Lecture 18: Generative Models

CS 475: Machine Learning

Raman Arora

April 5, 2017

Slides credit: Greg Shakhnarovich

Review: Kernel support vector machines

- Mercer kernels: $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j)$
- The optimization problem (learning SVM):

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right\}$$

  - Need to compute the *kernel matrix* for the training data
- Prediction:

$$\hat{y} = \operatorname{sign}\left(\hat{w}_0 + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

  - Need to compute $K(\mathbf{x}_i, \mathbf{x})$ for all SVs $\mathbf{x}_i$.

Review: Representer theorem

- Consider the optimization problem

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 \; \forall i$$

- Theorem: the solution can be represented as

$$\mathbf{w}^* = \sum_{i=1}^{N} \beta_i \mathbf{x}_i$$

- This is the "magic" behind Support Vector Machines!

Review: Representer theorem - proof I

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 \; \forall i \quad \Rightarrow \quad \mathbf{w}^* = \sum_{i=1}^{N} \beta_i \mathbf{x}_i$$

- Let $\mathbf{w}^* = \mathbf{w}_X + \mathbf{w}_\perp$, where
  $\mathbf{w}_X = \sum_{i=1}^{N} \beta_i \mathbf{x}_i \in Span(\mathbf{x}_1, \ldots, \mathbf{x}_N)$,
  $\mathbf{w}_\perp \notin Span(\mathbf{x}_1, \ldots, \mathbf{x}_N)$, i.e., $\mathbf{w}_\perp \cdot \mathbf{x}_i = 0$ for all $i = 1, \ldots, N$
- For all $\mathbf{x}_i$ we have

$$\mathbf{w}^* \cdot \mathbf{x}_i = \mathbf{w}_X \cdot \mathbf{x}_i + \mathbf{w}_\perp \cdot \mathbf{x}_i = \mathbf{w}_X \cdot \mathbf{x}_i$$

therefore,

$$y_i(\mathbf{w}^* \cdot \mathbf{x}_i + w_0) \geq 1 \quad \Rightarrow \quad y_i(\mathbf{w}_X \cdot \mathbf{x}_i + w_0) \geq 1$$

Review: Representer theorem - proof II

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 \;\forall i \quad \Rightarrow \quad \mathbf{w}^* = \sum_{i=1}^{N} \beta_i \mathbf{x}_i$$

- Now, we have

$$\|\mathbf{w}^*\|^2 = \mathbf{w}^*\cdot\mathbf{w}^* = (\mathbf{w}_X + \mathbf{w}_\perp)\cdot(\mathbf{w}_X + \mathbf{w}_\perp) = \underbrace{\mathbf{w}_X\cdot\mathbf{w}_X}_{\|\mathbf{w}_X\|^2} + \underbrace{\mathbf{w}_\perp\cdot\mathbf{w}_\perp}_{\|\mathbf{w}_\perp\|^2},$$

  since $\mathbf{w}_X\cdot\mathbf{w}_\perp = 0$.
- Suppose $\mathbf{w}_\perp \neq \mathbf{0}$. Then, we have a solution $\mathbf{w}_X$ that satisfies all the constraints, and for which
  $\|\mathbf{w}_X\|^2 < \|\mathbf{w}_X\|^2 + \|\mathbf{w}_\perp\|^2 = \|\mathbf{w}^*\|^2$.
- This contradicts optimality of $\mathbf{w}^*$, hence $\mathbf{w}^* = \mathbf{w}_X$. QED

Review: Kernel SVM in the primal

- Recall: $\hat{y} = \operatorname{sign}\left(\hat{w}_0 + \sum_{\alpha_i>0}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x})\right)$
- Can not write $\mathbf{w}$ explicitly; instead, optimize $\boldsymbol{\alpha}$
- How can we write the regularizer?

$$\|\mathbf{w}\|^2 = \mathbf{w}\cdot\mathbf{w} = \left[\sum_i \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i)\right]\cdot\left[\sum_j \alpha_j y_j \boldsymbol{\phi}(\mathbf{x}_j)\right]$$
$$= \sum_{i=1,j=1}^{N} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)$$

- The objective for learning is

$$\min_{\boldsymbol{\alpha}}\left\{\frac{\lambda}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) + \sum_i\left[1 - y_i\sum_j \alpha_j y_j K(\mathbf{x}_i,\mathbf{x}_j)\right]_+\right\}$$

Review: kernels

- Representer theorem:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 \; \forall i \quad \Rightarrow \quad \mathbf{w}^* = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$$

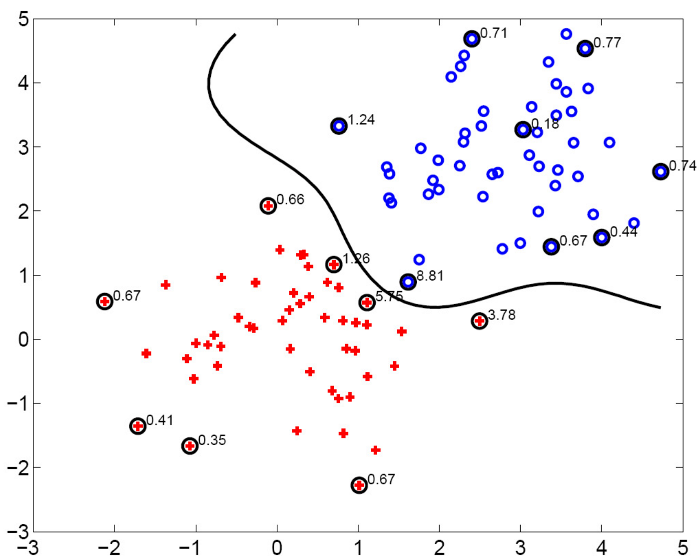
- Polynomial kernel (includes linear):

$$K(\mathbf{x}_i, \mathbf{x}_j; c, d) =, (c + \mathbf{x}_i \cdot \mathbf{x}_j)^d$$

- RBF kernel:

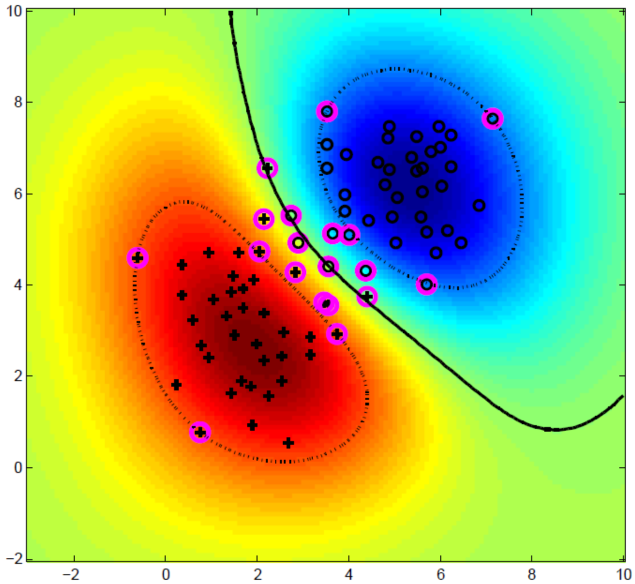
$$K(\mathbf{x}_i, \mathbf{x}_j; \sigma) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right).$$

Review: SVM with RBF (Gaussian) kernels



- Data are linearly separable in the (infinite-dimensional) feature space
- We don't need to explicitly compute dot products in that feature space – instead we simply evaluate the RBF kernel.

Review: SVM with RBF kernels: geometry



- positive margin: level set

$$\{\mathbf{x} : \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) = 1\}$$

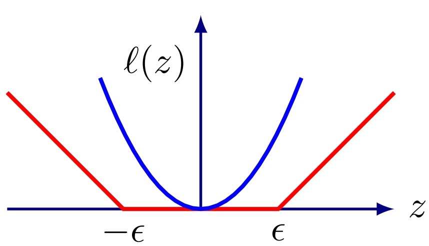
- negative margin: level set

$$\{\mathbf{x} : \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) = -1\}$$

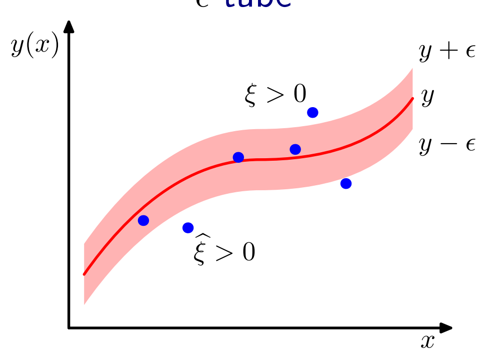
Review: SVM regression

- The key ideas:

ϵ-insensitive loss



ϵ-tube



- Two sets of slack variables:

$$\begin{aligned} y_i &\le f(\mathbf{x}_i) + \epsilon + \xi_i, \\ y_i &\ge f(\mathbf{x}_i) - \epsilon - \tilde{\xi}_i, \qquad \xi_i \ge 0,\ \tilde{\xi}_i \ge 0. \end{aligned}$$

- Optimization: $\min C \sum_i \left(\xi_i + \tilde{\xi}_i\right) + \frac{1}{2}\|\mathbf{w}\|^2$

SVM with more than two classes

- Some classifiers are "natively multiclass"
  e.g., decision trees
- With any natively binary classifier (AdaBoost; logistic regression; SVM), our options for $C > 2$ classes include:
- One-vs-all: build C classifiers
  need to reconcile; easy if have calibrated $p\left(y \,|\, \mathbf{x}\right)$
- One-vs-one: build $\binom{C}{2}$ classifiers
  need to reconcile; more problematic since can have inconsistencies
- Build some sort of "tournament", or a class tree
  often the most efficient; how to build the tree?
- Extend to multi-class by modifying the machinery
  softmax is an extension of logistic regression;
  multi-class SVM extension is next

Multiclass SVM: setup

- Many attempts to generalize SVM to multi-class; we will follow the one due to Crammer and Singer (2000).
- Basic idea: for C classes, learn $\mathbf{w}_c$ for $c = 1, \ldots, C$,

$$\hat{y}(\mathbf{x}; \underbrace{\mathbf{w}_1, \ldots, \mathbf{w}_C}_{\mathbf{W}}) \;=\; \underset{c}{\operatorname{argmax}}\, \mathbf{w}_c \cdot \mathbf{x}.$$

- Can stack $\mathbf{w}_c$s into rows of $\mathbf{W}$
- Empirical 0/1 loss on (x, y): $[\![\hat{y}(\mathbf{x}; \mathbf{W}) \;\neq\; y]\!]$
- Surrogate loss on $(\mathbf{x}, y)$:

$$\max_{r} \left\{ {\mathbf{w}_r}^T \mathbf{x} \;+\; 1 - \delta_{r,y_i} \right\} \;-\; {\mathbf{w}_y}^T \mathbf{x}$$

$\delta_{a,b} = 1$ iff $a = b$, otherwise 0.

Optimization

- Surrogate loss is a bound on 0/1 loss:

$$\frac{1}{N}\sum_i [\![\hat{y}(\mathbf{x}_i;\mathbf{W}) \neq y_i]\!] \leq \frac{1}{N}\sum_i \left[\max_r \left\{ \mathbf{w}_r^T\mathbf{x} + 1 - \delta_{y_i,r} \right\} - \mathbf{w}_y^T\mathbf{x}_i\right]$$

- Proceed as in (separable) SVM: want to find the lowest norm solution that achieves 1-margin

$$\min_{\mathbf{W}} \frac{\lambda}{2}\|\mathbf{W}\|_2^2$$
$$\text{s.t. } \forall\, i,\ c \neq y_i \quad \mathbf{w}_{y_i}^T\mathbf{x}_i - \mathbf{w}_c{}^T\mathbf{x}_i \geq 1.$$

where $\|\mathbf{W}\|_2^2$ is the Frobenius norm of $\mathbf{W}$.

Soft constraint version

- General (non-separable) case:

$$\min_{\mathbf{W}} \frac{\lambda}{2}\|\mathbf{W}\|_2^2 + \sum_i \xi_i$$
$$\text{s.t. } \forall\, i,\ c \neq y_i \quad \mathbf{w}_{y_i}^T\mathbf{x}_i - \mathbf{w}_c{}^T\mathbf{x}_i \geq 1 - \xi_i.$$

- Introducing Lagrange multipliers $\alpha_{i,r}$:

$$\min_{\mathbf{W},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}} \frac{\lambda}{2}\sum_c \|\mathbf{w}_c\|_2^2 + \sum_i \xi_i$$
$$+ \sum_i\sum_r \alpha_{i,r}\left[(\mathbf{w}_r^T - \mathbf{w}_{y_i}^T)\mathbf{x}_i - \delta_{y_i,r} + 1 - \xi_i\right]$$
$$\text{s.t.} \forall i,\ r, \quad \alpha_{i,r} \geq 0, \quad \xi_i \geq 0.$$

Generative models

Reminder: optimal classification

- Expected classification error is minimized by

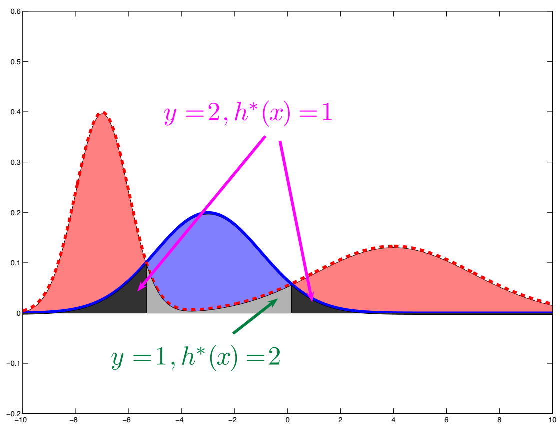
$$\begin{aligned} h(\mathbf{x}) &= \operatorname*{argmax}_c p\left(y = c \,|\, \mathbf{x}\right) \\ &= \operatorname*{argmax}_c \frac{p\left(\mathbf{x} \,|\, y = c\right) p(y = c)}{p(\mathbf{x})}. \end{aligned}$$

- The *Bayes classifier*:

$$\begin{aligned} h^*(\mathbf{x}) &= \operatorname*{argmax}_c \frac{p\left(\mathbf{x} \,|\, y = c\right) p(y = c)}{p(\mathbf{x})} \\ &= \operatorname*{argmax}_c p\left(\mathbf{x} \,|\, y = c\right) p(y = c) \\ &= \operatorname*{argmax}_c \left\{\log p\left(\mathbf{x} \,|\, y = c\right) + \log p(y = c)\right\}. \end{aligned}$$

Note: $p(\mathbf{x}) = \sum_c p(\mathbf{x}, y = c)$ is equal for all c, and can be ignored.

Bayes risk



- The risk (probability of error) of Bayes classifier h^* is called the *Bayes risk* R^*.
- This is the *minimal achievable* risk for the given $p(\mathbf{x}, y)$ with any classifier!
- In a sense, R^* measures the inherent difficulty of the classification problem.

$$R^* = 1 - \int_{\mathbf{x}} \max_c \{p(\mathbf{x} \,|\, c = y)\; p(y = c)\}\, d\mathbf{x}$$

Discriminant functions

- We can construct, for each class c, a *discriminant function*

$$\delta_c(\mathbf{x}) \triangleq \log p(\mathbf{x} \,|\, y = c) + \log p(y = c)$$

such that

$$h^*(\mathbf{x}) = \operatorname*{argmax}_c \delta_c(\mathbf{x}).$$

- Can simplify δ_c by removing terms and factors common for all δ_c since they won't affect the decision boundary.
For example, if $p(y = c) = 1/C$ for all c, can drop the prior term:

$$\delta_c(\mathbf{x}) = \log p(\mathbf{x} \,|\, y = c)$$

Two-category case

- In case of two classes $y \in \{\pm 1\}$, the Bayes classifier is

$$h^*(\mathbf{x}) = \operatorname*{argmax}_{c=\pm 1} \delta_c(\mathbf{x}) = \operatorname{sign}\left(\delta_{+1}(\mathbf{x}) - \delta_{-1}(\mathbf{x})\right).$$

- Decision boundary is given by $\delta_{+1}(\mathbf{x}) - \delta_{-1}(\mathbf{x}) = 0$.
  - Sometimes $f(\mathbf{x}) = \delta_{+1}(\mathbf{x}) - \delta_{-1}(\mathbf{x})$ is referred to as a discriminant function.
- With equal priors, this is equivalent to the *(log)-likelihood ratio test*:

$$h^*(\mathbf{x}) = \operatorname{sign}\left[\log \frac{p\left(\mathbf{x} \,|\, y = +1\right)}{p\left(\mathbf{x} \,|\, y = -1\right)}\right].$$

Equal covariance Gaussian case

- Consider the case of $p_c(\mathbf{x}) = \mathcal{N}\left(\mathbf{x};\, \mu_c, \mathbf{\Sigma}\right)$, and equal prior for all classes.

$$\begin{aligned}
\delta_k(x) &= \log p(\mathbf{x} \,|\, y = k) \\
&= \underbrace{-\log(2\pi)^{d/2} - \frac{1}{2}\log(|\mathbf{\Sigma}|)}_{\text{same for all } k} - \frac{1}{2}(\mathbf{x} - \mu_k)^T \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu_k) \\
&\propto \text{const} - \underbrace{\mathbf{x}^T \mathbf{\Sigma}^{-1} \mathbf{x}}_{\text{same for all } k} + \mu_k^T \mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T \mathbf{\Sigma}^{-1}\mu_k - \mu_k^T \mathbf{\Sigma}^{-1}\mu_k
\end{aligned}$$

- Now consider two classes r and q:

$$\delta_r(\mathbf{x}) \propto 2\mu_r^T \mathbf{\Sigma}^{-1}\mathbf{x} - \mu_r^T \mathbf{\Sigma}^{-1}\mu_r$$

$$\delta_q(\mathbf{x}) \propto 2\mu_q^T \mathbf{\Sigma}^{-1}\mathbf{x} - \mu_q^T \mathbf{\Sigma}^{-1}\mu_q$$

Linear discriminant

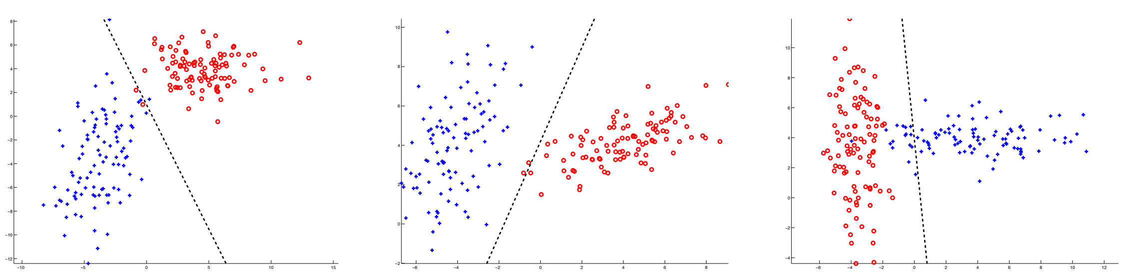
- Two class discriminants:

$$\begin{aligned}\delta_r(\mathbf{x}) - \delta_r(\mathbf{x}) &= 2\mu_r^T \mathbf{\Sigma}^{-1}\mathbf{x} - \mu_r^T \mathbf{\Sigma}^{-1}\mu_r \\ &- 2\mu_q^T \mathbf{\Sigma}^{-1}\mathbf{x} + \mu_q^T \mathbf{\Sigma}^{-1}\mu_q \\ &= \mathbf{w}^T\mathbf{x} + w_0\end{aligned}$$

- If we know what $\mu_{1,\ldots,C}$ and $\mathbf{\Sigma}$ are, we can compute the optimal $\mathbf{w}$, w_0 directly.
- What should we do when we don't know the Gaussians?

Generative models for classification

- In generative models one explicitly models $p(\mathbf{x}, y)$ or, equivalently, $p\left(\mathbf{x} \,|\, y = c\right)$ and $p(y = c)$, to derive discriminants.
- Typically, the model imposes certain parametric form on the assumed distributions, and requires estimation of the parameters from data.
  - Most popular: Gaussian for continuous, multinomial for discrete.
  - We will see later in this class *non-parametric* models.
- Often, the classifier is OK even if data clearly don't conform to assumptions.



Gaussians with unequal covariances

- What if we remove the restriction that $\forall c,\ \mathbf{\Sigma}_c = \mathbf{\Sigma}$?
- Compute ML estimate for $\mu_c, \mathbf{\Sigma}_c$ for each c.
- We get discriminants (and decision boundaries) *quadratic* in $\mathbf{x}$:

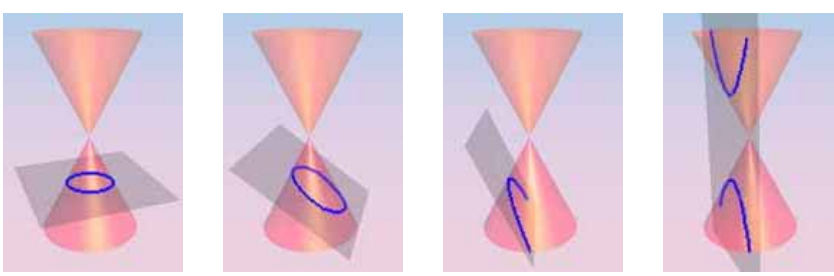
$$\delta_c(\mathbf{x}) \;=\; -\frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}_c^{-1}\mathbf{x} + \mu_c^T\mathbf{\Sigma}_c^{-1}\mathbf{x} - \text{const}_c(\mathbf{x})$$

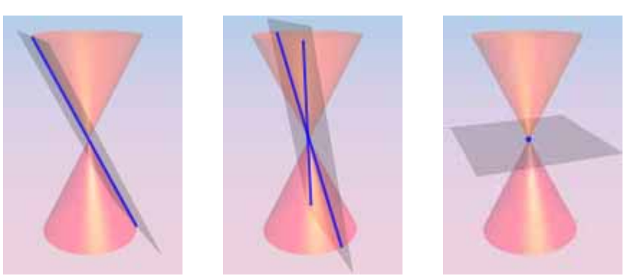
- Decision boundary with two classes:

$$\delta_1 - \delta_0 \;=\; 0$$

Quadratic decision boundaries

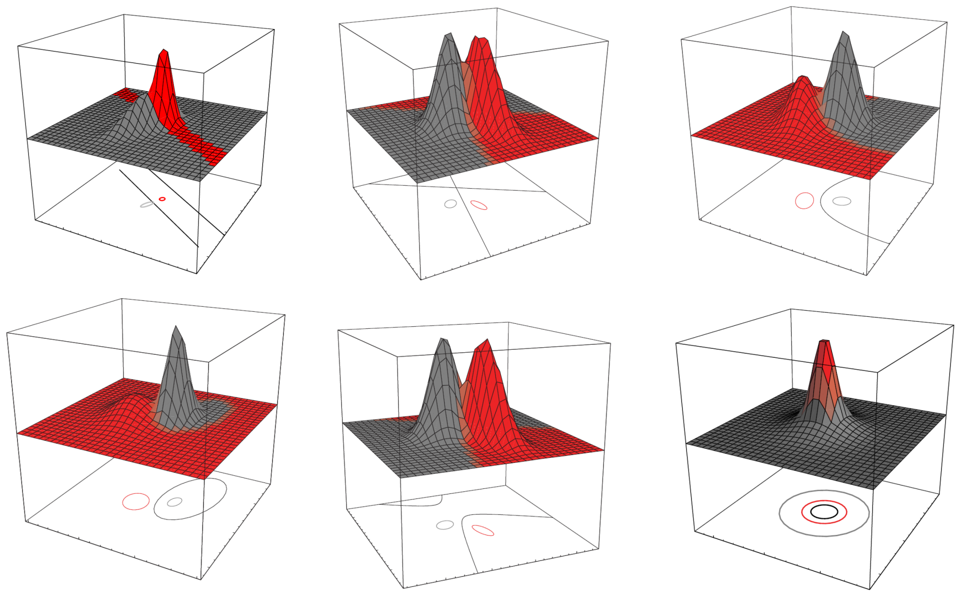
- What do quadratic boundaries look like in 2D?
- Second-degree curves can be any *conic section*:





- Can all of these arise from two Gaussian classes?

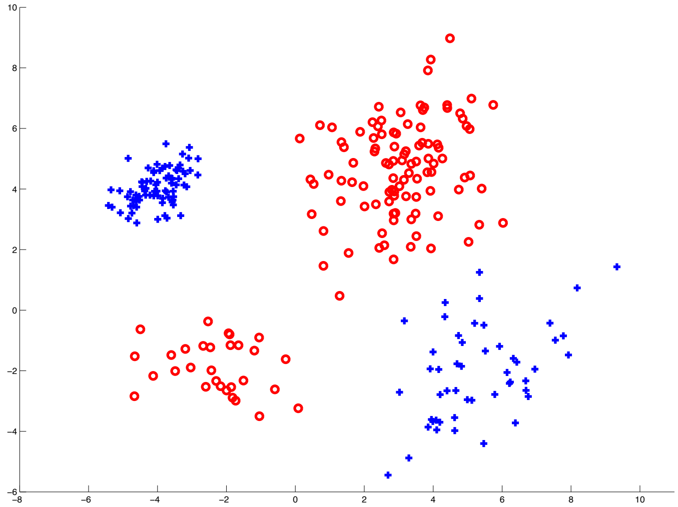
Quadratic decision boundaries



[Duda, Hart & Stork]

Mixture models

- So far, we have assumed that each class has a single coherent model.



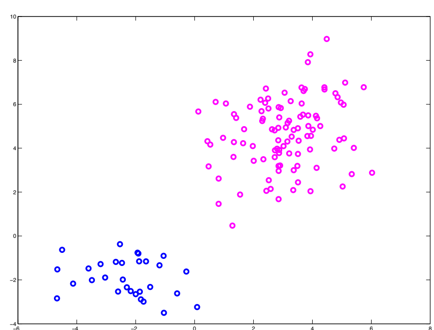
- What if the examples (within the same class) are from a number of distinct "types"?

Examples

- Images of the same person under different conditions: with/without glasses, different expressions, different views.
- Images of the same category but different sorts of objects: chairs with/without armrests.
- Multiple topics within the same document.
- Different ways of pronouncing the same phonemes.

Mixture models

- Assumptions:
  - k underlying types (components);
  - y_i is the identity of the component "responsible" for $\mathbf{x}_i$;
  - y_i is a *hidden* (*latent*) variable: never observed.
- A *mixture model*:



$$p(\mathbf{x};\, \boldsymbol{\pi}) \;=\; \sum_{c=1}^{k} p(y=c)p\left(\mathbf{x} \,|\, y=c\right).$$

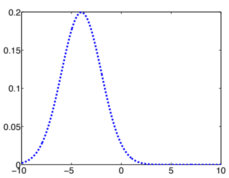
- $\pi_c \triangleq p(y=c)$ are the *mixing probabilities*
- We need to parametrize the component densities $p\left(\mathbf{x} \,|\, y=c\right)$.

Parametric mixtures

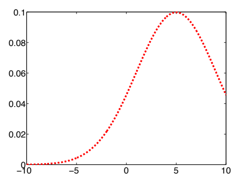
- Suppose that the parameters of the c-th component are θ_c. Then we can denote $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k]$ and write

$$p(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\pi}) \; = \; \sum_{c=1}^{k} \pi_c \cdot p(\mathbf{x};\, \boldsymbol{\theta}_c).$$

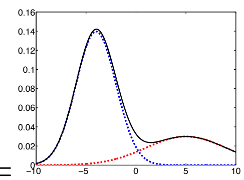
- Any valid setting of $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$, subject to $\sum_{c=1}^{k} \pi_c = 1$, produces a valid pdf.
- Example: mixture of Gaussians.



$\times 0.7 \quad +$



$\times 0.3 \quad = \quad =$



Generative model for a mixture

- The generative process with k-component mixture:
  - The parameters $\boldsymbol{\theta}_c$ for each component c are fixed.
  - Draw $y_i \;\sim\; [\pi_1, \ldots, \pi_k]$;
  - Given y_i, draw $\mathbf{x}_i \;\sim\; p\left(\mathbf{x} \,|\, y_i;\, \boldsymbol{\theta}_{y_i}\right)$.
- The entire generative model for $\mathbf{x}$ and y:

$$p(\mathbf{x}, y;\, \boldsymbol{\theta}, \boldsymbol{\pi}) \; = \; p(y; \boldsymbol{\pi}) \cdot p(\mathbf{x}|y;\, \boldsymbol{\theta}_y)$$

- Any data point $\mathbf{x}_i$ could have been generated in k ways.
- If the c-th component is a Gaussian, $p\left(\mathbf{x} \,|\, y = c\right) = \mathcal{N}\left(\mathbf{x};\, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$,

$$p(\mathbf{x};\, \theta, \boldsymbol{\pi}) \; = \; \sum_{c=1}^{k} \pi_c \cdot \mathcal{N}\left(\mathbf{x};\, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right),$$

where $\theta = [\mu_1, \ldots, \mu_k, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k]$.

Likelihood of a mixture model

- Idea: estimate set of parameters that maximize likelihood given the observed data.
- The log-likelihood of $\boldsymbol{\pi}, \boldsymbol{\theta}$:

$$\log p(X;\, \mathbf{p}, \theta) \; = \; \sum_{i=1}^{N} \log \sum_{c=1}^{k} \pi_c \mathcal{N}\left(\mathbf{x}_i;\, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right).$$

- No closed-form solution because of the sum inside log.
  - We need to take into account all possible components that could have generated $\mathbf{x}_i$.