Machine Learning
DSECL ZG565

Dr. Monali Mavani

BITS Pilani
Pilani Campus

Session Content

- Decision Theory - Bishop
- Probabilistic Generative Model versus Probabilistic Discriminative Model
- Logistic Regression –Bayesian Analysis (New chapter Tom Mitchell)
- Estimating Parameters Linear Regression : Closed form solution
- Linear basis function models (3.1 Bishop)
- Evaluation metrics

Decision Theory

Decision Theory

We have broken the **classification problem** down into two separate stages

- **Inference step**: Determine $p(\mathbf{x}, t)$ OR $p(\mathbf{x}|C_k)$ from training data. **(For regression problems t is** continuous variables, whereas for classification problems **t** represents class labels.)

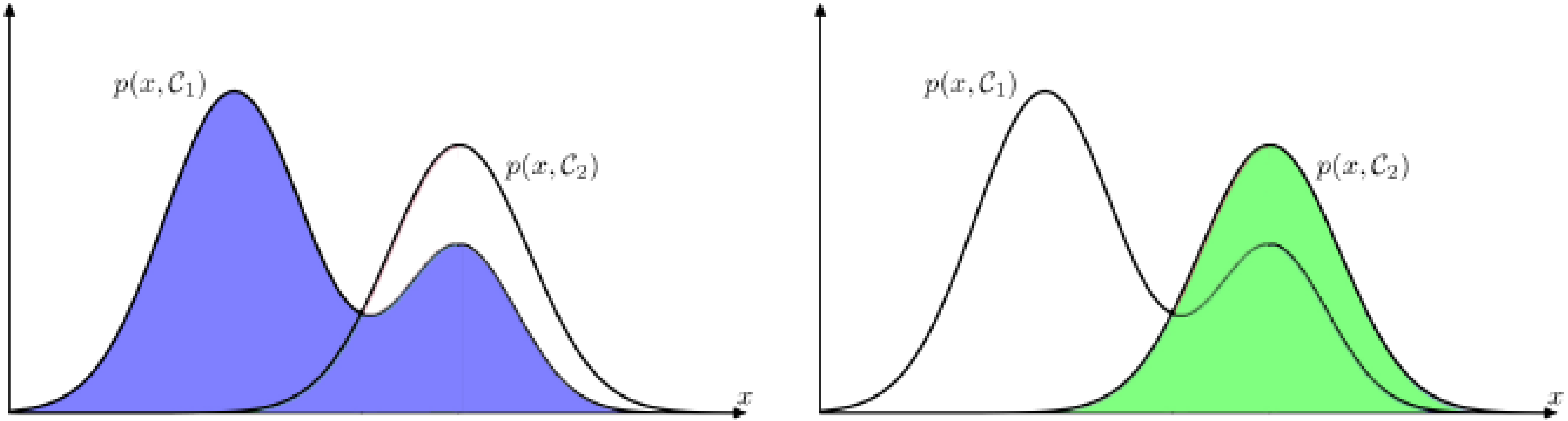
$$t = \arg\max_k \{ \underbrace{p(x|C_k)}_{Likelihood} \}$$

- **Decision step**: Determine optimal t for test input x: how to make optimal decisions given the appropriate probabilities.

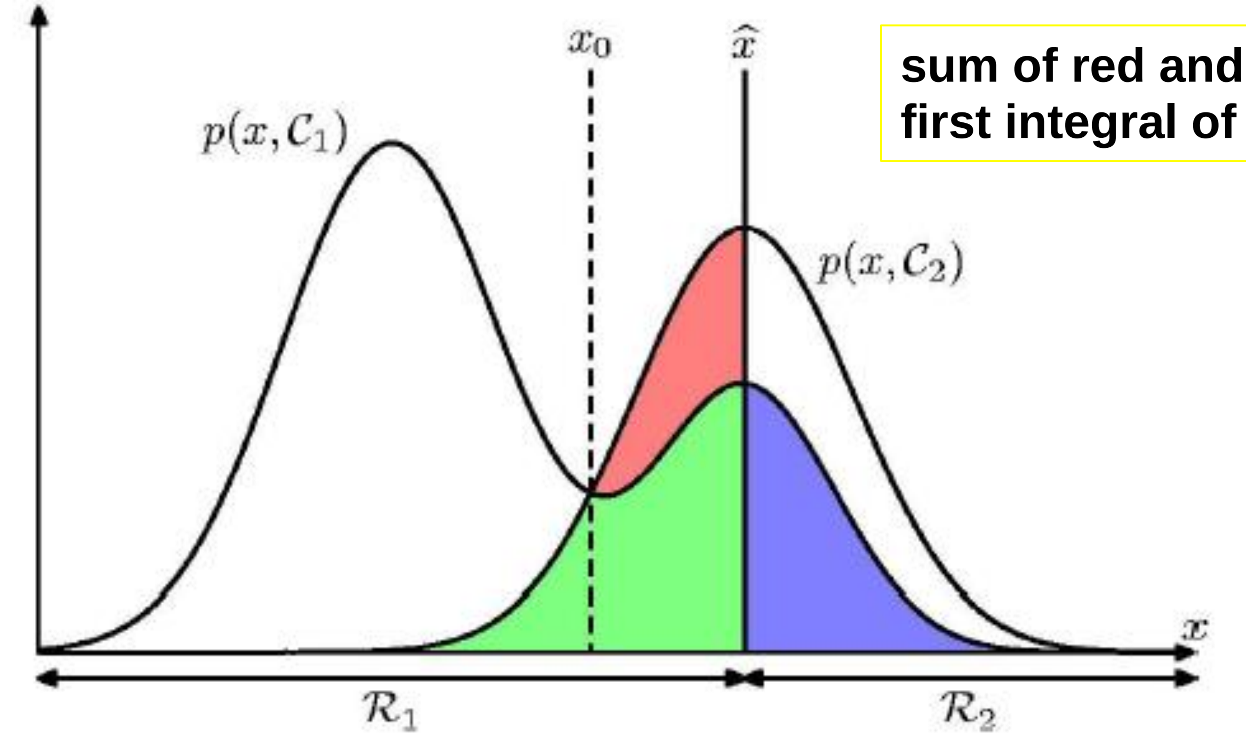
$$\begin{tikzcd}
x \arrow[r, "\text{inference}"] \arrow[rr, bend right, "\text{discriminant function}"'] & p(\mathcal{C}_k, x) \arrow[r, "\text{decision}"] & \mathcal{C}_k
\end{tikzcd}$$

Minimum Misclassification Rate

- Divide the input space into regions Rk called decision regions, one for each class, such that all points in Rk are assigned to class Ck
- Boundaries between decision regions are called decision boundaries or decision surfaces
- A mistake occurs when an input vector belonging to class C1 is assigned to class C2 or vice versa.



Minimum Misclassification Rate



sum of red and green is the first integral of equation

$\hat{x}$: decision boundary.
x_0: optimal decision boundary

$x_0 : \arg\min_{\mathcal{R}_1} \{p(\text{mistake})\}$

$$p(\text{mistake}) = p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) = \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\,\mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\,\mathrm{d}\mathbf{x}.$$

- Minimum error decision rule
- To minimize p(mistake) we should arrange that each **x** is assigned to whichever class has the smaller value of the integrand
- if $p(\mathbf{x}, C1) > p(\mathbf{x}, C2)$ for a given value of **x**, then we should assign that **x** to class $C1$.
- Since $p(x,C_k)=p(C_k|x)p(x)$, choose class for which *a posteriori* probability is highest
  - Called Bayes Classifier

BITS Pilani

Minimum Misclassification Rate

General case of *K* classes

$$
\begin{aligned}
p(\text{correct}) &= \sum_{k=1}^{K} p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\
&= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k)\, \mathrm{d}\mathbf{x}
\end{aligned}
$$

BITS Pilani

Inference and Decision

Three distinct approaches to Decision Problems

1. Generative
2. Discriminative
3. Discriminant Function

1. Generative Models

- Model class-conditional pdfs and prior probabilities
- First solve inference problem of determining class-conditional densities P(X|Y), for each class separately
- Then use Bayes theorem to determine posterior probabilities
- Then use decision theory to determine class membership
- “Generative” since sampling can generate synthetic data points
- Use the capacity of the model to characterize how the data is generated (both inputs and outputs)
- explicitly models the **actual distribution of each class**.
- Gaussians, Naïve Bayes, Mixtures of multinomials , Mixtures of Gaussians, Bayesian networks,

2. Discriminative Models

- Directly estimate posterior probabilities P(Y|X) directly from training data and use decision theory to assign each new x to one of the classes

- No attempt to model underlying probability distributions

- models the **decision boundary between the classes**

- Logistic regression, SVMs , **tree based classifiers (e.g. decision tree)** Traditional neural networks, Nearest neighbor

3. Discriminant Functions

- Find a function f (x) that maps each input x
- directly to class label
  - In two-class problem, f (.) is binary valued
    - f =0 represents class C1 and f =1 represents class C2
- Probabilities play no role
  - No access to posterior probabilities p(Y|X)
  - Linear discriminant, Fisher Linear Disc, Perceptron

Naive Bayes and logistic regression: two different modelling paradigms

Spam classification problem

First Strategy: discriminative (e.g., logistic regression)

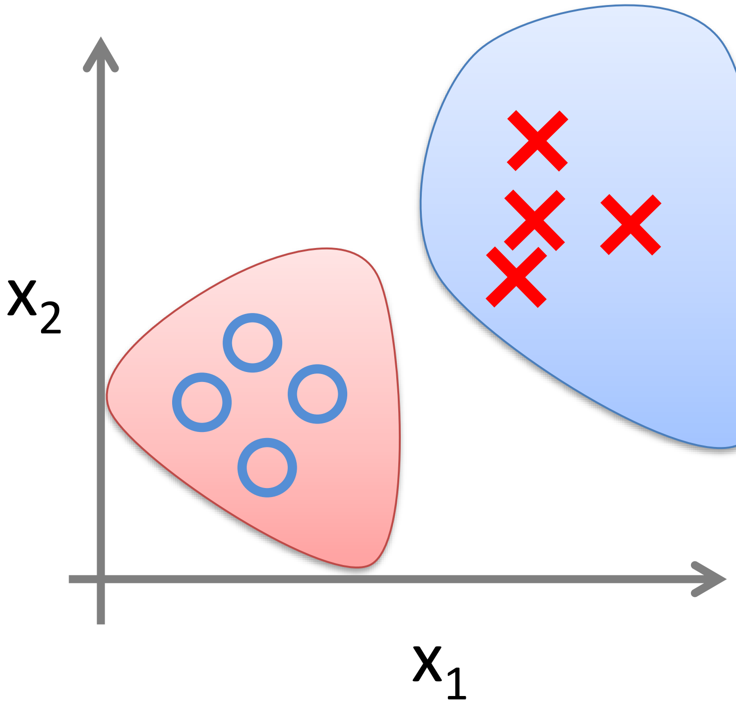
- Use training set to find a decision boundary in the feature space that separates spam and non-spam emails
- Given a test point, predict its label based on which side of the boundary it is on.

Second Strategy: generative (e.g., naive bayes)

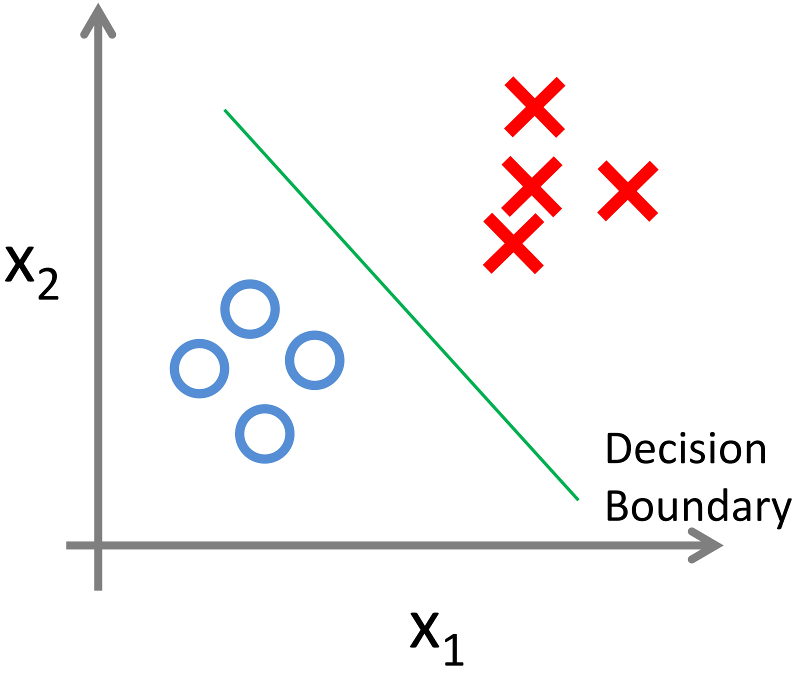
- Look at spam emails and build a model of what they look like.
- Similarly, build a model of what non-spam emails look like.
- To classify a new email, match it against both the spam and non-spam models to see which is the better fit.

Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative:



Discriminative:



Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative	Discriminative
Ex: Naïve Bayes	Ex: Logistic Regression
Estimate $P(Y)$ and $P(X\|Y)$	Finds class label directly $P(Y\|X)$
Prediction $\hat{y} = \text{argmax}_y\, P(Y = y)P(X = x\|Y = y)$	Prediction $\hat{y} = P(Y = y\|X = x) = \frac{1}{1+e^{-\theta^\top x}}$
Decision boundary	Probability distributions of the data

Logistic Regression –Bayesian Analysis

Logistic Regression and Gaussian Naïve Bayes Classifier

- Interestingly, the parametric form of P(Y|X) used by Logistic Regression is precisely the form implied by the assumptions of a Gaussian Naive Bayes classifier.

- Therefore, we can view Logistic Regression as a closely related alternative to GNB, though the two can produce different results in many cases

Estimating Parameters - Learning Logistic Regression

- Estimation of n+1 parameters say, $\theta = (\theta_0, \theta_1, \ldots\ldots, \theta_n)$ for which the probability of the observed data is the maximum on the training data set.
  - generative learning
    - the Gaussian Naive Bayes assumptions
  - discriminative learning
    - maximize the conditional data likelihood (or minimizing negative likelihood)
    - Uses gradient descent optimization

Bayesian inference

Bayesian inference for logistic analyses follows the usual pattern for all Bayesian analyses:

1. Write down the likelihood function of the data
2. Form a prior distribution over all unknown parameters.
3. Use Bayes theorem to find the posterior distribution over all parameters.

Where does the hypothesis function come from?

- Logistic regression hypothesis representation

$$P(Y=1|X) = h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}} = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2+\cdots+\theta_n x_n)}}$$

- Consider learning f: $X \rightarrow Y$, where
  - X is a vector of real-valued features $[X_1, \cdots, X_n]^\top$
  - Y is Boolean
  - Assume all X_i are conditionally independent given Y
  - **Model likelihood $P(X_i|Y=y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$ and assume variance is independent of class, i.e. $\sigma_{i0} = \sigma_{i1} = \sigma_i$**

$$P(X_i|Y=y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} e^{-\frac{1}{2}\left(\frac{X_i-\mu_{ik}}{\sigma_{ik}}\right)^2}$$

$$\boxed{P(x|y_k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_i^2}}}$$

  - **Model prior $P(Y)$ as Bernoulli π : $P(Y=1) = \pi$ and $P(Y=0) = 1-\pi$**

$$\boxed{\text{What is } P(Y|X_1, X_2, \cdots, X_n)?}$$

Slide credit: Tom Mitchell

Logistic Regression –Bayesian Analysis

innovate achieve lead

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)}$$

Applying Bayes rule

$$P(Y=1|X) = \frac{1}{1+\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

Divide by $P(Y=1)P(X|Y=1)$

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

Apply $\exp(\ln(\cdot))$

By independence assumption:

$$\boxed{\frac{P(X|Y=0)}{P(X|Y=1)} = \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)}}$$

$P(Y=1)=\pi$ and $P(Y=0)=1-\pi$
by modelling P(Y) as Bernoulli

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{P(Y=0)}{P(Y=1)}+\ln\frac{P(X|Y=0)}{P(X|Y=1)})}$$

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

Logistic Regression –Bayesian Analysis

Plug in $P(X_i|Y)$

$$P(Y=1|X) = \frac{1}{1+\exp\left(\ln\frac{1-\pi}{\pi}+\sum_i\left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)\right)}$$

$$P(Y=1|X) = \frac{1}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)}$$

$$= \frac{1}{1+\exp(\theta_0+\sum_i \theta_i X_i)}$$

$$w_0 = \ln\frac{1-\pi}{\pi}+\sum_i\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}$$

$$w_i = \frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}$$

$$P(Y=0|X) = 1-P(Y=1|X) = \frac{\exp(w_0+\sum_{i=1}^n w_i X_i)}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)}$$

linear classification rule

$$\frac{P(Y=0|X)}{P(Y=1|X)} = \exp\left(w_0+\sum_i w_i X_i\right)$$

$$\ln\frac{P(Y=0|X)}{P(Y=1|X)} = w_0+\sum_i w_i X_i$$

$P(X_i|Y)$ derivation

$$
\begin{aligned}
\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i-\mu_{i1})^2}{2\sigma_i^2}\right)} \\
&= \sum_i \ln \exp\left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{(X_i^2-2X_i\mu_{i1}+\mu_{i1}^2)-(X_i^2-2X_i\mu_{i0}+\mu_{i0}^2)}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{2X_i(\mu_{i0}-\mu_{i1})+\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)
\end{aligned}
$$

$$
P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\sum_i\left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right))}
$$

Where does the **cost** come from? - Logistic regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}))$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \, \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

Learning: fit parameter θ

$$\min_{\theta} J(\theta)$$

Prediction: given new x

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

Slide credit: Andrew Ng

maximum conditional likelihood - θ_{MCLE}

- **Goal**: choose θ to maximize conditional likelihood of training data
  - $P_\theta(Y=1|X=x) = h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}$
  - $P_\theta(Y=0|X=x) = 1 - h_\theta(x) = \frac{e^{-\theta^\top x}}{1+e^{-\theta^\top x}}$

conditional likelihood

- **Training data** $\mathrm{D} = \left\{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(m)}, y^{(m)}\right)\right\}$
- **Data likelihood** $= \prod_{i=1}^{m} P_\theta\left(\left(x^{(i)}, y^{(i)}\right)\right)$
- **Data conditional likelihood** $= \prod_{i=1}^{m} P_\theta\left(y^{(i)}|x^{(i)}\right)$

$$\boxed{\theta_{\text{MCLE}} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} P_\theta\left(y^{(i)}|x^{(i)}\right)}$$

Slide credit: Tom Mitchell

Expressing conditional log-likelihood

$$L(\theta) = \log \prod_{i=1}^{m} P_\theta\left(y^{(i)}|x^{(i)}\right) = \sum_{i=1}^{m} \log P_\theta\left(y^{(i)}|x^{(i)}\right)$$

Recall, each label $y^{(i)}$ is binary with prob. $P_\theta\left(y^{(i)}|x^{(i)}\right)$: Assume Bernoulli likelihood: (use PMF of Bernoulli dist.)

$$= \sum_{i=1}^{m} y^{(i)} \log P_\theta\left(y^{(i)} = 1|x^{(i)}\right) \left(1 - y^{(i)}\right) \log P_\theta\left(y^{(i)} = 0|x^{(i)}\right)$$

$$= \sum_{i=1}^{m} y^{(i)} \log \left(h_\theta(x^{(i)})\right) + \left(1 - y^{(i)}\right) \log(1 - h_\theta(x^{(i)}))$$

To turn Logliklihood into loss function flip the sign

$$\boldsymbol{\theta_{MCLE} = \underset{\theta}{argmax} \prod_{i=1}^{m} P_\theta\left(y^{(i)}|x^{(i)}\right) = -\underset{\theta}{\text{argmin}} \prod_{i=1}^{m} P_\theta\left(y^{(i)}|x^{(i)}\right)}$$

$$\boldsymbol{\theta_{MCLE}} = -\underset{\theta}{\textbf{argmin}} \sum_{i=1}^{m} y^{(i)} \log \left(h_\theta(x^{(i)})\right) + \left(1 - y^{(i)}\right) \log(1 - h_\theta(x^{(i)}))$$

Learning Model Parameters – Closed Form Solution (using vectorization)

Vectorization

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model:

$$h(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as $h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$

Vectorization

- Consider our model for n instances:

$$h\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

$\mathbb{R}^{(d+1)\times 1}$ $\qquad$ $\mathbb{R}^{n\times(d+1)}$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

Vectorization

- For the linear regression cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{2n}\underbrace{(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^{\mathsf{T}}}_{\mathbb{R}^{1\times n}}\underbrace{(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})}_{\mathbb{R}^{n\times 1}}$$

$\mathbb{R}^{n\times(d+1)}$

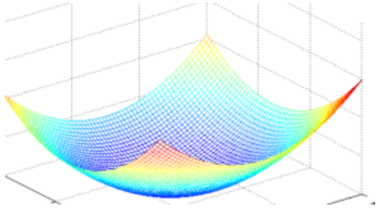
$\mathbb{R}^{(d+1)\times 1}$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Closed Form Solution

- Instead of using GD, solve for optimal $\boldsymbol{\theta}$ analytically
  - Notice that the solution is when $\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$



- Derivation:

1 x 1

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n} (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^\intercal (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})$$

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - \boxed{\boldsymbol{y}^\intercal \boldsymbol{X}\boldsymbol{\theta}} - \boxed{\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y}} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

Take derivative and set equal to 0, then solve for $\boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \cancel{\boldsymbol{y}^\intercal \boldsymbol{y}}) = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} - \boldsymbol{X}^\intercal \boldsymbol{y} = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} = \boldsymbol{X}^\intercal \boldsymbol{y}$$

Closed Form Solution: $\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$

Closed Form Solution

- Can obtain $\boldsymbol{\theta}$ by simply plugging X and $\boldsymbol{y}$ into

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \ldots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \ldots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \ldots & x_d^{(n)} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- If $X^\intercal X$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $d \leq n$

Gradient Descent vs Closed Form

Gradient Descent	Closed Form Solution
• Requires multiple iterations • Need to choose α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if n is large – Computing $(X^\top X)^{-1}$ is roughly $O(n^3)$

Linear Basis Function Models

Extending Linear Regression to More Complex Models

- The inputs **X** for linear regression can be:
  - Original quantitative inputs
  - Transformation of quantitative inputs
    - e.g. log, exp, square root, square, etc.
  - Polynomial transformation
    - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
  - Basis expansions
  - Dummy coding of categorical inputs
  - Interactions between variables
    - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques to fit non-linear datasets.

Linear Basis Function Models

- Basic Linear Model: $$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$
- Generalized Linear Model: $$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \phi_j(\boldsymbol{x})$$
- Once we have replaced the data by the outputs of the basis functions, fitting the generalized model is exactly the same problem as fitting the basic model
  - Unless we use the kernel trick – more on that when we cover support vector machines
  - Therefore, there is no point in cluttering the math with basis functions

Based on slide by Geoff Hinton

Linear Basis Function Models

- Generally,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

- Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions :

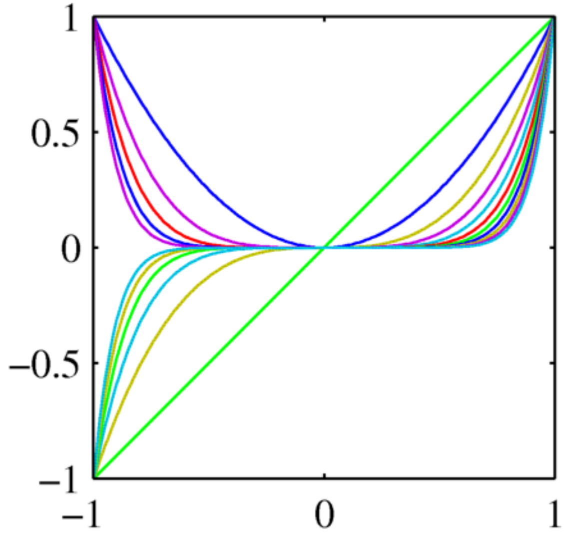
$$\phi_j(\boldsymbol{x}) = x_j$$

Linear Basis Function Models

- Polynomial basis functions:

$$\phi_j(x) = x^j$$

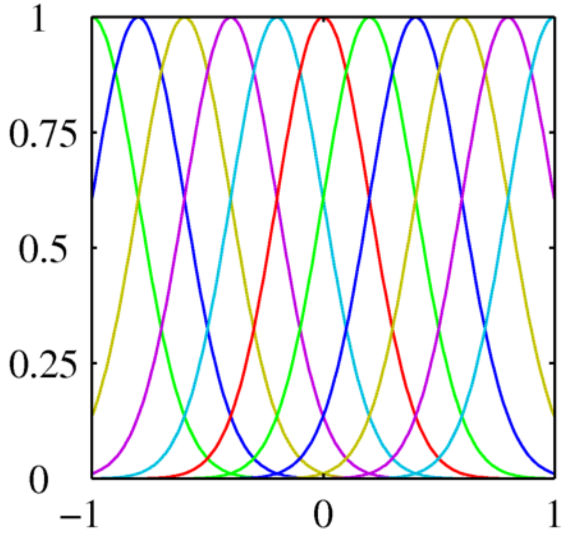
  - These are global; a small change in x affects all basis functions



- Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

  - These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



Based on slide by Christopher Bishop (PRML)

Linear Basis Function Models

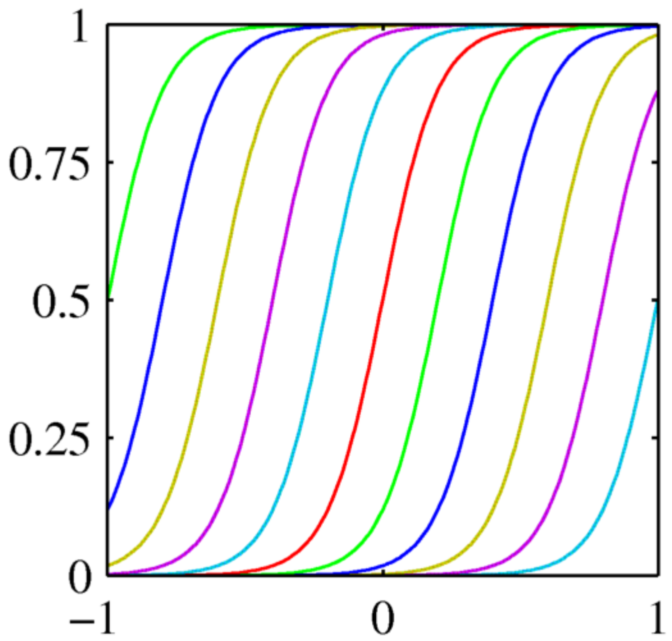
- Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

  - These are also local; a small change in x only affects nearby basis functions. μ_j and s control location and scale (slope).



By using nonlinear basis functions, we allow the function *y*(**x**,**w**) to be a nonlinear function of the input vector **x**. They are called linear models because this function is linear in **w**.

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- etc.

Evaluating Performance

- If y is discrete:
  - Accuracy: # correctly classified / # all test examples
    - Good for class balanced dataset
- Want evaluation metric to be in some range, e.g. [0 1]
  - 0 = worst possible classifier, 1 = best possible classifier

Confusion matrix

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Thanks