Linear Classification: Logistic Regression

MATH 697 AM:ST

October 24th, 2017

Warmup

Hyperplanes and distances

One way of describing a hyperplane H in $\mathbb{R}^p$ is as follows:

Take a pair (β_0, β) where $\beta_0 \in \mathbb{R}$ and $\beta \in \mathbb{R}^p \setminus \{0\}$

$$H = \{x \in \mathbb{R}^p \mid \beta_0 + (\beta, x) = 0\}$$

Observe that (β_0, β) and (β_0', β') describe the same hyperplane if and only if β and β' are colinear and $\beta_0 = \beta_0'$. Therefore, without loss of generality, we often assume that $\|\beta\|_2 = 1$.

Warmup

Hyperplanes and distances

We will in fact, consider **oriented hyperplanes**. The positive side of H will be the side to where β is pointing, and the other side will be called the negative side, accordingly.

In particular, given β the hyperplanes corresponding to $(0, \beta)$ and $(0, -\beta)$ are different, and in fact will be said to have opposite orientations.

Warmup

Hyperplanes and distances

Consider H given by (β_0, β) and $x \in \mathbb{R}^p$ on the positive side of H, then we have

$$\text{dist}(x, H) = \beta_0 + (\beta, x)$$

(assuming, that is, that $\|\beta\| = 1$)

Accordingly, if x lies on the negative side of H, then

$$\text{dist}(x, H) = -(\beta_0 + (\beta, x))$$

Warmup

Cutting space with hyperplanes

In general (i.e. β may not be normalized), the (signed) distance to H is given by

$$\operatorname{dist}(x, H) = \frac{\beta_0 + (\beta, x)}{\|\beta\|}$$

Classification for Gaussian mixtures

RECAP from last class

Consider a model where the points $\{x_1, \ldots, x_N\}$ are being drawn in an i.i.d. manner from some underlying probability distribution.

Denoting by X a random variable distributed by this probability distribution, the Bayesian classifier for a given input $x \in \mathbb{R}^p$ is

$$\hat{G}(x) = \operatorname*{argmax}_k \mathbb{P}(G = k \mid X = x)$$

Classification for Gaussian mixtures

RECAP from last class

When the points arise from a mixture of Gaussians, one can determine $\hat{G}$ entirely from linear operations.

For a Gaussian mixture, we have (via Bayes theorem)

$$\mathbb{P}(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$

Therefore

$$\frac{\mathbb{P}(G = k \mid X = x)}{\mathbb{P}(G = \ell \mid X = x)} = \frac{f_k(x)\pi_k}{f_\ell(x)\pi_\ell}$$

Classification for Gaussian mixtures

RECAP from last class

Since,

$$f_k(x) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\Sigma_k^{-1}(x-\mu_k), x-\mu_k)} \pi_k$$

taking the logarithm is advantageous, so

$$\log\left(\frac{\mathbb{P}(G = k \mid X = x)}{\mathbb{P}(G = \ell \mid X = x)}\right)$$
$$= \log\left(\frac{|\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\Sigma_k^{-1}(x-\mu_k), x-\mu_k)} \pi_k}{|\Sigma_\ell|^{-\frac{1}{2}} e^{-\frac{1}{2}(\Sigma_k^{-1}(x-\mu_\ell), x-\mu_\ell)} \pi_\ell}\right)$$

Classification for Gaussian mixtures

RECAP from last class

If all the Σ_k are all equal to some Σ:

$$\log\left(\frac{\mathbb{P}(G=k \mid X=x)}{\mathbb{P}(G=\ell \mid X=x)}\right)$$
$$= \log\left(\frac{\pi_k}{\pi_\ell}\right) + \log\left(\frac{e^{-\frac{1}{2}(\Sigma^{-1}(x-\mu_k), x-\mu_k)}}{e^{-\frac{1}{2}(\Sigma^{-1}(x-\mu_\ell), x-\mu_\ell)}}\right)$$

Classification for Gaussian mixtures

RECAP from last class

and we saw that

$$\log\left(\frac{\mathbb{P}(G=k \mid X=x)}{\mathbb{P}(G=\ell \mid X=x)}\right) = \delta_k(x) - \delta_\ell(x)$$

where the discriminant function δ_k is given by

$$\delta_k(x) = \log(\pi_k) + \frac{1}{2}(\Sigma^{-1}\mu_k, \mu_k) + (\Sigma^{-1}\mu_k, x)$$

Classification for Gaussian mixtures

RECAP from last class

Given a training set $\{x_1, \ldots, x_N\}$ with labels $g_i \in \{1, \ldots, K\}$, we have the estimators

$$\hat{\pi}_k = \frac{N_k}{N}, \quad N_k := \#\{i : g_i = k\}$$

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i:g_i=k} x_i$$

$$\hat{\Sigma}_{ab} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{i:g_i=k} (x_i - \hat{\mu}_k)_a (x_i - \hat{\mu}_k)_b$$

Classification for Gaussian mixtures

RECAP from last class

and we have the "estimated" discriminant functions, which are linear

$$\hat{\delta}_k(x) = \log(\hat{\pi}_k) + \frac{1}{2}(\hat{\Sigma}^{-1}\hat{\mu}_k, \mu_k) + (\hat{\Sigma}^{-1}\hat{\mu}_k, x)$$

Then, the classifier is

$$\hat{G}(x) = \operatorname*{argmax}_{k \in K} \hat{\delta}_k(x).$$

Classification for Gaussian mixtures

RECAP from last class

Let us go back to the case where we allow $\Sigma_k \neq \Sigma_\ell$, and consider

$$\log\left(\frac{\mathbb{P}(G=k \mid X=x)}{\mathbb{P}(G=\ell \mid X=x)}\right)$$

and we have that is equal to

$$\log\left(\frac{\pi_k}{\pi_\ell}\right) - \frac{1}{2}\log\left(\frac{|\Sigma_k|}{|\Sigma_\ell|}\right) - \frac{1}{2}(\Sigma_k^{-1}(x-\mu_k), x-\mu_k)$$
$$+ \frac{1}{2}(\Sigma_\ell^{-1}(x-\mu_\ell), x-\mu_\ell)$$

there aren't as many cancelations as in the case of a single Σ...

Classification for Gaussian mixtures

RECAP from last class

…but still, we have

$$\log\left(\frac{\mathbb{P}(G=k \mid X=x)}{\mathbb{P}(G=\ell \mid X=x)}\right) = \delta_k(x) - \delta_\ell(x)$$

where this time, $\delta_k(x)$ are quadratic functions

$$\delta_k(x) = \log\left(\pi_k\right) - \frac{1}{2}\log(|\Sigma_k|) - \frac{1}{2}(\Sigma_k^{-1}(x-\mu_k), x-\mu_k)$$

Classification for Gaussian mixtures

RECAP from last class

As before, when using this algorithm in practice, we take

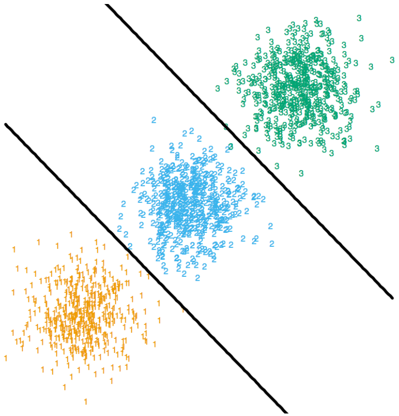
$$\hat{\delta}_k(x) := \log(\hat{\pi}_k) - \frac{1}{2}\log(|\hat{\Sigma}_k|) - \frac{1}{2}(\hat{\Sigma}_k^{-1}(x-\hat{\mu}_k), x-\hat{\mu}_k)$$

where $\hat{\pi}_k$, $\hat{\mu}_k$ and $\hat{\Sigma}_k^{-1}$ are the standard estimators, and then

$$\hat{G}(x) = \operatorname*{argmax}_k \hat{\delta}_k(x).$$

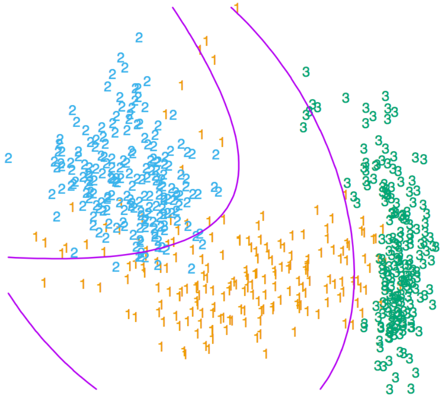
Classification for Gaussian mixtures

Examples



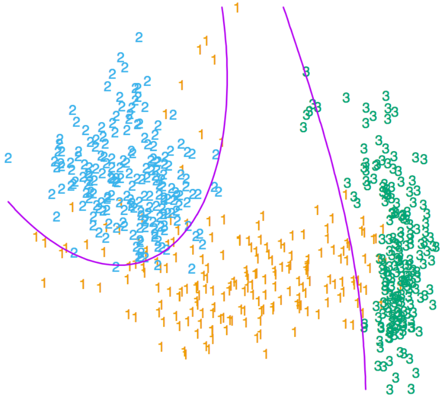
Linear Discriminant Analysis Vs. masking.

Classification for Gaussian mixtures



(for comparison) Regression with quadratic functions

Classification for Gaussian mixtures



(for comparison) and the result via QDA

Logistic regression

The regression on the indicator approach to the indicator matrix has a serious drawback: the discriminant functions given by posterior probabilities

$$\mathbb{P}(G = k \mid X = x)$$

are not going to be even close to linear, so our intent to approximate them via linear functions is somewhat misguided. Worse still, the discriminant functions $\delta_k(x)$ obtained in this way won't satisfy

$$0 \leq \delta_k \text{ and } \sum_{k=1}^{K} \delta_k(x) = 1$$

Logistic regression

This justifies in part the consideration of a logistic transformation, and assuming that

$$\mathbb{P}(G = k \mid X = x) = \frac{e^{L_k(x)}}{1 + \sum_{\ell=1}^{K-1} e^{L_\ell(x)}}$$

and

$$\mathbb{P}(G = K \mid X = x) = \frac{1}{1 + \sum_{j=1}^{K-1} e^{L_j(x)}}$$

where the L_k's are $K-1$ affine functions, written as

$$L_k(x) = \beta_{k,0} + (\beta_k, x)$$

Logistic regression

Observe that the last probability can be written as

$$\mathbb{P}(G = K \mid X = x) = \frac{e^{L_K(x)}}{1 + \sum_{j=1}^{K-1} e^{L_j(x)}}$$

if we take $L_K(x)$ to be trivial affine function, $L_K(x) \equiv 0$. As such, going forward we think of having actually K affine functions, where

$$L_k(x) = \beta_{K,0} + (\beta_K, x),\ \beta_{K,0} = 0,\ \beta_K = 0.$$

Logistic regression

Then, with the convention $\beta_{K,0} = 0, \beta_K = 0$, there is a more succinct expression

$$\mathbb{P}(G = k \mid X = x) = \frac{e^{L_K(x)}}{\sum_{j=1}^{K} e^{L_j(x)}}$$

for all $k \in \{1, \ldots, K\}$ and all x.

Logistic regression

Observe that these are indeed probabilities: they are all non-negative and adding them up in k yields 1.

It is customary to write, for $\mathbb{P}(G = k \mid X = x)$

$$p_k(x; \theta)$$

where θ is the vector of weights $(\beta_{1,0}, \beta_1, \ldots, \beta_{K,0}, \beta_K)$.

Logistic regression

Practical Considerations: Training Data

For classification, our training data is of the form $\{(x_i, g_i)\}_{i=1}^N$ where x_i are points in the input space, and $g_i \in \{1, \ldots, K\}$.

Given such a data set, then the likelihood is given by

$$\prod_{i=1}^{N} p_{g_i}(x_i; \theta)$$

and the log-likelihood function is

$$F(\theta) = \sum_{i=1}^{N} \log(p_{g_i}(x_i; \theta))$$

Logistic regression

Practical Considerations: Training Data

This is where having taken the logistic function on linear data pays off, since

$$\log(p_{g_i}(x_i;\theta)) = L_{g_i}(x_i) - \log\left(\sum_{j=1}^{K} e^{L_j(x_i)}\right)$$

that is, each is $\log(p_{g_i}(x_i;\theta))$ is equal to

$$\beta_{g_i,0} + (\beta_{g_i}, x_i) - \log\left(\sum_{j=1}^{K} e^{L_j(x_i)}\right)$$

Logistic regression

Practical Considerations: Training Data

Thus, the log-likelihood functional can be written as

$$F(\theta) := \sum_{i=1}^{N} \left\{ \beta_{g_i,0} + (\beta_{g_i}, x_i) - \log\left(\sum_{j=1}^{K} e^{L_j(x_i)} \right) \right\}$$

$$L_j(x_i) = \beta_{j,0} + (\beta_j, x_i), \quad \theta = (\beta_{1,0}, \beta_1, \ldots, \beta_{K,0}, \beta_K)$$

Now, we choose $\hat{\theta}$ as

$$\begin{aligned} \hat{\theta} &:= \operatorname*{argmax}_{\theta} F(\theta) \\ &= -\operatorname*{argmin}_{\theta}(-F(\theta)) \end{aligned}$$

Observe that $F(\theta)$ is a concave function.

Logistic regression

Maximizing the log-Likelihood

The concavity of $F(\theta)$ means that maximizing $F(\theta)$ yields a convex minimization problem ($-F(\theta)$ is convex).

Gradient flow (continuous):
Choose an initial guess θ_0 for the parameters, then follow its evolution through the differential equation

$$\dot{\theta} = -\nabla_\theta(-F) = \nabla_\theta F$$

As $t \to \infty$, $\theta(t)$ is expected to converge to the maximum of F.

Logistic regression

Maximizing the log-Likelihood

The concavity of $F(\theta)$ means that maximizing $F(\theta)$ yields a convex minimization problem ($-F(\theta)$ is convex).

Gradient flow (discrete):
Fix a discretization step* $\rho > 0$, then define iteratively

$$\theta_{k+1} := \theta_k + \rho \nabla F(\theta_k), \;\; k = 0, 1, 2 \ldots$$

* in the statistical inference literature this is known as the **learning rate**.

Logistic regression

Maximizing the log-Likelihood

Another approach is to of course find directly the solution $\theta*$

$$\nabla F(\theta*)$$

via other iteration methods, such as the Newton-Raphson algorithm.

Logistic regression

The Case Of Two Classes

For two classes, that is $K = 2$, the parameters reduce to a single scalar β_0 and a single slope vector $\beta \in \mathbb{R}^p$

$$p_1(x;\theta) = \mathbb{P}[G = 1 \mid X = x] = \frac{e^{\beta_0 + (\beta, x)}}{1 + e^{\beta_0 + (\beta, x)}}$$
$$p_2(x;\theta) = \mathbb{P}[G = 2 \mid X = x] = \frac{1}{1 + e^{\beta_0 + (\beta, x)}}$$

Of course, in this instance we have $p_2(x;\theta) = 1 - p_1(x;\theta)$.

Logistic regression

The Case Of Two Classes

$$\log(p_1(x;\theta)) = \beta_0 + (\beta, x) - \log(1 + e^{\beta_0 + (\beta, x)})$$
$$\log(p_2(x;\theta)) = -\log(1 + e^{\beta_0 + (\beta, x)})$$

Using indicator variables y_i for the class $G = 1$ ($y_i = 1$ of $g_i = 1$, $y_i = 0$ otherwise), we have

$$\begin{aligned} F(\beta_0, \beta) = \sum_{i=1}^{N} y_i(\beta_0 + (\beta, x_i) - \log(1 + e^{\beta_0 + (\beta, x_i)})) \\ - \sum_{i=1}^{N} (1 - y_i) \log(1 + e^{\beta_0 + (\beta, x_i)}) \end{aligned}$$

Logistic regression

The Case Of Two Classes

Simplifying, we have

$$F(\beta_0, \beta) = \sum_{i=1}^{N} y_i(\beta_0 + (\beta, x_i)) - \log(1 + e^{\beta_0 + (\beta, x_i)})$$

Since we may rewrite $\beta_0 + (\beta, x_i)$ as an inner product in $\mathbb{R}^{p+1}$, we may assume without loss of generality $\beta_0 = 0$, thus

$$F(\beta) = \sum_{i=1}^{N} y_i(\beta, x_i) - \log(1 + e^{(\beta, x_i)})$$

Logistic regression

The Case Of Two Classes

Let us compute the gradient, we have

$$\nabla F(\beta) = \sum_{i=1}^{N} y_i x_i - \nabla \left(\sum_{i=1}^{N} \log(1 + e^{(\beta, x_i)}) \right)$$

Logistic regression

The Case Of Two Classes

Let us compute the gradient, we have

$$\nabla F(\beta) = \sum_{i=1}^{N} y_i x_i - \nabla \left(\sum_{i=1}^{N} \log(1 + e^{(\beta, x_i)}) \right)$$

Simplifying

$$\nabla F(\beta) = \sum_{i=1}^{N} y_i x_i - \left(\sum_{i=1}^{N} \frac{e^{(\beta, x_i)} x_i}{1 + e^{(\beta, x_i)}} \right)$$

Logistic regression

The Case Of Two Classes

Let us compute the gradient, we have

$$\nabla F(\beta) = \sum_{i=1}^{N} y_i x_i - \nabla \left(\sum_{i=1}^{N} \log(1 + e^{(\beta, x_i)}) \right)$$

Simplifying

$$\nabla F(\beta) = \sum_{i=1}^{N} y_i x_i - \left(\sum_{i=1}^{N} \frac{e^{(\beta, x_i)} x_i}{1 + e^{(\beta, x_i)}} \right)$$

and we recognize the term on the right as $p_1(x_i; \beta) x_i$ so

$$\nabla F(\beta) = \sum_{i=1}^{N} (y_i - p_1(x_i; \beta)) x_i$$

Logistic regression

The Case Of Two Classes

As such, for a learning rate $\rho > 0$, the gradient descent algorithm takes the form

$$\beta_{k+1} = \beta_k + \rho \sum_{i=1}^{N} (y_i - p_1(x_i; \beta)) x_i$$

Compare the expression on the right to one-dimensional least squares!

(See Hastie-Tibshirani-Friedman for a review of another basic approach using the Newton-Raphson method)

Logistic regression VS. LDA

When doing Linear Discriminant Analysis, we have

$$\log\left(\frac{\mathbb{P}[G=k \mid X=x]}{\mathbb{P}[G=K \mid X=x]}\right) = \alpha_{k0} + (\alpha_k, x)$$

where

$$\begin{aligned} \alpha_{k0} &= \log\left(\frac{\pi_k}{\pi_K}\right) - \frac{1}{2}\left(\Sigma^{-1}(\mu_k+\mu_K), \mu_k-\mu_K\right) \\ \alpha_k &= \Sigma^{-1}(\mu_k-\mu_K) \end{aligned}$$

and this was a direct consequence of modeling our data via a Gaussian mixture with common covariance matrix Σ.

Logistic regression VS. LDA

Meanwhile, in K-class logistic regression,

$$\frac{\mathbb{P}[G = k \mid X = x]}{\mathbb{P}[G = K \mid X = x]} = \frac{e^{\beta_{0k} + (\beta_k, x)}}{e^{\beta_{0K} + (\beta_K, x)}}$$

Therefore,

$$\log\left(\frac{\mathbb{P}[G = k \mid X = x]}{\mathbb{P}[G = K \mid X = x]}\right) = \alpha_{k0} + (\alpha_k, x)$$

where this time

$$\begin{aligned} \alpha_{k0} &= \beta_{k0} - \beta_{K0} \\ \alpha_k &= \beta_k - \beta_K \end{aligned}$$

Logistic regression VS. LDA

So, in both models we have the same form for

$$\frac{\mathbb{P}[G = k \mid X = x]}{\mathbb{P}[G = K \mid X = x]} = \frac{e^{\beta_{0k} + (\beta_k, x)}}{e^{\beta_{0K} + (\beta_K, x)}}$$

Where is the difference, if any? Well, in LDA there is extra information: the X are distributed via a Gaussian mixture, from where it follows in particular that the conditional probabilities have the above form.

As such, the logistic regression is more general, and contains LDA as a special case –this doesn't mean LDA does not stand squarely on its own, of course.

Logistic regression VS. LDA

It is debatable how much more is gained by the assumption of a Gaussian mixture, versus the LDA assumption, and there is not wide agreement on this.

However, there seems to be an agreement that Logistic regression is the more robust modeling choice in general.

Now, an entirely different perspective on the use of hyperplanes to separate data points: the perceptron

Perceptrons

The Perceptron (Rosenblatt 1958)

$$\begin{aligned}&\text{Input: } x \in \mathbb{R}^p,\\ &\text{Output: } \operatorname{sign}(\beta_0 + (\beta, x))\end{aligned}$$

Basically, an analytic (an arguably pedantic) way of writing "we are going to choose a hyperplane H and classify points in one of two classes according to on what side of H the are".

Nothing new so far, the point is in how we choose β.

Perceptrons

Suppose we are given training data $\{x_i, g_i\}_{i=1}^N$, with $g_i \in \{1, 2\}$ being the class x_i belongs to.

Rosenblatt's idea:

Instead of penalizing all mislabeled points equal ($0 - 1$ Loss Function), **weight the contribution** of each mislabeled point **according to how far** it is from the separating hyperplane.

Perceptrons

Let $\mathcal{M} = \mathcal{M}(\beta_0, \beta)$ be the set of indices of misclassified points,

$$\mathcal{M} = \{i \mid y_i \text{ and } \beta_0 + (x_i, \beta) \text{ have opposing signs}\}$$

Here, y_i is defined as follows

$$y_i = 1 \text{ if } g_i = 1,\; y_i = -1 \text{ otherwise.}$$

Then, we can write

$$\begin{aligned} D(\beta_0, \beta) &:= \sum_{i=1}^{N} |\beta_0 + (x_i, \beta)| \chi_{\mathcal{M}}(x_i) \\ &= -\sum_{i \in \mathcal{M}} y_i(\beta_0 + (x_i, \beta)) \end{aligned}$$

Perceptrons

The gradient of $D(\beta_0, \beta)$ is as follows

$$\nabla_{\beta_0} D(\beta_0, \beta) = -\sum_{i \in \mathcal{M}} y_i, \quad \nabla D(\beta_0, \beta) = -\sum_{i \in \mathcal{M}} y_i x_i$$

(note that this perspective necessitates we allow β to have norm different from 1)

Perceptrons

Known drawbacks of perceptrons

1. Solution may depend heavily on initial guess (non-uniqueness of limit).
2. The algorithm may fail to converge at all (non-existence of a limit).
3. The number of steps needed for convergence may be very large.

Linear Classification Wrap up and onto PDE!

MATH 697 AM:ST

October 26th, 2017

Vapnik's Optimality Criterion

Recap

An alternative criterion for choosing a hyperplane was proposed by Vapnik in 1996.

Choose H so that the distance from H to the closest data point is maximized

Vapnik's Optimality Criterion

If H is given by (β_0, β), and $\|\beta\|_2 = 1$, the closest distance from H to the points $\{x_i\}_{i=1}^N$ is given by

$$\begin{aligned} J(\beta_0, \beta) &:= \max\{M \mid M \leq |\beta_0 + (x_i, \beta)|,\ i = 1, \ldots, N\} \\ &= \max\{M \mid M \leq y_i(\beta_0 + (x_i \beta)),\ i = 1, \ldots, N\} \end{aligned}$$

where as usual, $y_i = 1$ or $y_i = -1$ according to whether x_i lies in class #1 or #2.

Vapnik's Optimality Criterion

For $\|\beta\|_2 \neq 1$, we normalize the vector and use the same definition of $J(\cdot)$ from before, and we obtain

$$J(\beta_0, \beta) = \max\{M \mid M\|\beta\|_2 \leq y_i(\beta_0 + (x_i\beta)),\ i = 1, \dots, N\}$$

So, we have the maximization problem

$$\max_{\beta_0, \beta} J(\beta_0, \beta)$$

This, turns out, can be recast as a quadratic optimization problem with linear constraints.

Vapnik's Optimality Criterion

Advantages/drawbacks

1. The problem can be cast as linear program, lots of methods available to solve it.
2. The resulting classifier often agrees with logistic regression.
3. Does not use "far away" data in determining the boundary, so outliers do not affect the outcome.
4. Depending on the data model we are dealing with, if the data is not well separated then the separating hyperplane may yield a high error rate (i.e. two Gaussians with close by means and common covariance matrix).

Part II: PDE and Calculus of Variations methods

Elliptic Equations: Discrete and Continuous Settings

1. This approach attempts to exploit geometry that tends to be intrinsic to the data (the data is not necessarily embedded as points in $\mathbb{R}^{p+1} \ldots$)
2. Highly motivated by PDE, Probability, and the Calculus of Variations, Harmonic Analysis.
3. Initially of great success in image and signal processing.
4. Allows one to consider notions of smoothness in discrete settings.

Keywords going forward: Diffusion, weighted graphs, spectral graph theory, diffusion wavelets, manifold learning, nonlinear dimensionality reduction, Fourier analysis.

The Laplacian

In $\mathbb{R}^d$

Let us recall two operators:

$$\begin{aligned} \nabla &: \text{(scalar fields)} \mapsto \text{(vector fields)} \\ \text{div} &: \text{(vector fields)} \mapsto \text{(scalar fields)} \end{aligned}$$

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

$$\text{div}(X) = \frac{\partial X_1}{\partial x_1} + \ldots + \frac{\partial X_n}{\partial x_n}$$

The Laplacian

In $\mathbb{R}^d$

The Laplacian is defined as the composition of these two

$$\Delta f = \operatorname{div}(\nabla f)$$

This operator has several equivalent expressions, emphasizing its different roles across topics, e.g.

$$\frac{\partial^2 f}{\partial x_1^2} + \ldots + \frac{\partial^2 f}{\partial x_n^2}$$

or

$$\operatorname{tr}(D^2 f)$$

where of course, $(D^2 f(x))_{ij} = \frac{\partial^2}{\partial x_i x_j} f(x)$ is the Hessian matrix.

The Laplacian

In $\mathbb{R}^d$

The Laplacian $\Delta f(x)$ measures the **infinitesimal mean oscillation of** f at x, by which we mean that

$$\Delta f(x) = \lim_{r \to 0^+} \frac{1}{r^2 |\partial B_r(x)|} \int_{\partial B_r(x)} f(y) - f(x)\, d\sigma(x)$$

The Laplacian

In a Riemannian Manifold

(Other settings beyond $\mathbb{R}^d$ for later)

The Laplace-Beltrami operator
Given a **Riemannian metric** $g_{ij}(x)$ in a domain $D \subset \mathbb{R}^d$, we write for any $f : D \mapsto \mathbb{R}$

$$\begin{aligned}\Delta_g f(x) &= \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial x_i} \left(g^{ij} \sqrt{|g|} \frac{\partial f}{\partial x_j} \right) \\ &= \mathrm{div}_g(\nabla_g f)\end{aligned}$$

The Laplacian

In a weighted graph

(Other settings beyond $\mathbb{R}^d$ for later)

Graph Laplacian
Let G be a graph with weight matrix w_{ij}, then the Laplacian for this graph is defined by

$$\Delta f_i = \sum_j w_{ij}(f_j - f_i)$$

For the rest of today:

1. Some background on Fourier analysis and the heat equation.
2. The smoothing effect of the heat equation: examples.
3. Eigenfunctions of the Laplacian
4. Harmonic functions and the mean value property
5. Graphs and their Laplacians

The eight week in one slide

1. Logistic regression and maximum likelihood gives us a parametric model yielding logistic discriminant functions and involves a convex functional.
2. Accordingly, fitting model parameters to given training data amounts to a convex optimization problem, and one may use methods such as gradient descent to tune the parameters.
3. Logistic regression in some sense contains Linear Discriminant Analysis as a special case, although both make different assumptions on the data!.
4. The Laplacian measures the infinitesimal deviation of a function from its local average.
5. The heat equation instantaneously smooths functions, whether in the whole space, or in a compact subset.