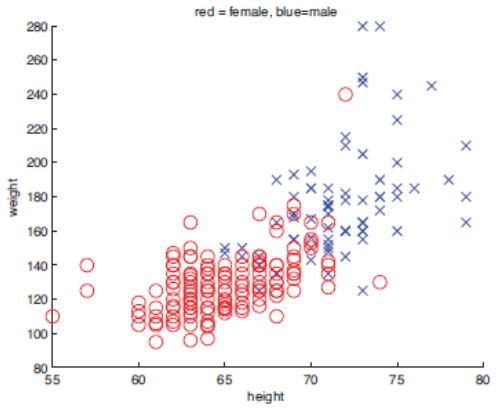
Probabilistic Generative Models

Probabilistic Generative Models

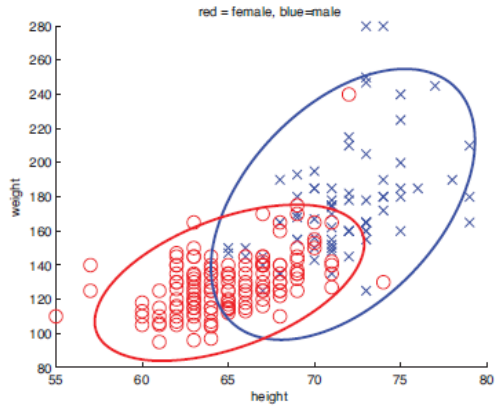
model the **class-conditional densities $p(\mathbf{x}/C_k)$**, as well as the **class priors $p(C_k)$**

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

class-conditional densities $p(\mathbf{x}/C)$ $p(\mathbf{x}|y = c, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$



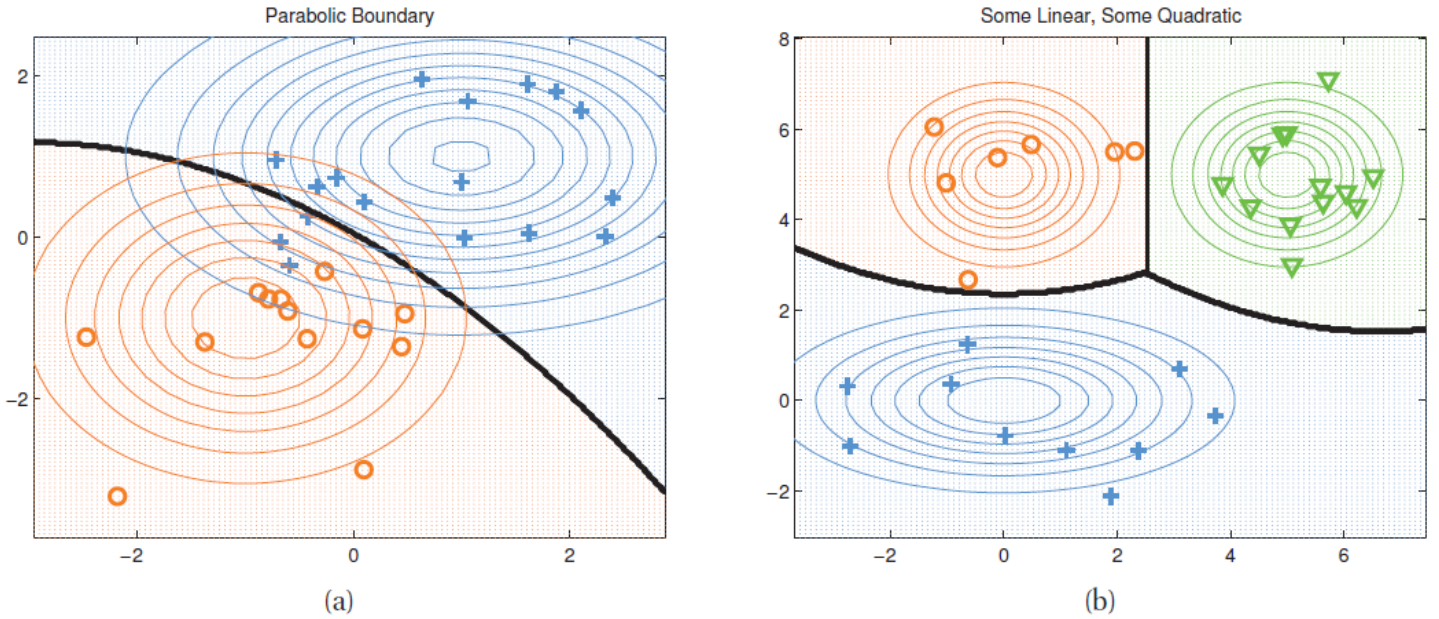
(a)



(b)

Quadratic discriminant analysis

$$p(y=c|\mathbf{x},\boldsymbol{\theta}) \quad = \quad \frac{\pi_c|2\pi\boldsymbol{\Sigma}_c|^{-\frac{1}{2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^T\boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)\right]}{\sum_{c'}\pi_{c'}|2\pi\boldsymbol{\Sigma}_{c'}|^{-\frac{1}{2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{c'})^T\boldsymbol{\Sigma}_{c'}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{c'})\right]}$$



Linear Discriminant Analysis

- covariance matrices are **tied** or **shared** across classes, **Σ***c* = **Σ**.

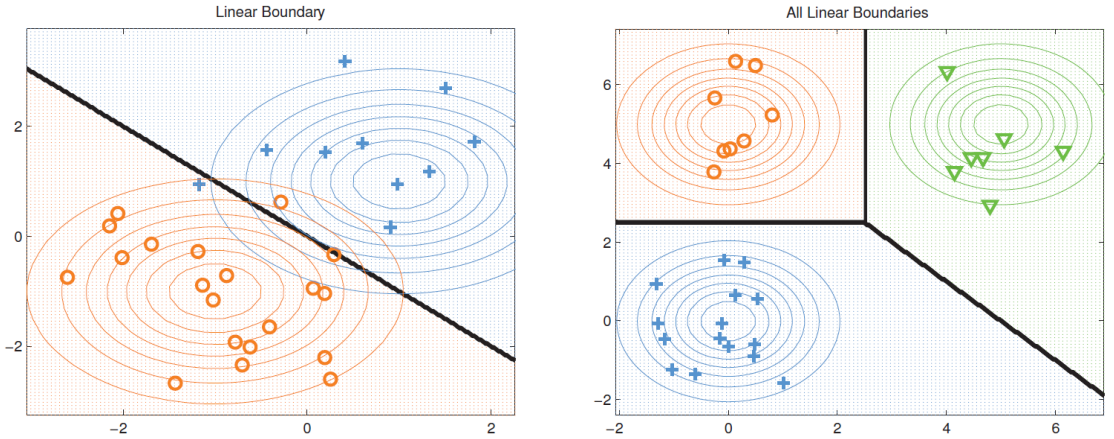
$$p(y=c|\mathbf{x},\boldsymbol{\theta}) \propto \pi_c \exp\left[\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c\right]$$

$$= \exp\left[\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c + \log\pi_c\right]\exp[-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}]$$

$$p(y=c|\mathbf{x},\boldsymbol{\theta}) = \frac{e^{\boldsymbol{\beta}_c^T\mathbf{x}+\gamma_c}}{\sum_{c'} e^{\boldsymbol{\beta}_{c'}^T\mathbf{x}+\gamma_{c'}}}$$

$$\gamma_c = -\frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c + \log\pi_c$$

$$\boldsymbol{\beta}_c = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c$$



Discriminant Analysis : Parameetr estimation

- Models joint probability of observing input and output

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y_i = c)\log \pi_c\right] + \sum_{c=1}^{C}\left[\sum_{i:y_i=c}\log \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\right]$$

$$\hat{\pi}_c = \frac{N_c}{N}, \qquad \hat{\boldsymbol{\mu}}_c = \frac{1}{N_c}\sum_{i:y_i=c}\mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c}\sum_{i:y_i=c}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^T$$

- MLE can badly overfit in high dimensions.
- Use a diagonal covariance matrix for each class, which assumes the features are conditionally independent; this is equivalent to using a **naive Bayes classifier**
- Other approaches : Use **MAP** and **Bayesian approaches**

Naïve Bayes Classifier

- Features are conditionally independent given the class label.

$$p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{j=1}^{D} p(x_j|y=c,\boldsymbol{\theta}_{jc})$$

- "naive" since we do not expect the features to be independent, but results in classifiers that work well
  - model is quite simple and hence it is relatively immune to overfitting, as a lower number of parameters need to be estimated due to independence assumption.
- class-conditional density p(x|y)

Naïve Bayes Classifier

- Features are conditionally independent given the class label.

$$p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{j=1}^{D} p(x_j|y=c,\boldsymbol{\theta}_{jc})$$

- class-conditional density p(x|y)
  - In the case of real-valued features, we can use the Gaussian distribution: $p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{j=1}^{D} \mathcal{N}(x_j|\mu_{jc},\sigma_{jc}^2)$, where μ_{jc} is the mean of feature j in objects of class c, and σ_{jc}^2 is its variance.
  - In the case of binary features, $x_j \in \{0,1\}$, we can use the Bernoulli distribution: $p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{j=1}^{D} \mathrm{Ber}(x_j|\mu_{jc})$, where μ_{jc} is the probability that feature j occurs in class c.
  - In the case of categorical features, $x_j \in \{1,\ldots,K\}$, we can model use the multinoulli distribution: $p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{j=1}^{D} \mathrm{Cat}(x_j|\boldsymbol{\mu}_{jc})$, where $\boldsymbol{\mu}_{jc}$ is a histogram over the K possible values for x_j in class c.

Training NBC : ML Estimation

$$p(\mathbf{x}_i, y_i|\boldsymbol{\theta}) = p(y_i|\boldsymbol{\pi}) \prod_j p(x_{ij}|\boldsymbol{\theta}_j) = \prod_c \pi_c^{\mathbb{I}(y_i=c)} \prod_j \prod_c p(x_{ij}|\boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i=c)}$$

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\theta}_{jc})$$

Bernoulli class condiitonal likelihood

$$\hat{\pi}_c = \frac{N_c}{N} \qquad \hat{\theta}_{jc} = \frac{N_{jc}}{N_c}$$

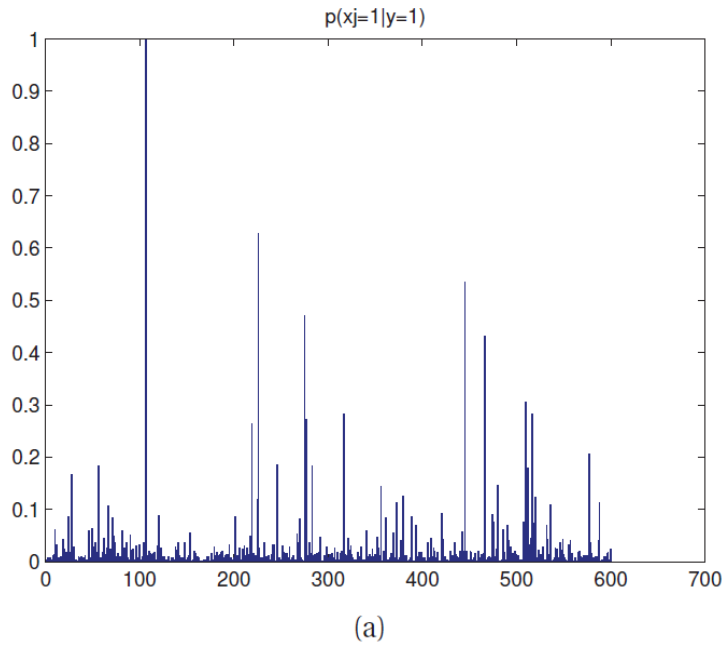
The method is easily generalized to handle features of mixed type.

NBC : Algorithm

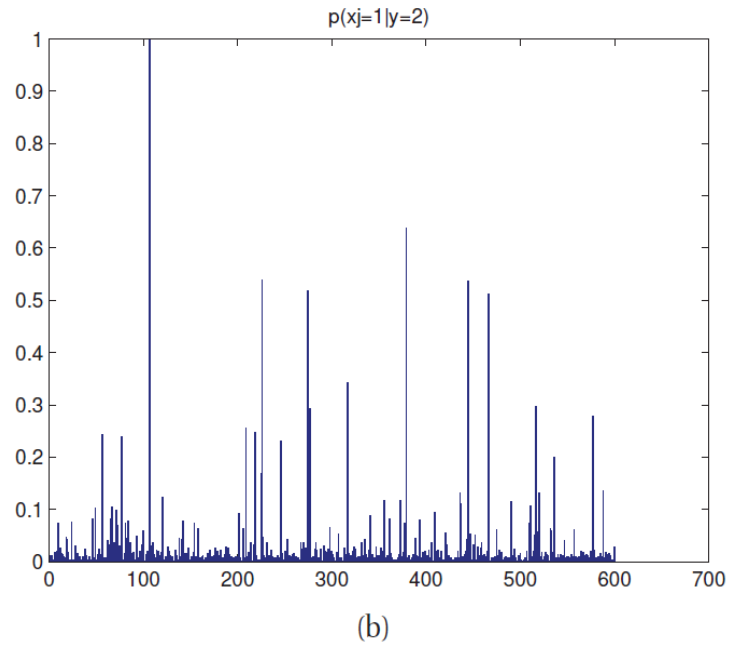
Algorithm 3.1: Fitting a naive Bayes classifier to binary features

```
1 N_c = 0, N_jc = 0;
2 for i = 1 : N do
3     c = y_i // Class label of i'th example;
4     N_c := N_c + 1 ;
5     for j = 1 : D do
6         if x_ij = 1 then
7             N_jc := N_jc + 1
8 π̂_c = N_c / N, θ̂_jc = N_jc / N
```

$O(ND)$ time.



(a)



(b)

Spam Classification example

HAM examples	**SPAM examples**
• d1: {good} • d2: {very good} P(y=ham)= 2/5	• d3: {bad} • d4: {very bad} • d5: {very bad very bad} p(y=spam)= 3/5

- Test data: d6: {good bad very bad} - **SPAM OR HAM??**
- Vocabulary : V={good, bad, very}
- Use Naive Bayes classifier:
- P(ham | d6)= $\prod$ P(word | y=ham) P(y=ham)
  = P(good | ham) * P(bad | ham) * P(very | ham)

Word frequencies wrt class	P(good | y)	P(bad | y)	P(very | y)
Class 0: y = spam	0/3	3/3	2/3
Class 1: y = ham	2/2	0/2	1/2

- Estimate parameters using ML:
- P(ham | d6)= P(good | ham) * P(bad | ham) * P(very | ham)* P(ham)
- P(spam | d6)= P(good | spam) * P(bad | spam) * P(very | spam)* P(spam)

- What is the problem with this ?

Discriminative vs Generative

- **Generative Models :** model the joint distribution p(x, Ck) directly and then normalize to obtain the posterior probabilities.

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

- Most demanding because it involves finding the joint distribution over both x and Ck. For many applications, x will have high dimensionality, and consequently we may need a large training set. if we only wish to make classification decisions, then it can be wasteful of computational resources.
- **Easy to fit?** very easy to fit generative classifiers. we can fit a naive Bayes model and an LDA model by simple counting and averaging. logistic regression requires solving a convex optimization problem which is much slower.
- **Fit classes separately?** In a generative classifier, we estimate the parameters of each class conditional density independently, so we do not have to retrain the model when we add more classes.
- **Well-calibrated probabilities?** Some generative models, such as naive Bayes, make strong independence assumptions which are often not valid
- Generative models can easily handle unlabelled data and missing features

Classification : Evaluation metrics

Accuracy : $$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[y_i == \hat{y}_i]$$

		Actual Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct
Sensitivity (Recall)	TP / (TP + FN)	The percentage of positive cases that were predicted as positive
Specificity	TN / (TN + FP)	The percentage of negative cases that were predicted as negative