Geomstats: an overview

Luís F. Pereira

luis.gomes-pereira@inria.fr

SIAM CSE 2023, February 27

Geomstats

1. Geometry: Instantiate a manifold

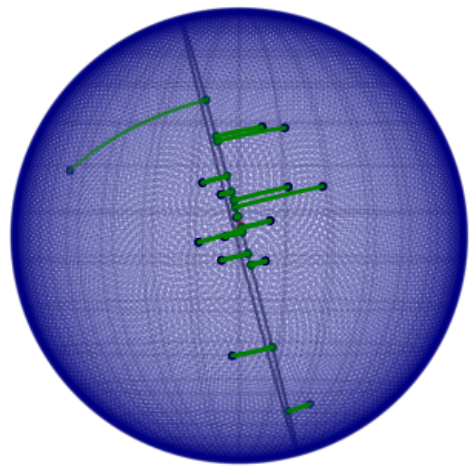
```
from geomstats.geometry.hypersphere import Hypersphere
sphere = Hypersphere(dim=2)
```

2. Statistics/Learning: Run estimation or learning

```
from geomstats.learning.frechet_mean import FrechetMean
estimator = FrechetMean(space)
estimator.fit(points)
```

Statistics on manifolds

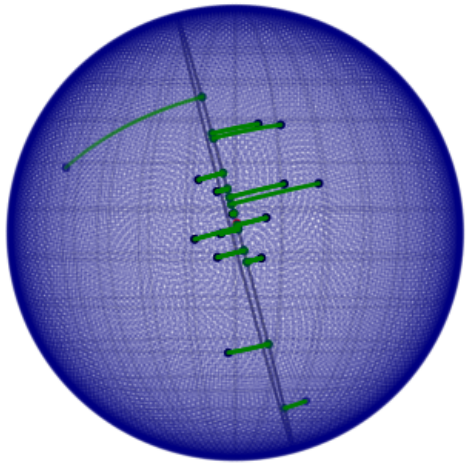
```
from geomstats.geometry.hypersphere \
    import Hypersphere
from geomstats.learning.pca \
    import TangentPCA

space = Hypersphere(dim=2, equip=True)

estimator = TangentPCA(space)

# X.shape: (n_samples, dim +1)
estimator.fit(X)

v_ = estimator.components_[0]
```



PCA on manifolds

```
estimator = TangentPCA(space)
estimator.fit(X)
```

$$v = \underset{v}{\operatorname{argmax}} \frac{1}{N} \sum_{j=1}^{N} d\left(\mu, \pi_{\gamma_v}\left(x_j\right)\right)^2$$

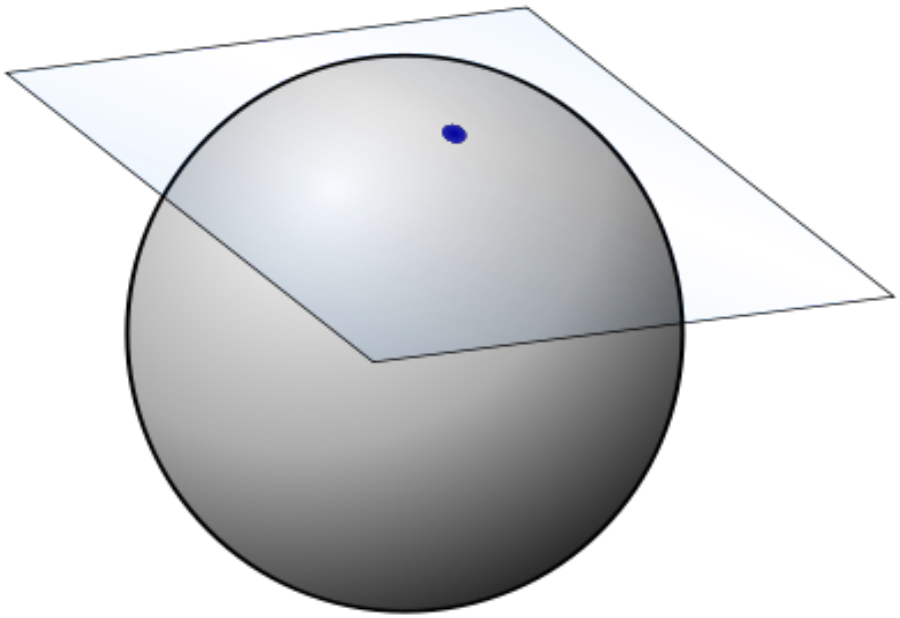


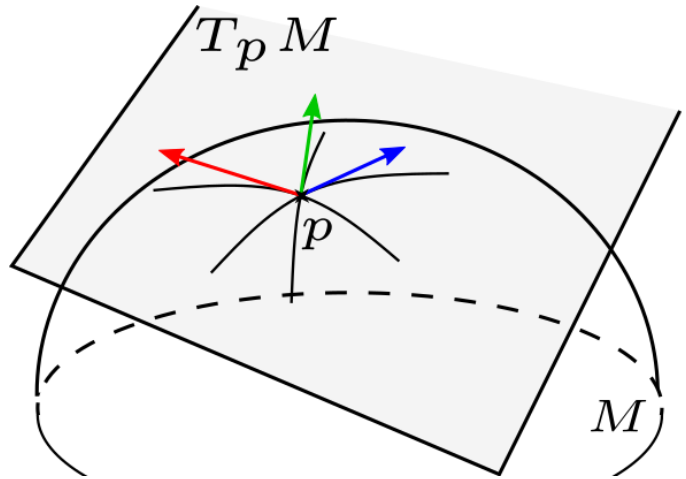
In practice:

1. Move points to tangent space at Fréchet mean
2. Perform standard PCA

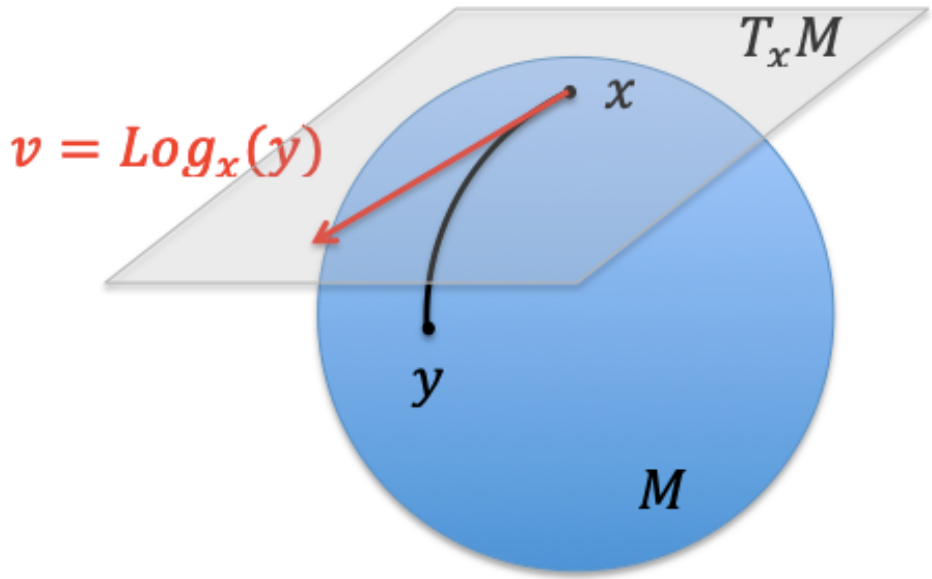
Tangent space

- Vector space associated to a point in a manifold



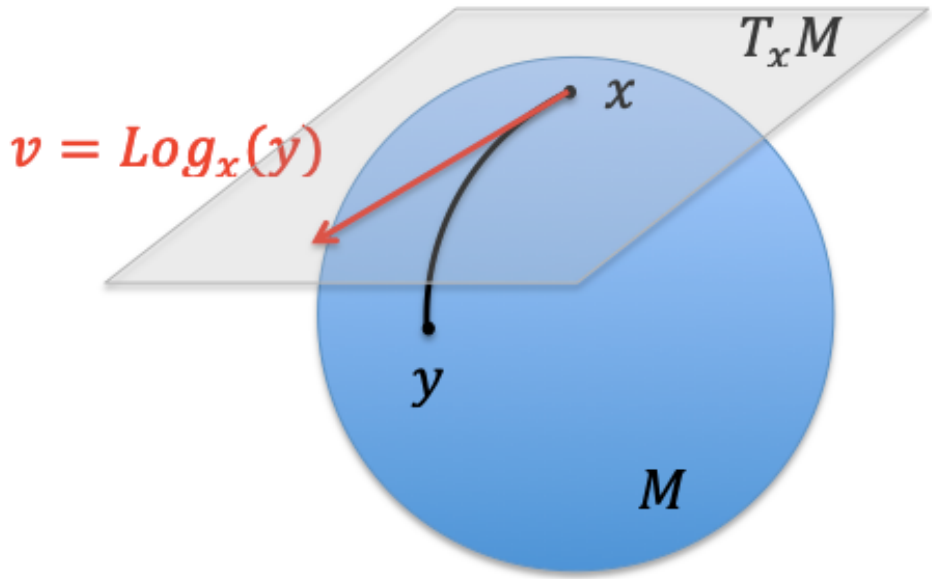


Log, Exp, and geodesics



$\mathrm{Log}_x : M \to T_xM$ $\qquad$ $v = \mathrm{Log}_x(y)$

Log, Exp, and geodesics



$$\mathrm{Log}_x : M \to T_xM \qquad v = \mathrm{Log}_x(y)$$

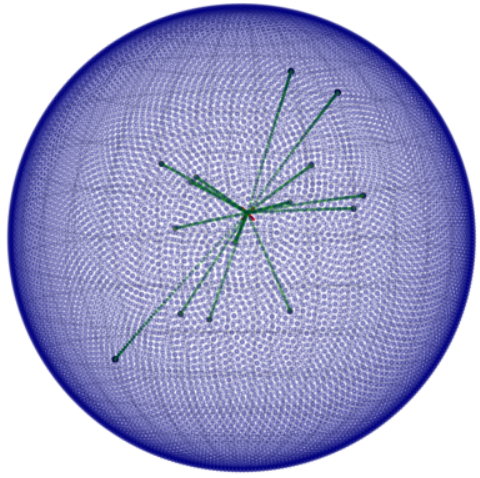
$$\mathrm{Exp}_x : T_xM \to M \qquad y = \mathrm{Exp}_x(v)$$

$$\ddot{x}_t^k = -\sum_{i,j}^{d} \Gamma_{ij}^k(x_t)\, \dot{x}_t^i \dot{x}_t^j, k = 1, \ldots, d$$

Fréchet mean

```
estimator = FrechetMean(space)
estimator.fit(X)

mean = estimator.estimate_
```

$$\bar{x} = \underset{x \in M}{\operatorname{argmin}} \sum_{i=1}^{n} d\left(x_i, x\right)^2$$



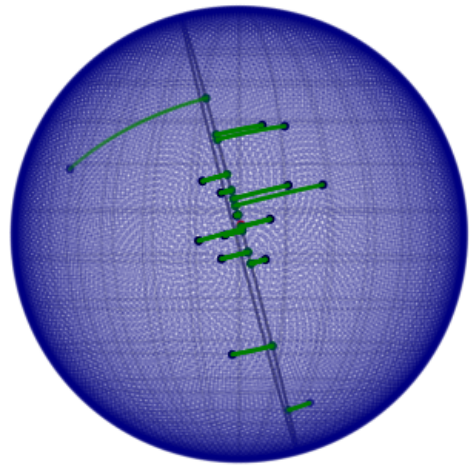
An algorithm:

```
while condition:
    grad = -gs.sum(space.metric.log(X, mean)) / n_samples
    mean = space.metric.exp(step_size*grad, mean)
```

A quick recap

1. Move points to tangent space at Fréchet mean
2. Perform standard PCA

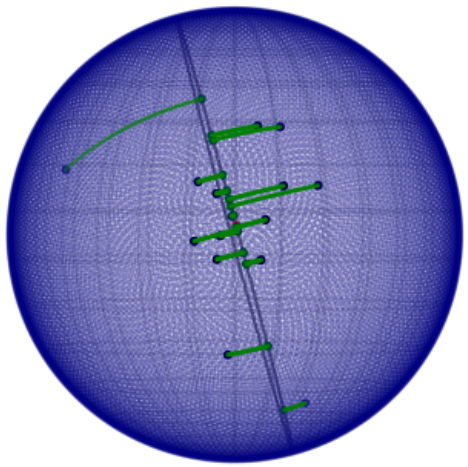
```
from sklearn.decomposition import PCA

estimator = PCA()
estimator.fit(logs)
```



Projection onto a geodesic

$$\pi_{\gamma_v}(x_j) = \underset{t \in \mathbb{R}}{\operatorname{argmin}}\, d\left(x_j, \operatorname{Exp}_\mu(tv)\right)^2$$

```
geod = space.metric.geodesic(
    mean,
    initial_tangent_vec=tangent_vec
)

point_ = geod(t)
space.metric.squared_dist(point_, point)
```



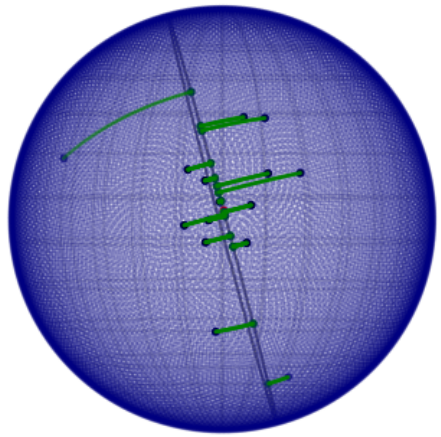
Inner product and distance

$$g_x(\cdot, \cdot) \text{ on } T_xM$$

$$d(x, y)^2 = g_x(v, v), \text{where } v = \mathrm{Log}_x(y)$$

```
v = space.metric.log(y, x)
squared_dist = space.metric.inner_product(v, v, x)

# or
squared_dist = space.metric.squared_dist(x, y)
```

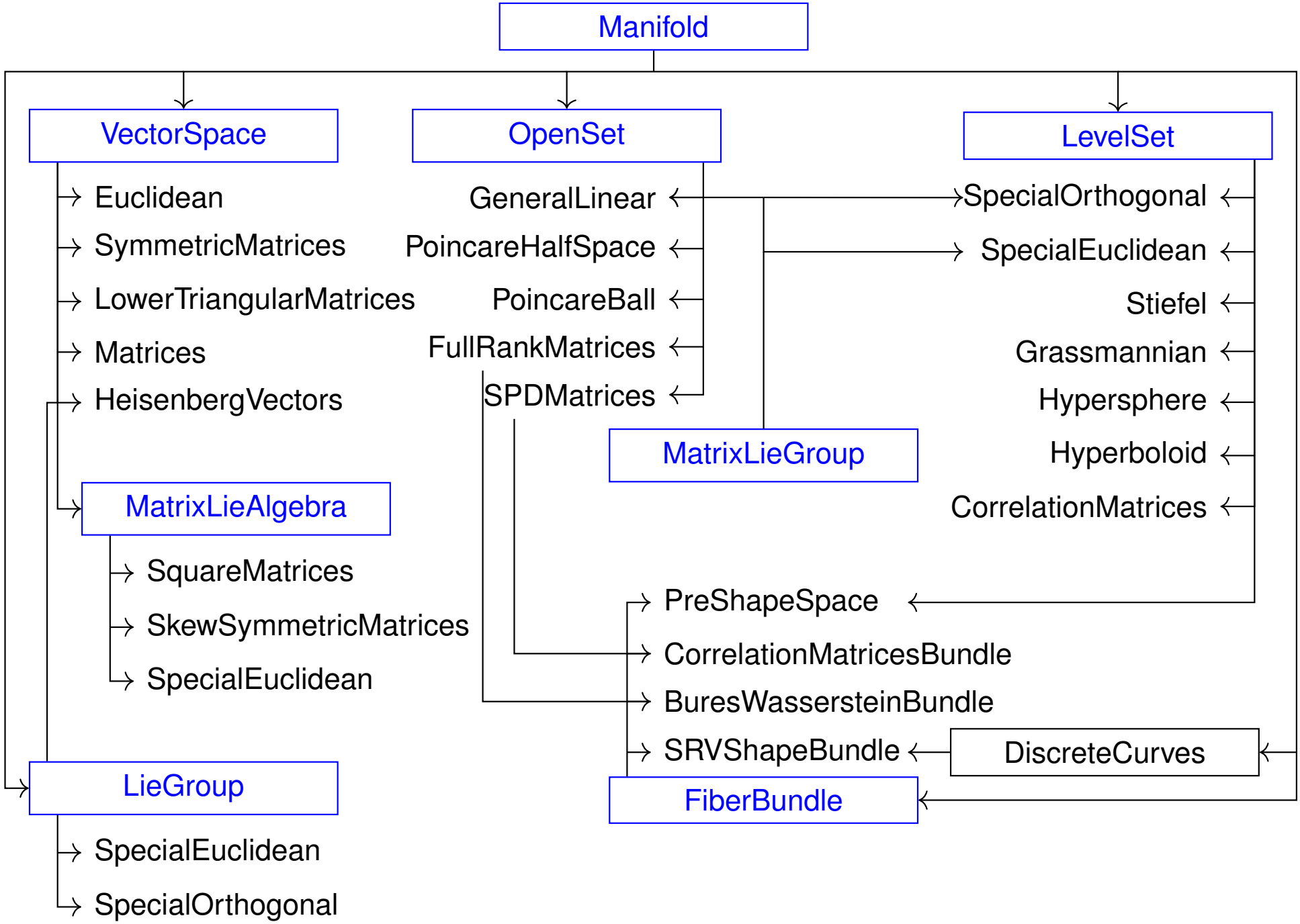
Geometry



(M, g)

---

space = Hypersphere(dim=2, equip=True)


Manifold
VectorSpace
OpenSet
LevelSet
Euclidean
SymmetricMatrices
LowerTriangularMatrices
Matrices
HeisenbergVectors
GeneralLinear
PoincareHalfSpace
PoincareBall
FullRankMatrices
SPDMatrices
SpecialOrthogonal
SpecialEuclidean
Stiefel
Grassmannian
Hypersphere
Hyperboloid
CorrelationMatrices
MatrixLieGroup
MatrixLieAlgebra
SquareMatrices
SkewSymmetricMatrices
SpecialEuclidean
PreShapeSpace
CorrelationMatricesBundle
BuresWassersteinBundle
SRVShapeBundle
DiscreteCurves
FiberBundle
LieGroup
SpecialEuclidean
SpecialOrthogonal

Level Set: Hypersphere

$$S^d = \left\{ x \in \mathbb{R}^{d+1} \mid \|x\|^2 = 1 \right\}$$

```
from geomstats.geometry.hypersphere import Hypersphere

manifold = Hypersphere(dim=2)

point = manifold.random_point()
print(manifold.belongs(point))  # True

vec = manifold.embedding_space.random_point()
tangent_vec = manifold.to_tangent(vec, point)
print(manifold.is_tangent(tangent_vec, point))  # True
```

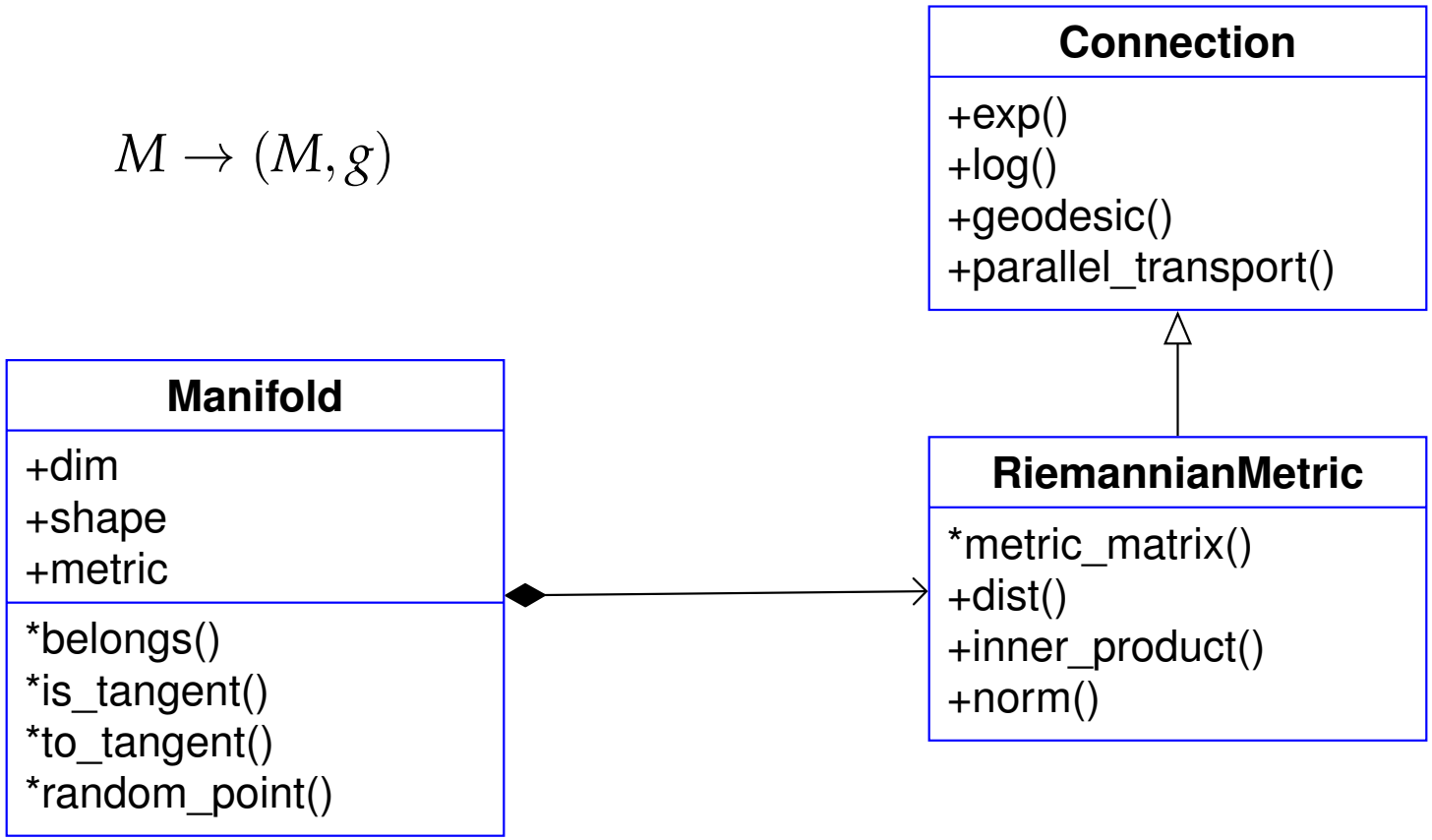
Open Set: SPD Matrices

$$\mathrm{SPD}(n) = \left\{ \Sigma \in \mathrm{Sym}(n) \mid \forall x \neq 0, x^\top \Sigma x > 0 \right\}$$

```
from geomstats.geometry.spd_matrices import SPDMatrices

manifold = SPDMatrices(n=2)

point = manifold.random_point()
print(manifold.belongs(point))  # True

vec = manifold.embedding_space.random_point()
tangent_vec = manifold.to_tangent(vec, point)
print(manifold.is_tangent(tangent_vec, point))  # True
```

Connection

$$M \to (M, g)$$



```
my_manifold = MyManifold(*args, **kwargs, equip=False)
my_manifold.equip_with_metric(MyRiemannianMetric)
```

Connection: Hypersphere

$$\langle v, w \rangle_x = \sum_{i=1}^{d} v_i w_i = v^\top w$$

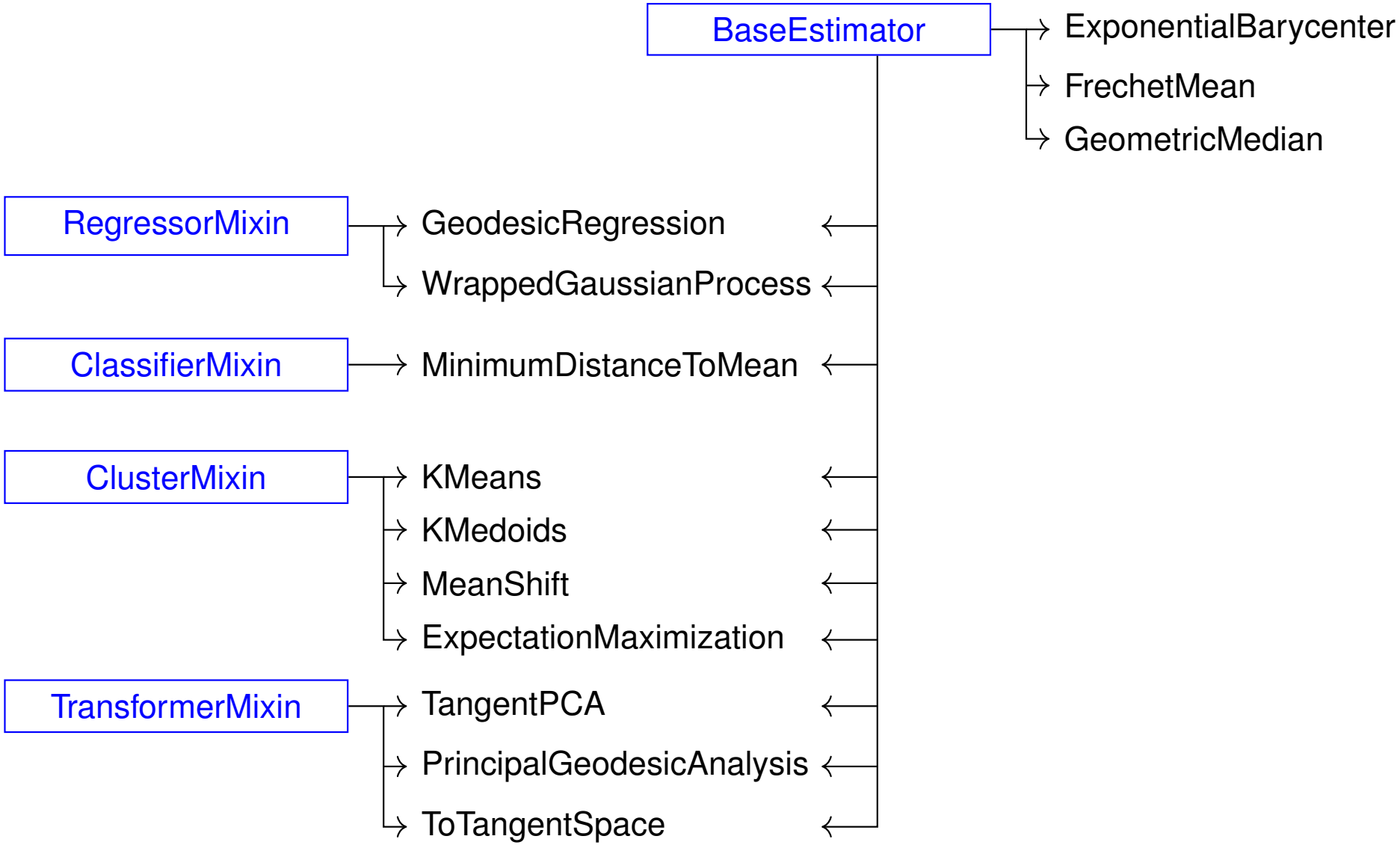
```
manifold = Hypersphere(dim=2, equip=False)
manifold.equip_with_metric(HypersphereMetric)

base_point = manifold.random_point()
vecs = manifold.embedding_space.random_point(2)

vec_1, vec_2 = manifold.to_tangent(vecs, base_point)
print(manifold.metric.inner_product(vec_1, vec_2, base_point))

point = manifold.metric.exp(vec_1, base_point)
vec_1_ = manifold.metric.log(point, base_point)
print(gs.allclose(vec_1, vec_1_))  # True

norm = manifold.metric.norm(vec_1, base_point)
dist = manifold.metric.dist(point, base_point)
print(gs.isclose(dist, norm))  # True
```

Connection: SPD Matrices

$$g_\Sigma(V, W) = \mathrm{tr}\left(\Sigma^{-1} V \Sigma^{-1} W\right)$$

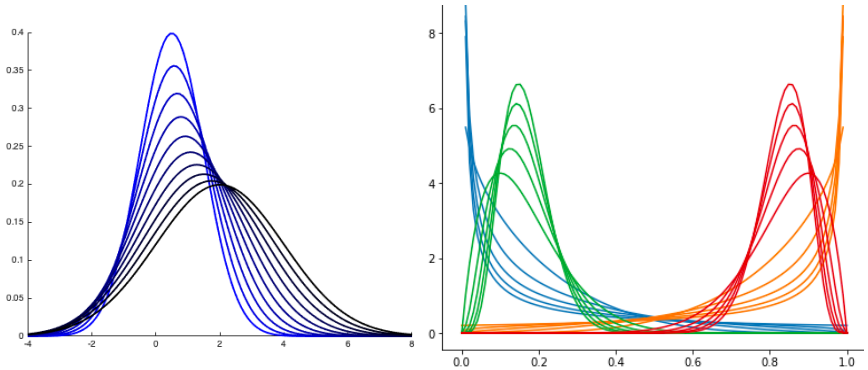
```
manifold = SPDMatrices(n=3, equip=False)
manifold.equip_with_metric(SPDAffineMetric)

base_point = manifold.random_point()
vecs = manifold.embedding_space.random_point(2)

vec_1, vec_2 = manifold.to_tangent(vecs, base_point)
print(manifold.metric.inner_product(vec_1, vec_2, base_point))

point = manifold.metric.exp(vec_1, base_point)
vec_1_ = manifold.metric.log(point, base_point)
print(gs.allclose(vec_1, vec_1_))  # True

norm = manifold.metric.norm(vec_1, base_point)
dist = manifold.metric.dist(point, base_point)
print(gs.isclose(dist, norm))  # True
```

Learning


BaseEstimator
ExponentialBarycenter
FrechetMean
GeometricMedian
RegressorMixin
GeodesicRegression
WrappedGaussianProcess
ClassifierMixin
MinimumDistanceToMean
ClusterMixin
KMeans
KMedoids
MeanShift
ExpectationMaximization
TransformerMixin
TangentPCA
PrincipalGeodesicAnalysis
ToTangentSpace

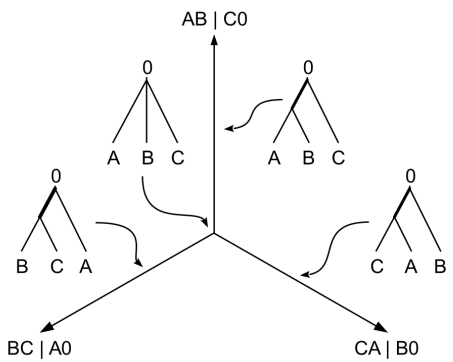
Other modules

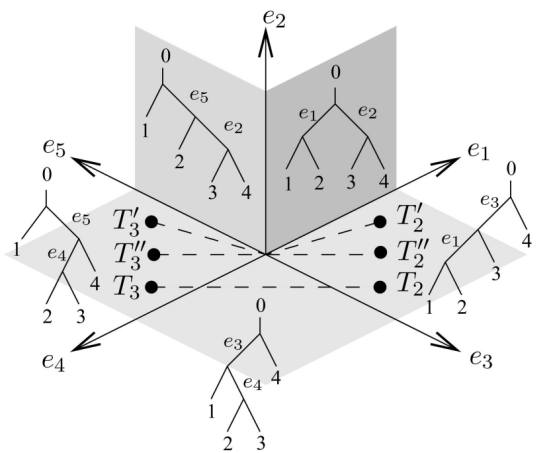
- Information geometry: learning on parametric probability distributions
  - `InformationManifold`
  - `FisherRaoMetric`



- Complex manifold (with Yann Cabanes)
  - Duplicate manifold structure to handle complex fields
  - `ComplexMatrices`, `HermitianMatrices`, `HPDMatrices`, `Siegel`

- Stratified spaces (with Anna Calissano and Jonas Lueg)
  - `Spider`
  - `GraphSpace`
  - `BHVSpace`, `WaldSpace`



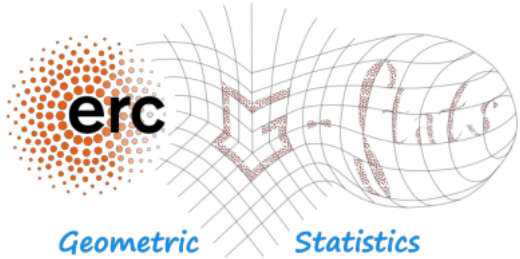


Backend

```
gs.sum(space.metric.log(X, mean))
```

```
import geomstats.backend as gs
```

	numpy	autograd	pytorch
numerical precision		float64	
gpu			✓
automatic differentiation		✓	✓



Thank you for your attention!

https://geomstats.ai/
https://github.com/geomstats/geomstats

Nicolas Guigui, Nina Miolane, Xavier Pennec. **Introduction to Riemannian Geometry and Geometric Statistics: from basic theory to implementation with Geomstats**. Foundations and Trends in Machine Learning.