9 Anisotropic Gaussians, Maximum Likelihood Estimation, QDA, and LDA

ANISOTROPIC GAUSSIANS

[Recall from our last lecture the probability density function of the multivariate normal distribution in its full generality. x and μ are d-vectors]

Normal PDF: $f(x) = n(q(x)), \qquad n(q) = \dfrac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-q/2}, \qquad q(x) = (x-\mu)^\top \Sigma^{-1} (x-\mu)$

$\uparrow \qquad\qquad \uparrow \qquad\qquad \uparrow$

$\mathbb{R} \to \mathbb{R}$, exponential determinant of Σ $\mathbb{R}^d \to \mathbb{R}$, quadratic

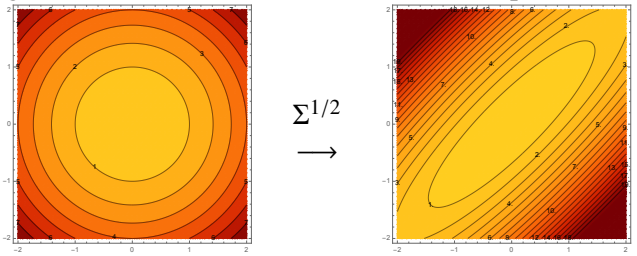
[The covariance matrix Σ and its symmetric square root and its inverse all play roles in our intuition about the multivariate normal distribution. Consider their eigendecompositions.]

$\Sigma = V\Gamma V^\top$ covariance matrix

$\uparrow$ eigenvalues of Σ are variances along the eigenvectors, $\Gamma_{ii} = \sigma_i^2$

$\Sigma^{1/2} = V\Gamma^{1/2}V^\top$ maps spheres to ellipsoids ($\Sigma^{1/2}$ was A in last lecture)

$\uparrow$ eigenvalues of $\Sigma^{1/2}$ are Gaussian widths / ellipsoid radii / standard deviations, $\sqrt{\Gamma_{ii}} = \sigma_i$

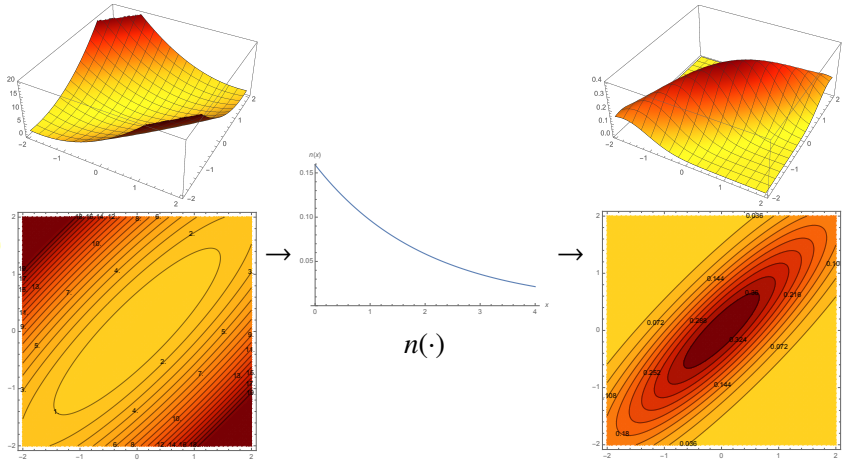


$\Sigma^{1/2} \longrightarrow$

$\longleftarrow$ quadratic form $q(x) = (x-\mu)^\top \Sigma^{-1} (x-\mu)$

$\Sigma^{-1} = V\Gamma^{-1}V^\top$ precision matrix (metric tensor) [$\uparrow$ quadratic form of Σ^{-1} defines contours]

[Recall from last lecture that the isocontours of the multivariate normal distribution are the same as the isocontours of the quadratic form of the precision matrix Σ^{-1}.]



$q(x)$ $\qquad$ $n(\cdot)$ $\qquad$ $f(x) = n(q(x))$

Maximum Likelihood Estimation for Anisotropic Gaussians

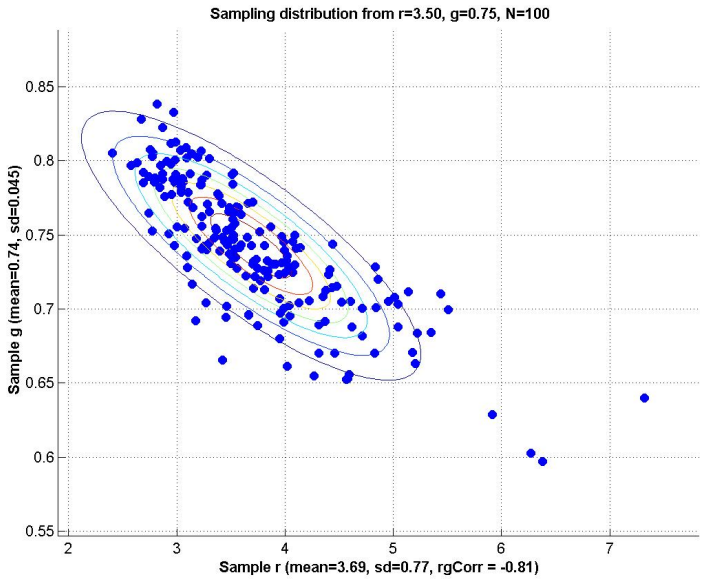
Given sample points $X_1, \ldots, X_n$ and classes $y_1, \ldots, y_n$, find best-fit Gaussians.
X_i is a column vector. [To fit the standard definition of the Gaussian distribution $f(x)$.]

[Once again, we want to fit the Gaussian that maximizes the likelihood of generating the sample points in a specified class. This time I won't derive the maximum-likelihood Gaussian; I'll just tell you the answer.]

For QDA:

$$\hat{\Sigma}_C = \frac{1}{n_C} \sum_{i:y_i=C} \underbrace{(X_i - \hat{\mu}_C)(X_i - \hat{\mu}_C)^\top}_{\text{outer product matrix, } d \times d} \qquad \Leftarrow \text{ conditional covariance for pts in class C}$$

[where n_C is the number of points in class C.]
Prior $\hat{\pi}_C$, mean $\hat{\mu}_C$: same as before
[$\hat{\pi}_C$ is number of points in class C ÷ total sample points; $\hat{\mu}_C$ is mean of sample points in class C.]



maxlike.jpg [Maximum likelihood estimation takes these points and outputs this Gaussian].

$\hat{\Sigma}_C$ is positive semidefinite, but not always definite!
[If there are some zero eigenvalues, the standard version of QDA just doesn't work. We can try to fix it by eliminating the zero-variance dimensions (eigenvectors). Homework 3 suggests a way to do that.]

For LDA:

$$\hat{\Sigma} = \frac{1}{n} \sum_{C} \sum_{i:y_i=C} (X_i - \hat{\mu}_C)(X_i - \hat{\mu}_C)^\top \qquad \Leftarrow \text{ pooled within-class covariance matrix}$$

[Let's revisit QDA and LDA and see what has changed now that we know anisotropic Gaussians. The short answer is "not much has changed, but the graphs look cooler." By the way, capital X once again represents a random variable.]

QDA

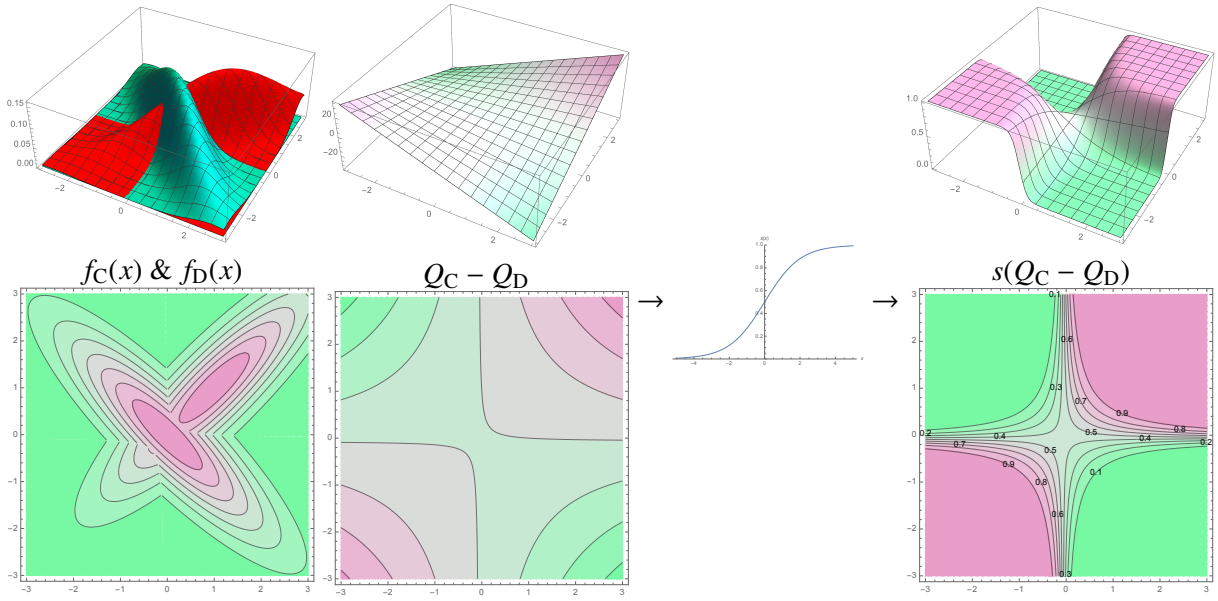
Choosing C that maximizes $f(X = x|Y = \mathrm{C})\,\pi_\mathrm{C}$ is equivalent to maximizing the quadratic discriminant fn

$$Q_\mathrm{C}(x) = \ln\Big((\sqrt{2\pi})^d\, f_\mathrm{C}(x)\,\pi_\mathrm{C}\Big) = -\frac{1}{2}(x-\mu_\mathrm{C})^\top \Sigma_\mathrm{C}^{-1}\,(x-\mu_\mathrm{C}) - \frac{1}{2}\ln|\Sigma_\mathrm{C}| + \ln\pi_\mathrm{C}$$

↑ Gaussian PDF for C

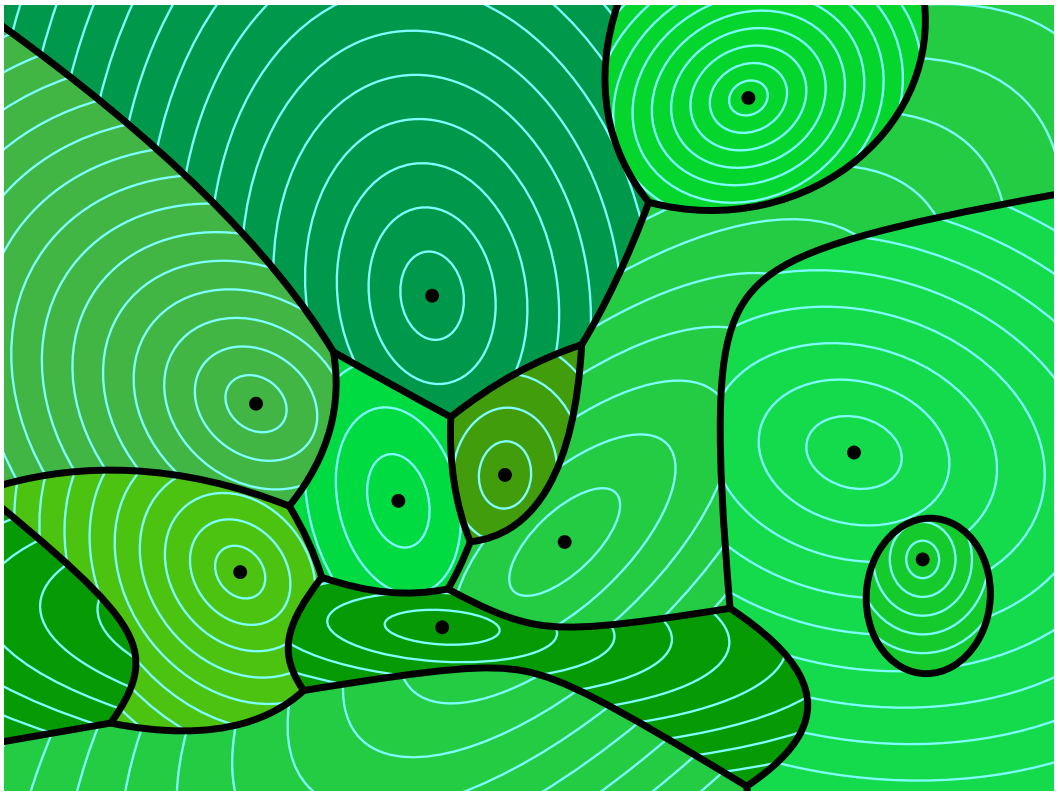
[This works for any number of classes. In a multi-class problem, you just pick the class with the greatest quadratic discriminant for x.]

2 classes: Decision fn $Q_\mathrm{C}(x) - Q_\mathrm{D}(x)$ is quadratic, but may be indefinite
⇒ Bayes decision boundary is a quadric.
Posterior is $P(Y = \mathrm{C}|X = x) = s(Q_\mathrm{C}(x) - Q_\mathrm{D}(x))$ where $s(\cdot)$ is logistic fn



qdaaniso3d.pdf, qdaanisocontour.pdf, qdaanisodiff3d.pdf, qdaanisodiffcontour.pdf, logistic.pdf, qdaanisoposterior3d.pdf, qdaanisoposteriorcontour.pdf

[(Show this figure on a separate "whiteboard.") An example where the decision boundary is a hyperbola—which is not possible with isotropic Gaussians. At left, two anisotropic Gaussians. Center left, the difference $Q_\mathrm{C} - Q_\mathrm{D}$. After applying the logistic function to this difference we obtain the posterior probabilities at right, which tells us the probability our prediction is correct. Observe that we can see the decision boundary in both contour plots: it is $Q_\mathrm{C} - Q_\mathrm{D} = 0$ and $s(Q_\mathrm{C} - Q_\mathrm{D}) = 0.5$. We don't need to apply the logistic function to find the decision boundary, but we do need to apply it if we want the posterior probabilities.]



aniso.pdf [When you have many classes, their QDA decision boundaries form an anisotropic Voronoi diagram. Interestingly, a cell of this diagram might not be connected.]

LDA

One $\hat{\Sigma}$ for all classes.

[Once again, the quadratic terms cancel each other out so the decision function is linear and the decision boundary is a hyperplane.]

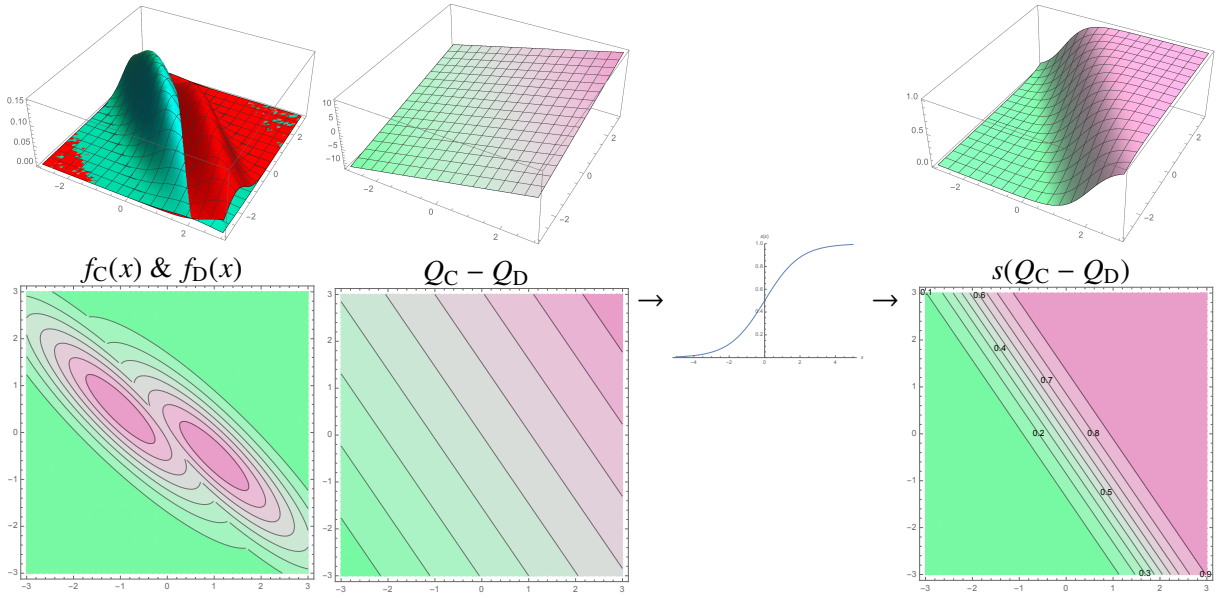
$$Q_C(x) - Q_D(x) = \underbrace{(\mu_C - \mu_D)^\top \Sigma^{-1}}_{w^\top} x \underbrace{- \frac{\mu_C^\top \Sigma^{-1} \mu_C - \mu_D^\top \Sigma^{-1} \mu_D}{2} + \ln \pi_C - \ln \pi_D}_{+\alpha}$$

Choose class C that maximizes the linear discriminant fn

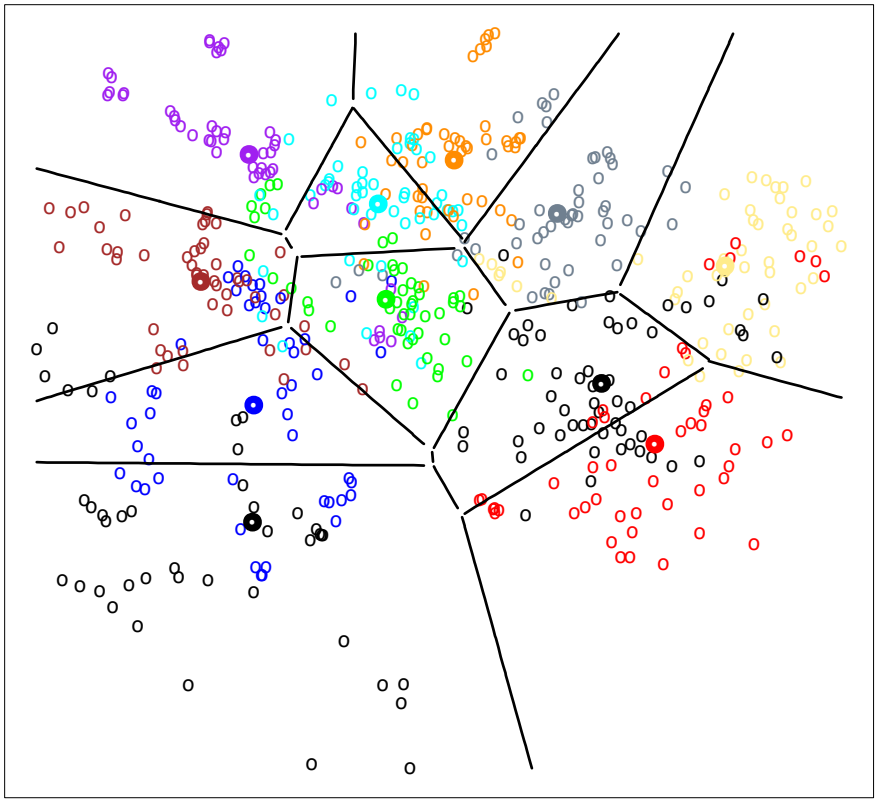
$$\mu_C^\top \Sigma^{-1} x - \frac{1}{2} \mu_C^\top \Sigma^{-1} \mu_C + \ln \pi_C \qquad \text{[works for any \# of classes]}$$

2 classes: Decision boundary is $w^\top x + \alpha = 0$
Posterior is $P(Y = C|X = x) = s(w^\top x + \alpha)$

[Note that we use a linear solver to efficiently compute $\mu_C^\top \Sigma^{-1}$ just once, so the classifier can evaluate test points quickly.]



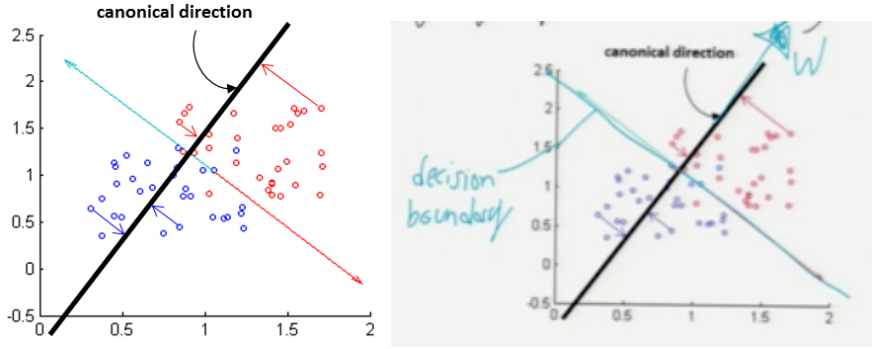
ldaaniso3d.pdf, ldaanisocontour.pdf, ldaanisodiff3d.pdf, ldaanisodiffcontour.pdf, logistic.pdf, ldaanisoposterior3d.pdf, ldaanisoposteriorcontour.pdf
[(Show this figure on a separate "whiteboard.") In LDA, the decision boundary is always a hyperplane. Note that Mathematica messed up the top left plot a bit; there should be no red in the left corner, nor blue in the right corner.]



LDAdata.pdf (ESL, Figure 4.11) [An example of LDA with messy data. The points are not sampled from perfect Gaussians, but LDA still works reasonably well.]

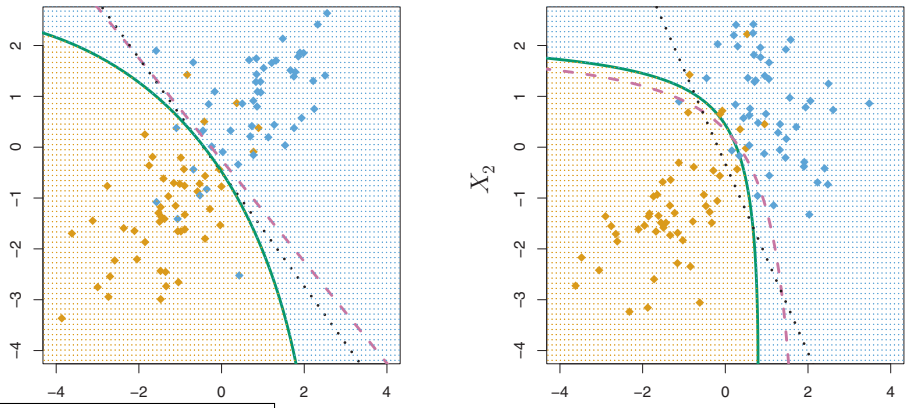
Notes:

– LDA often interpreted as projecting points onto normal w; cutting the line in half.



project.png

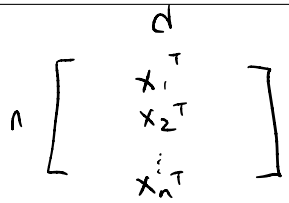
– For 2 classes,
  – LDA has $d + 1$ parameters (w, α);
  – QDA has $\frac{d(d+3)}{2} + 1$ params;
  – QDA more likely to overfit. [The danger is much bigger when the dimension d is large.]



ldaqda.pdf (ISL, Figure 4.9) [In these examples, the Bayes optimal decision boundary is purple (and dashed), the QDA decision boundary is green, the LDA decision boundary is black (and dotted). When the optimal boundary is linear, as at left, LDA gives a more stable fit whereas QDA may overfit. When the optimal boundary is curved, as at right, QDA often gives you a better fit.]

– With added features, LDA can give nonlinear boundaries; QDA nonquadratic.
– We don't get **true** optimum Bayes classifier
  – estimate distributions from finite data
  – real-world data not perfectly Gaussian
– Changing priors or loss = adding constants to discriminant fns
  [So it's very easy. In the 2-class case, it's equivalent to changing the isovalue . . .]
– Posterior gives decision boundaries for 10% probability, 50%, 90%, etc.
  – choosing isovalue = probability p is same as choosing asymmetrical loss p for false positive, $1 - p$ for false negative; OR as choosing $\pi_C = 1 - p$, $\pi_D = p$.

[LDA & QDA are the best method in practice for many applications. In the STATLOG project, either LDA or QDA were in the top three classifiers for 10 out of 22 datasets. But it's not because all those datasets are Gaussian. LDA & QDA work well when the data can only support simple decision boundaries such as linear or quadratic, because Gaussian models provide stable estimates. See ESL, Section 4.3]



Some Terms

Let X be $n \times d$ design matrix of sample pts
Each row i of X is a sample pt $X_i^\top$.

[Now I'm using capital X as a matrix instead of a random variable vector. I'm treating X_i as a column vector to match the standard convention for multivariate distributions like the Gaussian, but $X_i^\top$ is a row of X.]

centering X: subtracting $\mu^\top$ from each row of X. $X \to \dot{X}$
[$\mu^\top$ is the mean of all the rows of X. Now the mean of all the rows of $\dot{X}$ is zero.]

Let R be uniform distribution on sample pts. Sample covariance matrix is

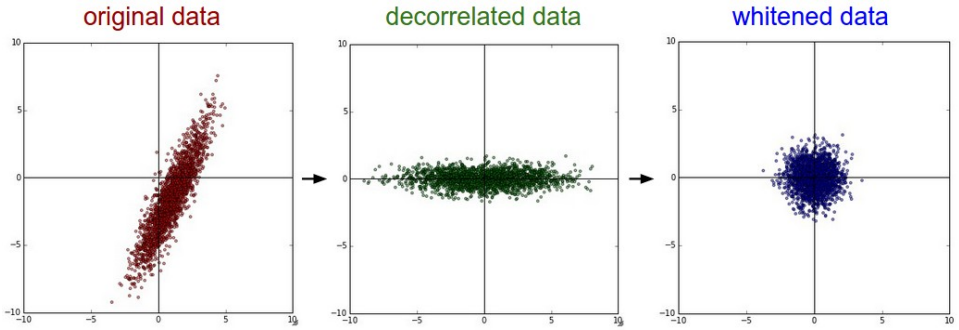
$$\mathrm{Var}(R) = \frac{1}{n}\dot{X}^\top\dot{X}.$$

[This is the simplest way to remember how to compute a covariance matrix for QDA. Imagine you have a design matrix X_C that contains only the sample points of class C; then you have $\hat{\Sigma}_C = \frac{1}{n_C}\dot{X}_C^\top\dot{X}_C$.]

[When we have points from an anisotropic Gaussian distribution, sometimes it's useful to perform a linear transformation that maps them to an axis-aligned distribution, or maybe even to an isotropic distribution.]

decorrelating $\dot{X}$: applying rotation $Z = \dot{X}V$, where $\mathrm{Var}(R) = V\Lambda V^\top$
[rotates the sample points to the eigenvector coordinate system]

Then $\mathrm{Var}(Z) = \Lambda$. [$Z$ has diagonal covariance. If $X_i \sim \mathcal{N}(\mu, \Sigma)$, then approximately, $Z_i \sim \mathcal{N}(0, \Lambda)$.]
[Proof: $\mathrm{Var}(Z) = \frac{1}{n}Z^\top Z = \frac{1}{n}V^\top\dot{X}^\top\dot{X}V = V^\top\mathrm{Var}(R)V = V^\top V\Lambda V^\top V = \Lambda$.]



whiten.jpg

sphering $\dot{X}$: applying transform $W = \dot{X}\,\mathrm{Var}(R)^{-1/2}$ [Recall that $\Sigma^{-1/2}$ maps ellipsoids to spheres.]

whitening X: centering + sphering, $X \to W$

Then W has covariance matrix I. [If $X_i \sim \mathcal{N}(\mu, \Sigma)$, then approximately, $W_i \sim \mathcal{N}(0, I)$.]

[Whitening input data is often used with other machine learning algorithms, like SVMs and neural networks. The idea is that some features may be much bigger than others—for instance, because they're measured in different units. SVMs penalize violations by large features more heavily than they penalize small features. Whitening the data before you run an SVM puts the features on an equal basis.]

[One nice thing about discriminant analysis is that whitening is built in.]

[Incidentally, what we've done here—computing a sample covariance matrix and its eigenvectors/values—is about 75% of an important unsupervised learning method called principal components analysis, or PCA, which we'll learn later in the semester.]