta) = \sum_{i \in S_t} (y^i - 1) x^i + \frac{\exp(-\theta^\top x^i) x^i}{1 + \exp(-\theta^\top x^i)}$$

- Each iteration use a different subset $S_t, t = 1, 2, \ldots.$
- Eventually loop through the entire training data, and may loop through the data multiple times



Step sizes and stopping criterion

- For SGD, due to noise introduced by random sampling data, need to use decreasing step size, such as $O\left(\frac{1}{t}\right)$
  - Step size too small: not making too much progress
  - Step size too large: overshoot

- The optimality condition is $\nabla l(\theta) = 0$

- When $\|\nabla l(\theta)\|$ or $\|\nabla l(\theta)\|/\|\nabla l(\theta_0)\|$ is smaller than a pre-specified tolerance value, stop when the gradient becomes small $\|\nabla l(\theta)\|$, e.g., *l*-2 norm

Multiclass logistic regression

Assign input vector $x^i, i = 1, \dots, m$ into one of classes $c, c = 1, \dots, C$

Assume that the posterior distribution take a particular form:

$$P(y^i = c | x^i, \theta_1, \dots, \theta_C) = \frac{\exp\left(\theta_c{}^\top x^i\right)}{\sum_{c'} \exp\left(\theta_{c'}{}^\top x^i\right)}$$

Fit parameters $\{\theta_c\}, c = 1, \dots, C$

Solving maximum likelihood

Implementation

Look at logistic regression:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
Nearest neighbors:
https://scikit-learn.org/stable/modules/neighbors.html
Naïve Bayes:
https://scikit-learn.org/stable/modules/naive_bayes.html


SUMMARY