LINEAR MODELS FOR CLASSIFICATION (PART 1)

Hsin-Min Lu
盧信銘
台大資管系

Topics

- Introduction
- Discriminant Function
- Probabilistic Generative Models

- Probabilistic Discriminative Models

- Laplace Approximation
- Bayesian Logistic Regression

INTRODUCTION

Introduction

- Regression:
  - Map input vector x to one or more continuous target variables t
- Classification:
  - Map input vector x to one of K discrete classes $(K \geq 2)$
- Ordinal Regression:
  - Map input vector x to one of K discrete classes that have an ordering (e.g., on-line evaluation (stars))

Linear Classification Models

- Simple scenario: disjoint classes
- Divide input space into decision regions
- Linear model decision surface are linear function of input x
  - D-1 dimension hyperplane in D dimensional input space
- Linearly separable:
  - Classes in a dataset can be separated exactly using linear decision surface

Probabilistic Models

- Converting Regression to Class Output
- Target variables *t* now represent class labels
- Binary class representation is convenient
  - Two classes: $t \in \{0,1\}$, t=1 represent C_1; t=0 represent C_0
  - Interpret value of *t* as the probability that class is C_1
  - Probabilities need to take value between 0 and 1
  - For K>2, use 1-of-K coding scheme
    - E.g. K=5, $t = (0,1,0,0,0)^T$ represents class 2
    - Value t_k interpreted as probability of class C_k

Two Approaches to Classification

- Discriminant function
  - Directly assign x to a specific class
  - E.g. Fisher's linear discriminant function, perceptron
- Probabilistic models
  - Model $p(C_k|x)$ and make optimal decisions

- Probabilistic models separating inference from decision
  - Accommodate different loss function
  - Compensate for unbalanced data
  - Combine models

Probabilistic Classification Models

- Generative
  - Model class conditional densities by $p(x|C_k)$ together with prior probabilities $p(C_k)$
  - $p(C_k|x) \propto p(x|C_k)p(C_k)$
- Discriminative
  - Directly model conditional probabilities $p(C_k|x)$

Converting Linear Regression Models to Linear Classification Models

- Linear regression model y(x,w) is a linear function of parameters w
- In simplest case also a linear function of variable x
  - $y(x) = w^T x + w_0$
- For classification we wish to obtain a discrete output or posterior probabilities in range (0,1)
- Use generalization of this model $y(x) = f(w^T x + w_0)$
  - $f(\cdot)$ is an activation function
  - $f^{-1}(\cdot)$ is a link function

Generalized Linear Model

- $y(x) = f(w^T x + w_0)$
- The decision surface correspond to $y(x) = constant$ or $w^T x + w_0$=constant
- So decision boundaries are linear even if $f(\cdot)$ is nonlinear
  - ➔Generalized Linear Model
- No longer linear w.r.t to w due to $f(\cdot)$
  - Lead to more complex models for classification than regression

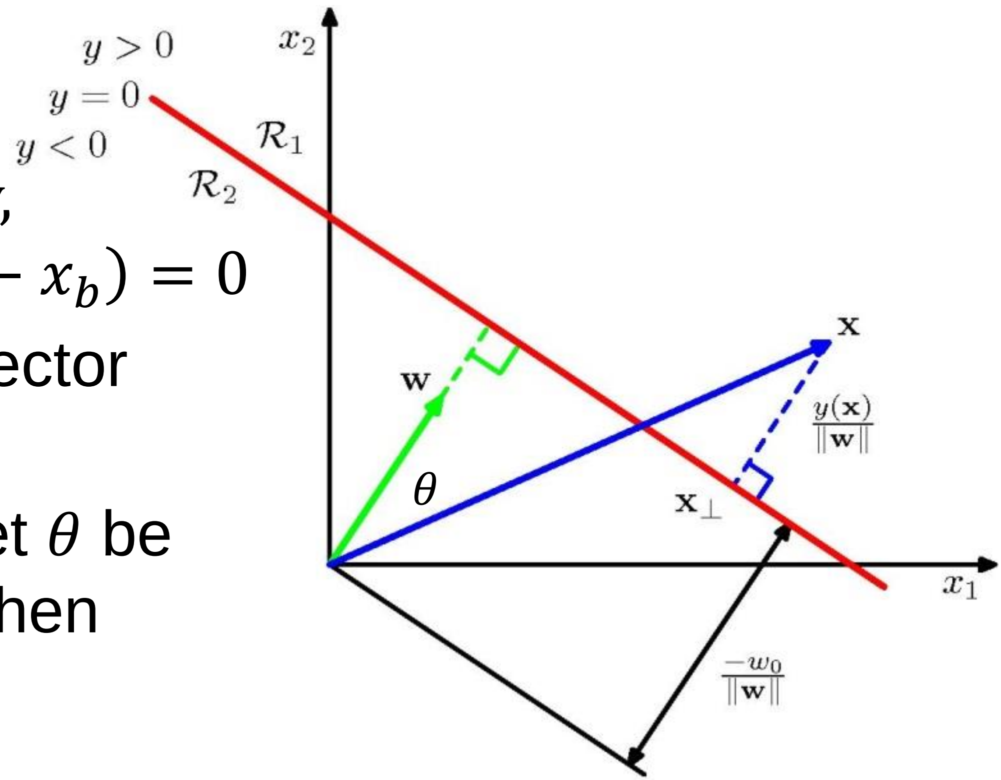
DISCRIMINANT FUNCTIONS

Linear Discriminant Function

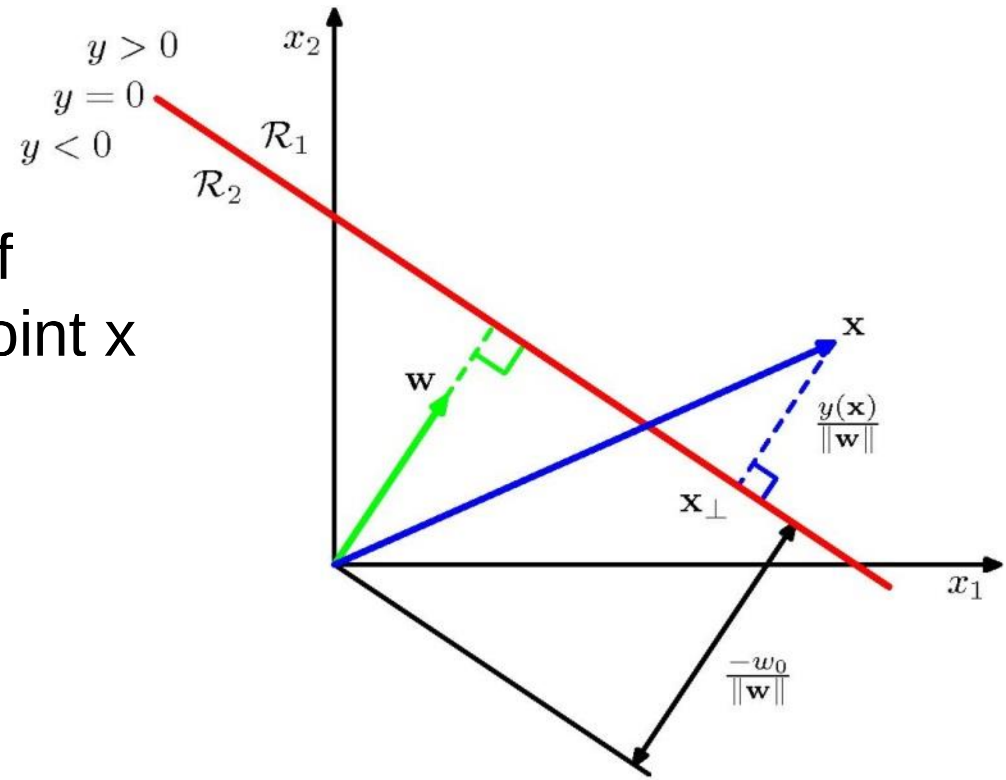
- Assigns input vector x to one of K classes denoted by C_k
- Restrict attention to linear discriminants (decision hyperplanes)
- Two-class linear discriminant function:
  - $y(x) = w^T x + w_0$
  - w is a weight vector and w_0 is bias
  - Assign x to C_1 if $y(x) \geq 0$, else C_2

Geometry of Linear Discriminant Functions

- $y(x) = w^T x + w_0$
- Decision boundary: $y(x) = 0$
- For x_a and x_b on the boundary, $y(x_a) - y(x_b) = 0$, ➔ $w^T(x_a - x_b) = 0$
- ➔ w is orthogonal to every vector on surface.
- Given an arbitrary vector x, let θ be the angle between x and w, then $\cos\theta = \frac{x \cdot w}{\|x\|\|w\|}$
- The projection length of x on w is $\|x\|\cos\theta = \frac{w^T x}{\|w\|} = \frac{y(x) - w_0}{\|w\|} = \frac{y(x)}{\|w\|} + \frac{-w_0}{\|w\|}$



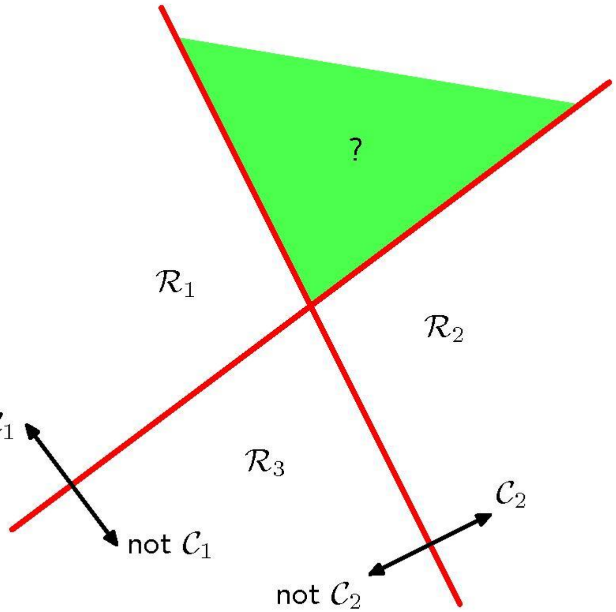
Geometry of Linear Discriminant Functions



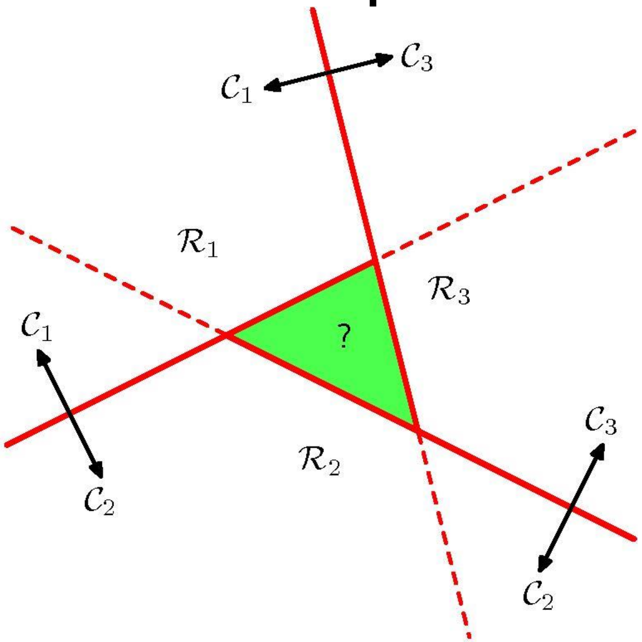
- $y(x) = w^T x + w_0$
- $y(x)$ gives signed measure of perpendicular distance r of point x to decision surface ($r = \frac{y(x)}{\|w\|}$)
- w_0 sets distance of origin to surface ($\frac{y(0)}{\|w\|} = \frac{w_0}{\|w\|}$)
- With dummy input $x_0 = 1$, and $\omega = (w_0, w^T)^T$, $z = (x_0, x^T)^T$, then $y(x) = \omega^T z$
  - Pass through origin in augmented D+1 dimensional space

Multiple Classes with 2-class classifiers

- One-versus-the-rest
- Build a K class discriminant using K-1 classifiers



- One-versus-one
- Build K(K-1)/2 binary discriminant classifiers, one for each pair



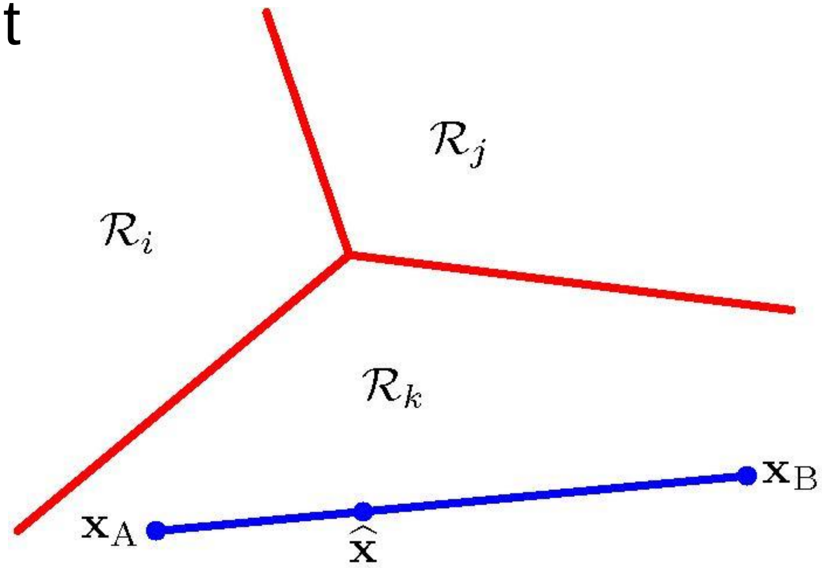
Both result in ambiguous regions of input space

Multiple Classes with K Discriminants

- Consider a single K class discriminants of the form
  - $y_k(x) = w_k^T x + w_{k0},\ k = 1,2,\dots,K$
  - Assign a point x to class C_k if $y_k(x) > y_j(x)$ for all $j \neq k$
- Decision boundary between class C_k and C_j is given by $y_k(x) = y_j(x)$
  - Corresponds to D-1 dimensional hyperplane defined by
  $$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0$$
  - Same form as the decision boundary for 2-class case
  - Decision regions of such a discriminant are always singly connected and convex
    - Proof on next slide

Convexity of Decision Regions

- Consider two points x_a and x_b both in decision region R_k
- Any point $\hat{x}$ on line connecting x_a and x_b can be expressed as $\hat{x} = \lambda x_a + (1 - \lambda) x_b$, where $0 \leq \lambda \leq 1$.
- From linearity of discriminant functions $y_k(x) = w_k^T x + w_{k0}$
- Combining the two, we have $y_k(\hat{x}) = \lambda y_k(x_a) + (1 - \lambda) y_k(x_b)$
- Because x_a and x_b lie inside R_k, it follows that $y_k(x_a) > y_j(x_a)$ and $y_k(x_b) > y_j(x_b)$ for all $j \neq k$.
- Hence $\hat{x}$ also lies inside R_k
- Thus R_k is singly-connected and convex



Learning the Parameters of Linear Discriminant Functions

- Three methods
  - Least squares
  - Fisher's Linear Discriminant (omitted)
  - Perceptrons

Least Square for Classification

- Simple closed-form solution exists
- Each C_k, $k = 1,2, \dots, K$, is described by its own linear model
$$y_k(x) = w_k^T x + w_{k0}$$
- Create augmented vector $x = (1, x^T)^T$ and $w_k = \left(w_{k0}, w_k^T\right)^T$(length is D+1)
- Grouping into vector notation $y(x) = W^T x$
  - W is the parameter matrix whose kth column is a D+1 dimensional vector (including bias)
  - $W = [w_1 \; w_2 \; \dots w_K]$
  - $y(x)$ is K by 1
- New input vector x is assigned to class for which output $y_k = w_k^T x$ is largest
- Determine W by minimizing squared error

Learning Parameters Using Least Squares

- Training data $\{x_n, t_n\}$, $n = 1,2,\dots,N$.
  t_n is a column vector of K dimensions using 1-of-K form
- Define matrices

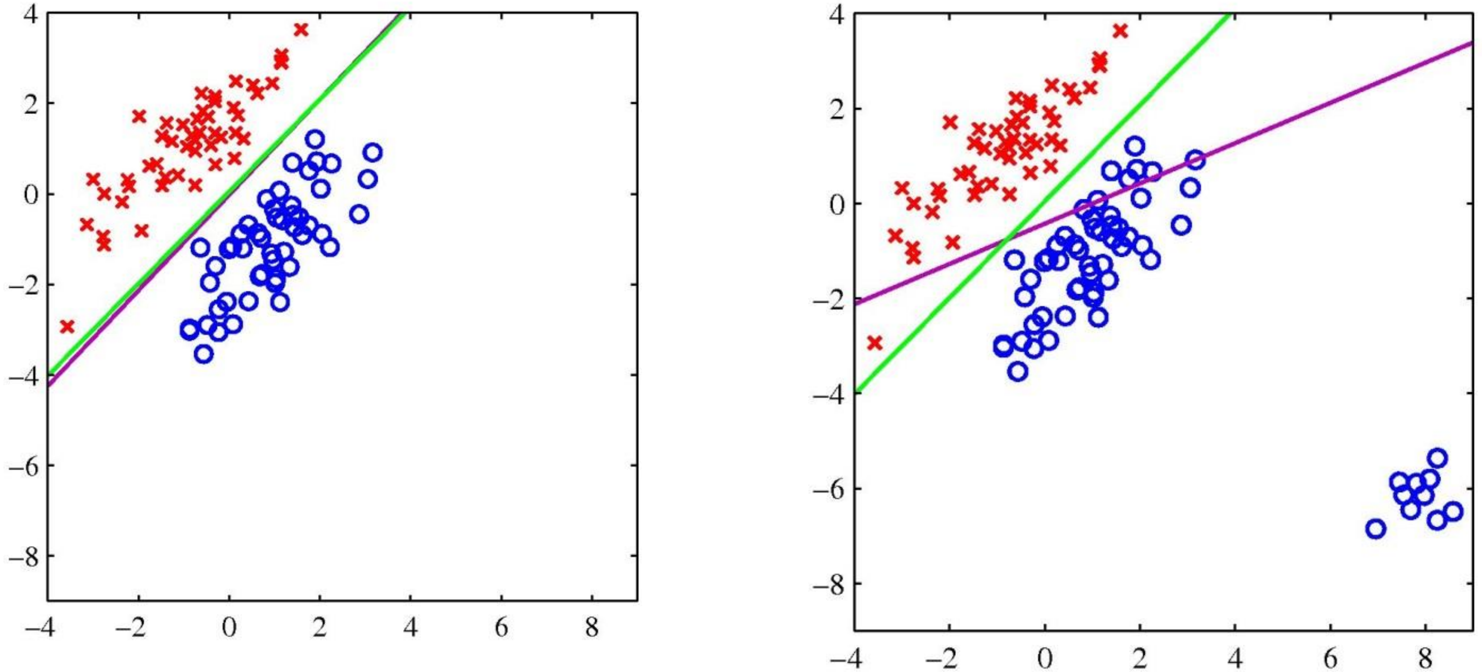
$$T = \begin{bmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_N^T \end{bmatrix}, X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}, \text{ Error vector} = \overset{N\times(D+1)}{X}\ \overset{(D+1)\times K}{W} - \overset{N\times K}{T}$$

- Let $B = XW - T = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_N^T \end{bmatrix}$, we are trying to minimize the sum of squared errors $\sum_{j=1}^{K} \sum_{i=1}^{N} e_{ij}^2$.

Learning Regression Parameters

- We have k vectors to learn: $W = [w_1 \; w_2 \; \dots w_K]$
- Each w_i controls the $i - th$ dimension of the output.
- All of the K vectors can be learned separately.
- That is, we can learn K separate regression models, each use one of the K columns in T as the outcome vector in the linear regression models we introduced before.
- To predict the output class for an input x, compute $y_i = w_i^T x$ for $i = 1, 2, \dots, K$, and
- classify x to class k for which y_k is the maximum.

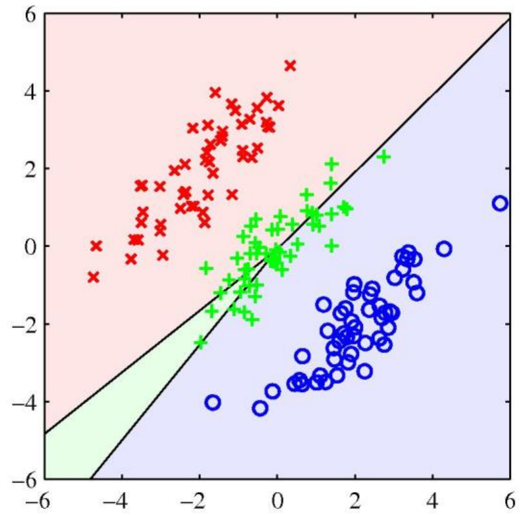
[Limitation] Least Square is Sensitive to Outliers

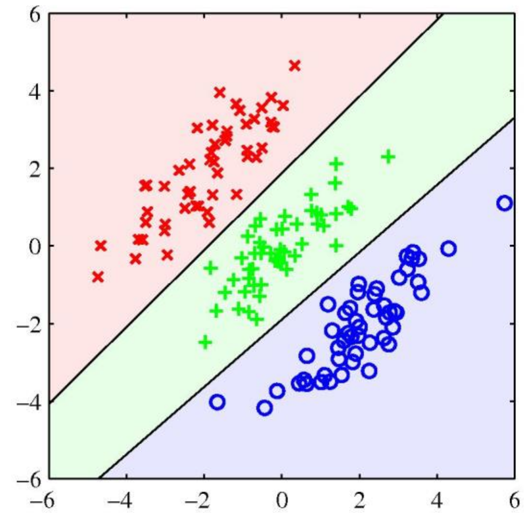


- Sum of squared errors penalized predictions that are “too correct” (magenta lines)
  - Long way from decision boundary
  - Logistic regression (green lines) does not have this problem
  - SVMs have an hinge error function that have a similar shape to that of the logistic regression

Disadvantages of Least Squares

Ordinary regression





Logistic regression

- Regions assigned to green class is too small, mostly misclassified
- Logistic regression performs well
- Disadvantage of least square:
  - Lack robustness to outliers
  - Gaussian assumption may not be a good choice for classification problems
  - Binary target values have a distribution far from Gaussian

Perceptron Algorithm

- Two-class model
  - Input vector x transformed by a fixed nonlinear function to give feature vector $\phi(x)$
  - $y(x) = f(w^T\phi(x))$
    where non-linear activation $f(\cdot)$ is a step function
  - $f(a) = \begin{cases} +1 \; if \; a \geq 0 \\ -1 \; if \; a < 0 \end{cases}$
- Use a target coding scheme
  - t=1 for class C_1, and t=-1 for C_2 similar to the activation function

Perceptron Error Function

- Error function: number of misclassifications
- This error function is a piecewise constant function of w with discontinuities (unlike regression)
- Hence no closed form solution

Perceptron Criterion

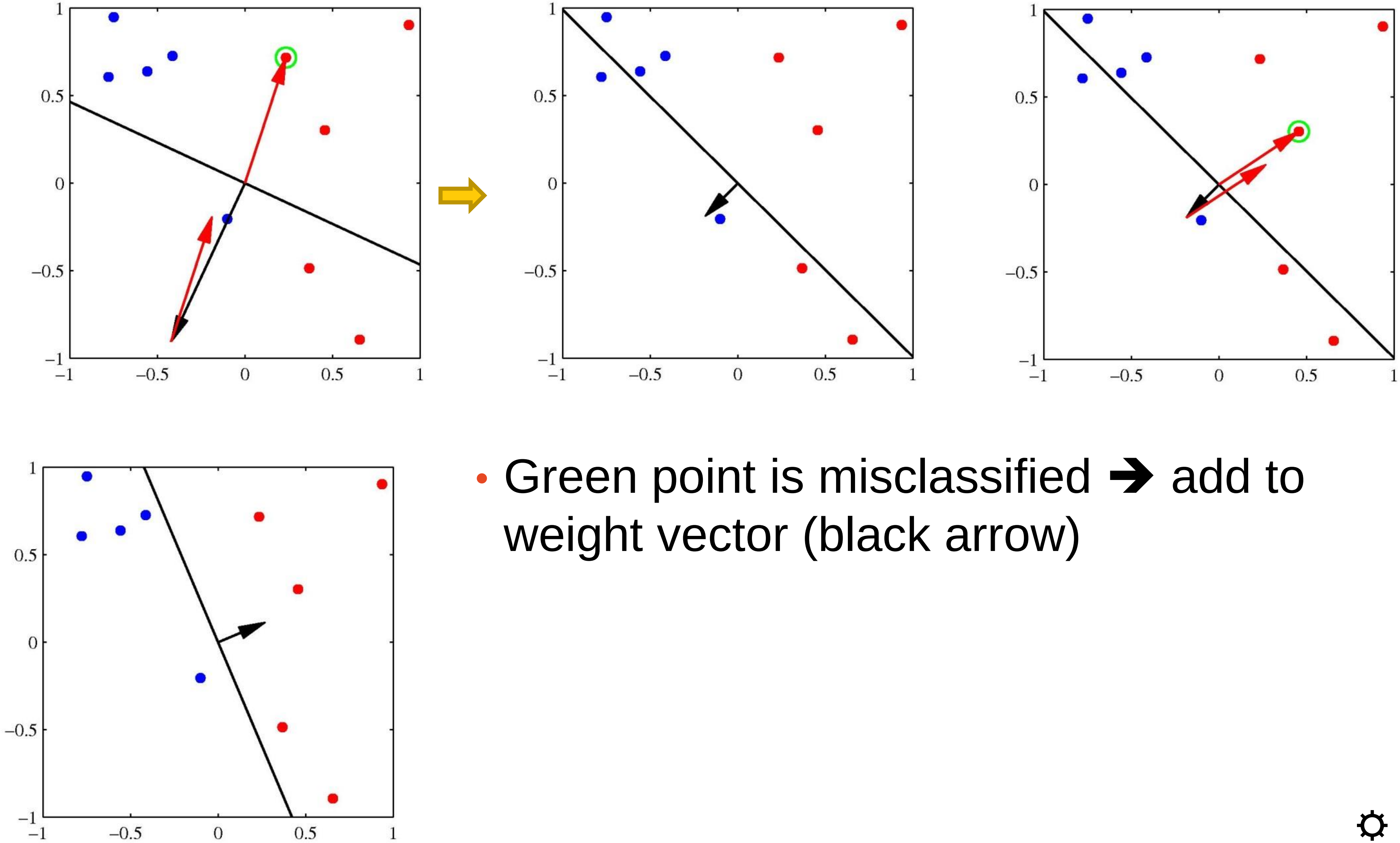
- Seek w such that $x_n \in C_1$ will have $w^T x_n \geq 0$ where as $x_n \in C_2$ will have $w^T x_n < 0$
- Target value $t \in \{-1, 1\}$. C_1 ➔ t=1, all patterns need to satisfy $w^T \phi(x_n) t_n > 0$
- For each misclassified sample, perceptron criterion tried to minimize $E_p(w) = -\sum_{n \in \mathrm{M}} w^T \phi_n t_n$
- M denotes set of all misclassified patterns and $\phi_n = \phi(x_n)$

Perceptron Algorithm

- Error function $E_p(w) = -\sum_{n\in \mathrm{M}} w^T \phi_n t_n \equiv \sum_{n\in M} E_n$
- Stochastic gradient descent
  - a.k.a Sequential gradient descent
  - Loop through M, for each data point n, change in weight is given by
  - $w^{\tau+1} = w^{\tau} - \eta \nabla \mathrm{E_n(w)} = w^{\tau} + \eta \phi_n t_n$
  - $\nabla E_n(w) = \frac{\partial E_n(w)}{\partial w} = -\phi_n t_n$

    η is learning rate (set to 1), τ is a step index
- The algorithm
  - Cycle through the training patterns in turn
  - If incorrectly classified for class C_1, add to weight vector
  - If incorrectly classified for class C_2, subtract from weight vector

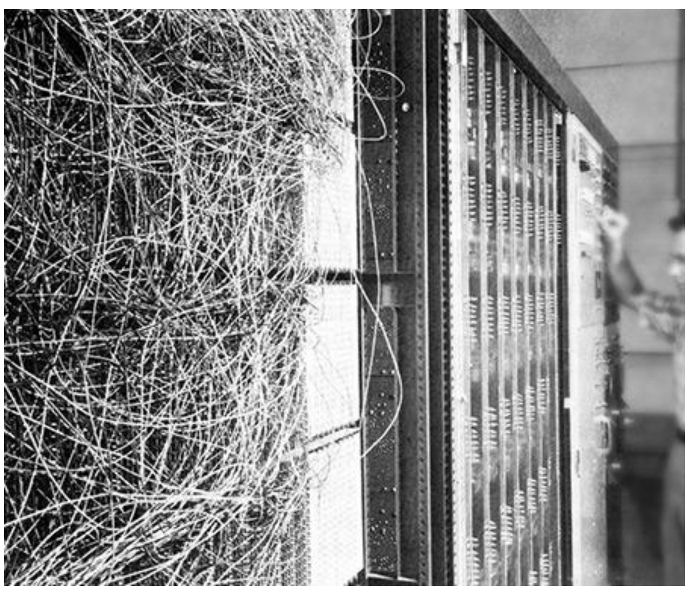
Perceptron Learning



- Green point is misclassified ➔ add to weight vector (black arrow)

History of Perceptron

- Perceptron invented by Frank Rosenblatt



- Mark 1 perceptron hardware (patching board, allowing different configurations)

Disadvantages of Perceptrons

- Does not converge if classes not linearly separable
- Does not provide probabilistic output
- Not readily generalized to K>2 classes

Comparison of Parameter Learning Approaches

- Least Squares
  - Not robust to outliers, model close to target values
- Fisher’s linear discriminant (omitted)
  - Separation determined by within-class and between class variance
  - Can be seen as a special case of least square
- Perceptrons
  - Does not converge if classes not linearly separable
  - Does not provide probabilistic output
  - Not readily generalized to K>2 classes

PROBABILISTIC GENERATIVE MODELS

Overview of Methods for Classification

- Generative Models (two-step)
  - Infer class-conditional densities $p(x|C_k)$ and priors $p(C_k)$
  - Use Bayes theorem to determine posterior probabilities
  - $p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$
- Discriminative Models (One-step)
  - Directly infer posterior probabilities $p(C_k|x)$
- Decision Theory
  - In both cases, use decision theory to assign each new x to a class

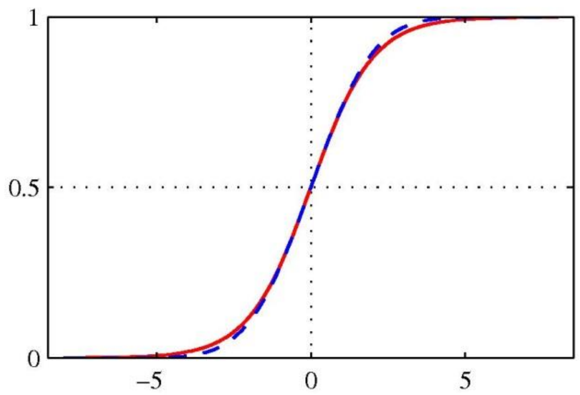
Generative Model

- Model class conditionals $p(x|C_k)$ and priors $p(C_k)$
- Compute posterior $p(C_k|x)$ from Bayes theorem
- Two class case
- Posterior for class C_1 is

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} = \frac{1}{1 + \exp(-a)}$$
$$= \sigma(a)$$

where $a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$ (Log-likelihood Ratio)

Logistic Sigmoid Function



- $\sigma(a) = \frac{1}{1+\exp(-a)}$
- Property: $\sigma(-a) = 1 - \sigma(a)$
- Inverse: $a = \ln(\frac{\sigma}{1-\sigma})$
- If $\sigma(a) = p(C_1|x)$, then inverse represent $\ln[p(C_1|x)/p(C_2|x)]$ → Log ratio of probabilities
- Sigmoid: “S”-shape or squashing function
  - Maps the real line to (0, 1)

Softmax: Generalization of logistic sigmoid

- For *K=2* we have logistic sigmoid
- For *K > 2*, we have

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \boxed{\frac{\exp(a_k)}{\sum_j \exp(a_j)}}$$

If *K=2* this reduces to earlier form

$$\begin{aligned} p(C_1|x) &= \exp(a_1)/\left[\exp(a_1) + \exp(a_2)\right] \\ &= 1/[1 + \exp(a_2 - a_1)] \\ &= 1/[1 + p(x|C_2)p(C_2)/p(x|C_1)p(C_1 \\ &= 1/[1 + \exp(-a)] \end{aligned}$$

where $a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$

  - Quantities a_k are defined by

$$a_k = \ln p(x|C_k)p(C_k)$$

- Known as the *soft-max* function
  - Since it is a smoothed max function
    - If $a_k \gg a_j$ for all $k \neq j$ then $p(C_k|x) = 1$ and 0 for the rest
    - A general technique for finding max of several a_k

Generative Model with Gaussian Input

- Gaussian class-conditional densities with same covariance matrix

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1} (x-\mu_k)\right\}$$

- Two-class case

$$p(C_1|x) = \sigma\left(\ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}\right) = \sigma(w^T x + w_0)$$

  - where

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{p(C_1)}{p(C_2)}$$

[Exercise]

  - Quadratic terms in x cancel due to common covariance
  - A linear function of x in argument of logistic sigmoid

Two Gaussian Classes

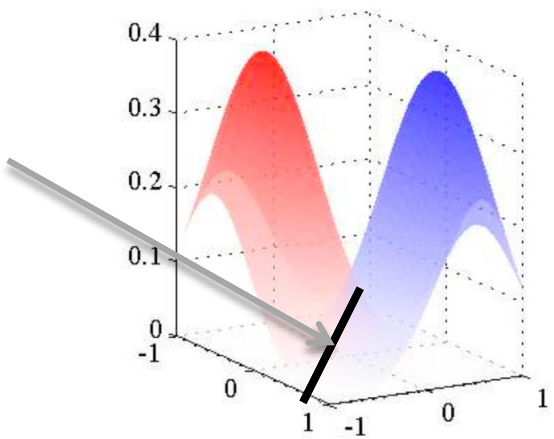
Two-dimensional input space

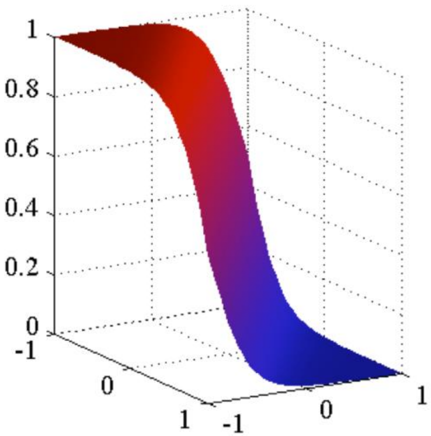
x=(x_1,x_2)

Class-conditional densities $p(\mathrm{x}|C_k)$

Posterior probability $p(C_1|\mathrm{x})$

Linear
Decision
boundary





A logistic sigmoid of a linear function of x

Two classes in different colors

Values are positive (need not sum to 1)

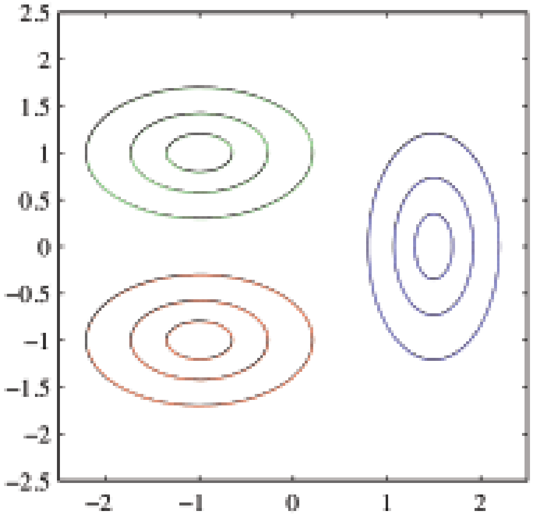
Red ink proportional to $p(C_1/\mathrm{x})$
Blue ink to $p(C_2/\mathrm{x})=1-p(C_1/\mathrm{x})$

Continuous case with *K* >2

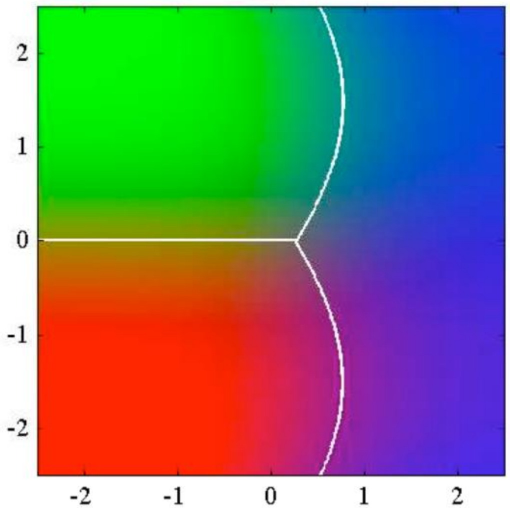
- $p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$
- with Gaussian class conditionals
  - where $a_k(\mathrm{x}) = w_k^T \mathrm{x} + w_{k0}$
  - $w_k = \Sigma^{-1}\mu_k$
  - $w_{k0} = -\frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \ln p(C_k)$
  - If we did not assume shared covariance matrix we get a quadratic discriminant

Three-class case with Gaussian models

Both Linear and Quadratic Decision boundaries



Class-conditional Densities
C1 and *C2* have same covariance matrix



Posterior Probabilities
Between *C1* and *C2* boundary is linear,
Others are quadratic
RGB values correspond to posterior probabilities

M.L.E. for Gaussian Parameters

- Assuming parametric forms for $p(x|C_k)$ we can determine values of parameters and priors $p(C_k)$ using maximum likelihood
- Dataset given $\{x_n, t_n\}, n = 1, 2, \ldots, N$
  - $t_n = 1$ for class C_1
  - $t_n = 0$ for class C_2
  - Prior probability $p(C_1) = \pi$, $p(C_2) = 1 - \pi$
  - $p(x_n, C_1) = p(C_1)p(x_n|C_1) = \pi N(x_n|u_1, \Sigma)$
  - $p(x_n, C_2) = p(C_2)p(x_n|C_2) = (1 - \pi)N(x_n|u_2, \Sigma)$
- Likelihood is:
  - $p(\boldsymbol{t}|\pi, u_1, u_2, \Sigma) = \prod_{n=1}^{N}[\pi N(x_n|u_1, \Sigma)]^{t_n}[(1 - \pi)N(x_n|u_2, \Sigma)\ ]^{1-t_n}$
  - $\boldsymbol{t} = (t_1, t_2, \ldots, t_N)^T$
- Convenient to maximize log of likelihood

Max Likelihood for Prior

- Estimates for prior probabilities
  - Log likelihood function that depend on π are

$$\sum_{n=1}^{N} t_n \ln \pi + (1 - t_n) \ln(1 - \pi)$$

  - Set derivative to zero and rearrange

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N_1 + N_2}$$

MLE for π is fraction of points

N_1 is data point in class C_1; N_2 is data point in class C_2 (Exercise)

- What will happen is one of the two classes is missing in training data?

$$p(\mathbf{t}|\pi, u_1, u_2, \Sigma)$$
$$= \sum_{n=1}^{N} [\pi N(x_n|u_1, \Sigma)]^{t_n} [(1 - \pi) N(x_n|u_2, \Sigma)\ ]^{1-t_n}$$

Max Likelihood for Means

- Estimates for class means
  - Pick log likelihood function depending only on μ_1
  - $\sum_{n-1}^{N} t_n \ln N(x_n|\mu_1, \Sigma)$
$$= -\frac{1}{2}\sum_{n-1}^{N} t_n (x-\mu_1)^T \Sigma^{-1} (x-\mu_1) + const$$
  - Setting derivative to zero and solving $\mu_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n x_n$
  - Similarly $\mu_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-t_n)x_n$

Mean of all input vectors x_n assigned to class C_1

Max Likelihood for Covariance Matrix

Solution for Shared Covariance Matrix

Pick out terms in log-likelihood function depending on Σ

$$-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\mathbf{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$-\frac{1}{2}\sum_{n=1}^{N}(1-t_n)\ln|\mathbf{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}(1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_2)$$

$$= -\frac{N}{2}\ln|\mathbf{\Sigma}| - \frac{N}{2}\mathrm{Tr}\left\{\mathbf{\Sigma}^{-1}\mathbf{S}\right\}$$

$$\mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}}$$

$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}.$$

Tricks:

$$\frac{\partial \ln|A|}{\partial A} = (A^{-1})^T$$

$$\frac{\partial \mathrm{Tr}(\mathbf{A}\mathbf{X}^{-1}\mathbf{B})}{\partial \mathbf{X}} = -(\mathbf{X}^{-1}\mathbf{B}\mathbf{A}\mathbf{X}^{-1})^T$$

Weighted average of the two separate covariance matrices

Setting first derivative to zero and solve $\Sigma = S$

[Exercise]

Discrete Features

- Assuming binary features $x_i \in \{0,1\}$
  - With D inputs, distribution is a table of 2^D values
- Naive Bayes assumption: independent features
  - Class-conditional distributions have the form
    $p(\mathrm{x}|C_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i}(1-\mu_{ki})^{1-x_i}$
  - Substituting in the form needed for normalized exponential
    $a_k(\mathrm{x}) = \ln(p(x|C_k)p(C_k)) = \sum_{i=1}^{D}\{x_i \ln \mu_{ki} + (1-x_i)\ln(1-\mu_{ki})\} + \ln p(C_k)$
  - which is linear in x
- Similar results for discrete variables which take M>2 values

Summary of probabilistic linear classifiers

- Defined using
  - logistic sigmoid $P(C_1|x) = \sigma(a)$ where a is LLR with Bayes odds
  - soft-max functions $P(C_k|x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$
- Continuous case with shared covariance
  - we get linear functions of input x
- Discrete case with independent features also results in linear functions

QUESTIONS?