Using machine learning methods in geometric modeling

Georg Umlauf

Institute for Optical Systems

Faculty of Computer Science

University of Applied Science Constance, Germany

Joint work with Pascal Laube, Merlin Blume, Manuel Caputo, Matthias Franz

Marseille, 3 November 2016

INSTITUTE FOR OPTICAL SYSTEMS

Contents

- **Part 1: Learning knot placements** for b-spline curve approximation and t-spline surface skinning
  - Using support vector machines (SVMs) to learn good knot vectors for spline curve/surface approximation.

Contents

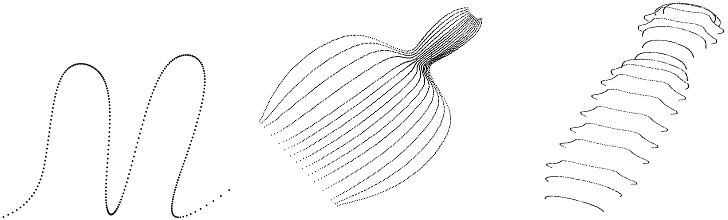
- **Part 1: Learning knot placements** for b-spline curve approximation and t-spline surface skinning
  - Using support vector machines (SVMs) to learn good knot vectors for spline curve/surface approximation.
- **Part 2: Learning surface primitive classification** from point clouds
  - Using SVMs to classify point clouds to geometric primitive classes.
  - Using stacked auto-encoders (SAEs) to learn geometric features used for classification.

Part 1: Learning knot placements for b-spline curve and t-spline surface approximation

2d-input Given a sequence of points $\mathbf{p}_i$ for curve approximation.

3d-input Given an array of sequences of points for surface skinning (lofting).

Output: Good knot vector(s) for spline approximation.



IOS INSTITUTE FOR OPTICAL SYSTEMS

Curve approximation

Curves: Compute the control points c_j of a b-spline curve of degree l

$$C(t) = \sum_{j=0}^{J} c_j \cdot N_j^l(t)$$

minimizing

$$\sum_{i=0}^{m} \|\mathbf{p}_i - C(t_i)\|^2.$$

Curve approximation

Curves: Compute the control points c_j of a b-spline curve of degree l

$$C(t) = \sum_{j=0}^{J} c_j \cdot N_j^l(t)$$

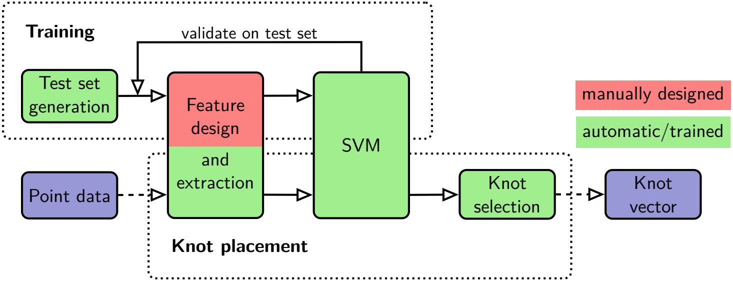
minimizing

$$\sum_{i=0}^{m} \|\mathbf{p}_i - C(t_i)\|^2.$$

Skinning: Later...

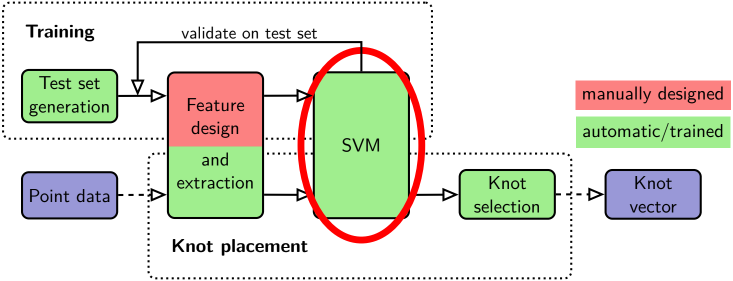
Approach

- Design geometric point features for knot placement.
- Train a support vector machine (SVM) to learn the quality of a candidate knot.
  - For validation use approximation of test data.
- Use the trained SVM to generate knot vectors for new approximation tasks.



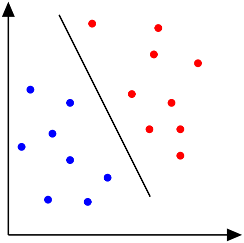
Approach

- Design geometric point features for knot placement.
- Train a support vector machine (SVM) to learn the quality of a candidate knot.
  - For validation use approximation of test data.
- Use the trained SVM to generate knot vectors for new approximation tasks.



Support vector machines (SVMs) (1)

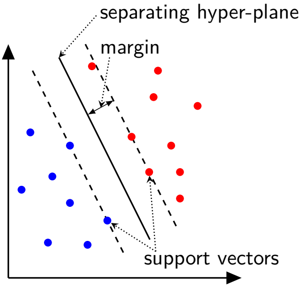
1. Map data points to feature vectors $\mathbf{x}_i$.
   - For training label the data with correct class y_i.



Support vector machines (SVMs) (1)

1. Map data points to feature vectors $\mathbf{x}_i$.
   - For training label the data with correct class y_i.
2. For a linear decision function $\mathrm{sign}(\omega^t \mathbf{x} + b)$, maximize the margin between the classes by minimizing

$$\omega^t \omega / 2 \qquad \text{s.t.} \qquad y_i(\omega^t \mathbf{x}_i + b) \geq 1.$$



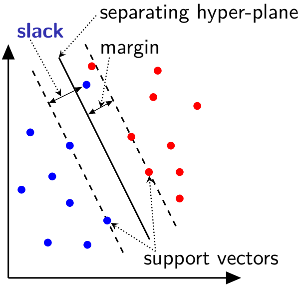
Support vector machines (SVMs) (1)

1. Map data points to feature vectors $\mathbf{x}_i$.
   - For training label the data with correct class y_i.
2. For a linear decision function $\operatorname{sign}(\omega^t \mathbf{x} + b)$, maximize the margin between the classes by minimizing

$$\omega^t \omega / 2 \qquad \text{s.t.} \qquad y_i(\omega^t \mathbf{x}_i + b) \geq 1.$$

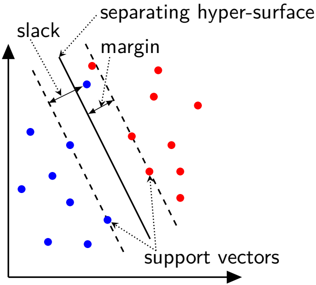
3. For noisy data add **slack** ξ_i, i.e. minimize

$$\omega^t \omega / 2 + C \sum \xi_i \qquad \text{s.t.} \qquad y_i(\omega^t \mathbf{x}_i + b) \geq 1 - \xi_i.$$



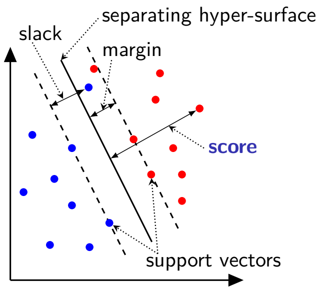
Support vector machines (SVMs) (2)

4. For non-linearly-separable data, replace the scalar product with a kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, i.e. use the decision function
$$\operatorname{sign}\left(\sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right).$$
   - This is solved using Lagrange multipliers and dualization with $\omega = \sum \alpha_i y_i \mathbf{x}_i$.



Support vector machines (SVMs) (2)

4. For non-linearly-separable data, replace the scalar product with a kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, i.e. use the decision function
$$\text{sign}\left(\sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right).$$
   - This is solved using Lagrange multipliers and dualization with $\omega = \sum \alpha_i y_i \mathbf{x}_i$.
5. The **score** measures the signed distance to the separating hyper-surface.

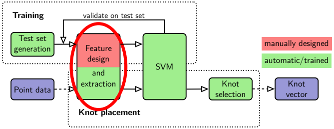


Geometric features for curve knot placement (1)

What are good **geometric features for knot placement?**

See e.g. [Park/Lee2007; Piegl/Tiller2000+2012; Razdan; Yuan/Chen/Zhou2013; etc.].

- Point distances.

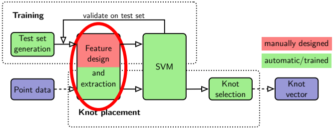


Geometric features for curve knot placement (1)

What are good **geometric features for knot placement?**

See e.g. [Park/Lee2007; Piegl/Tiller2000+2012; Razdan; Yuan/Chen/Zhou2013; etc.].

- Point distances.
- Features derived from discrete curvature estimates κ_i.



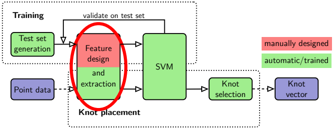
Geometric features for curve knot placement (1)

What are good **geometric features for knot placement?**

See e.g. [Park/Lee2007; Piegl/Tiller2000+2012; Razdan; Yuan/Chen/Zhou2013; etc.].

- Point distances.
- Features derived from discrete curvature estimates κ_i.
- **Local curvature maxima** (LCM) are points p_i with

$$\kappa_i > \max(\kappa_{i-1}, \kappa_{i+1}) \quad \text{and} \quad \kappa_i \geq \kappa_{\text{average}}.$$



Geometric features for curve knot placement (1)

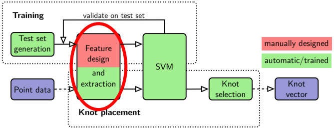
What are good **geometric features for knot placement?**

See e.g. [Park/Lee2007; Piegl/Tiller2000+2012; Razdan; Yuan/Chen/Zhou2013; etc.].

- Point distances.
- Features derived from discrete curvature estimates κ_i.
- **Local curvature maxima** (LCM) are points p_i with

$$\kappa_i > \max(\kappa_{i-1}, \kappa_{i+1}) \quad \text{and} \quad \kappa_i \geq \kappa_{\text{average}}.$$

- Features derived from the LCM
  - Angles to closest LCM.
  - Euclidean, arc-length, or parametric point distances to the closest LCM.

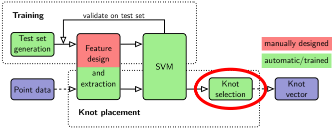


INSTITUTE FOR OPTICAL SYSTEMS

Geometric features for curve knot placement (2)

Subsequent **knot selection**

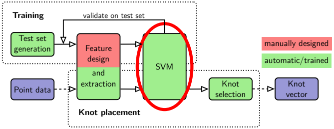
- Use salient features as knots and, in regions without salient features, sample uniformly,
- ... or use averages of uniform parameter averages...



Score-based knot placement

How well do points with **salient geometric features perform for approximation?**

- **Score maxima** of a trained SVM.
  - Choose peaks by prominence: Highest value you have to at least descend to to reach a higher peak.



INSTITUTE FOR OPTICAL SYSTEMS

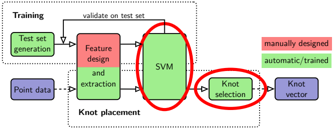
Score-based knot placement

How well do points with **salient geometric features perform for approximation?**

- **Score maxima** of a trained SVM.
  - Choose peaks by prominence: Highest value you have to at least descend to to reach a higher peak.

Subsequent **knot selection**

- For knot insertion, find closest score peak to middle point of knot segment with maximum score-sum. .
- In regions without score peaks, sample uniformly or score-weighted.

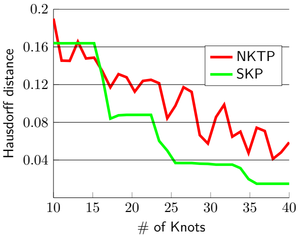


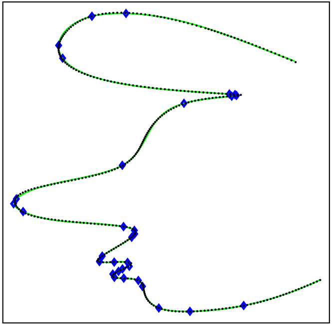
INSTITUTE FOR OPTICAL SYSTEMS

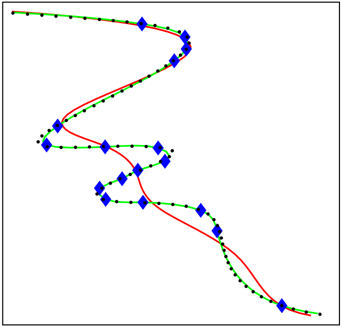
Results for SVM curves knot placement

- Score from high score (red) to low score (blue)

Results for b-spline curve approximation







Surface skinning

Approximation and approach analog to curves.

Curves: ... done.

Skinning: Compute the control points $c_{i,j}$ of a t-spline surface of degree (k, l)

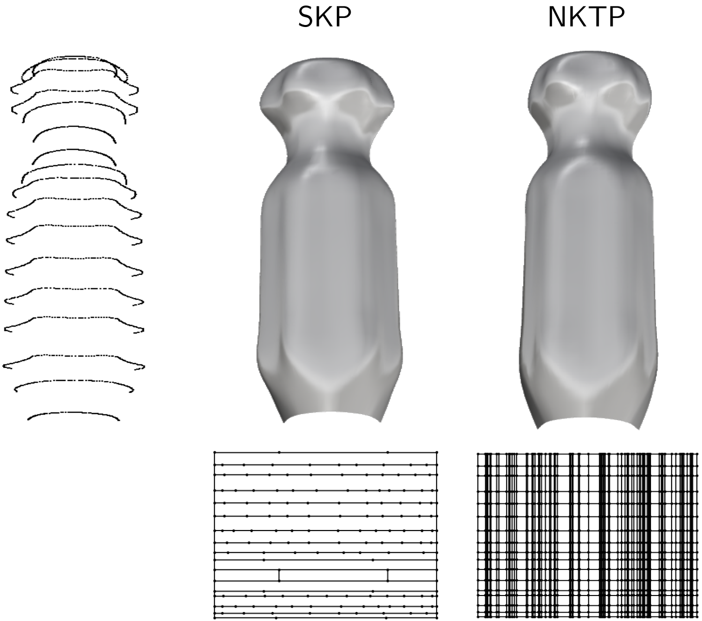
$$S(u, v) = \sum_{j=0}^{J} \sum_{i=0}^{I_j} c_{i,j} \cdot N_{i,j}^k(u) \cdot N_j^l(v)$$

minimizing

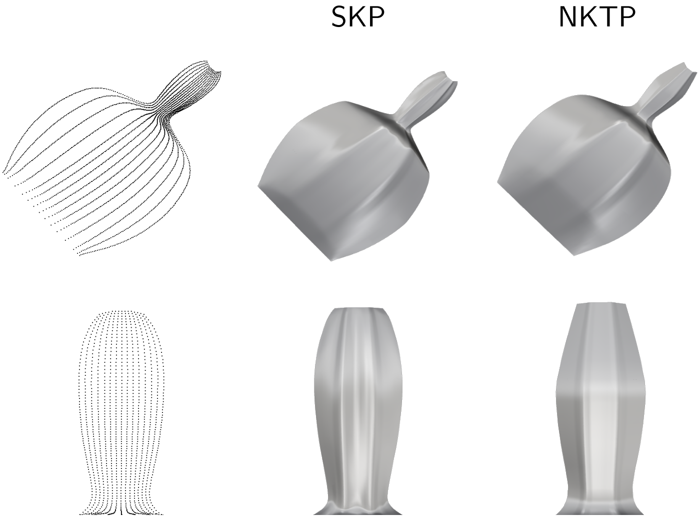
$$\sum_{j=0}^{m} \sum_{i=0}^{n_j} \|\mathbf{p}_{i,j} - S(u_i, v_j)\|^2$$

where each $N_{i,j}^k$ has its own knot vector $\mathbf{u}^{(j)}$.

Results for t-spline skinning (1)

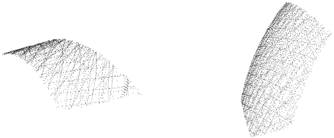


Results for t-spline skinning (2)

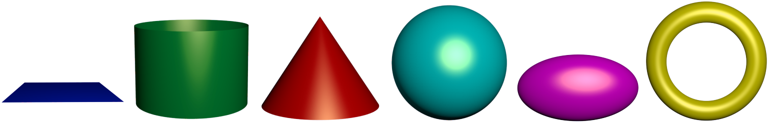


Part 2: Learning surface primitive classification

Input: Point clouds sampled from a patch of a primitive surface.



Output: Classification to the correct primitive class.

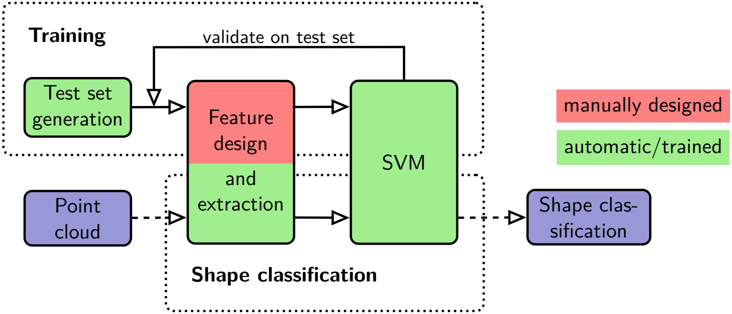


Planes Cylinders Cones Spheres Ellipsoids Tori

SVM primitive classification (1)

General approach analog to knot-placement.

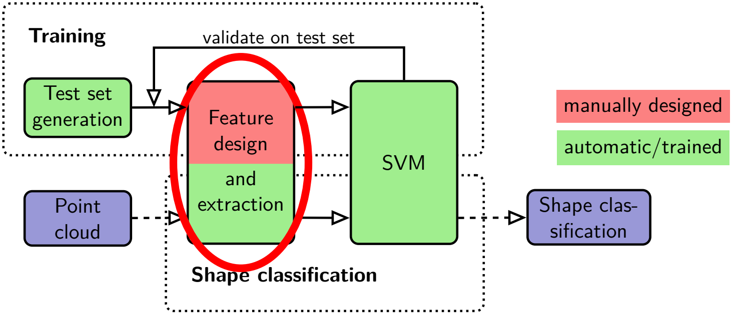
- Design geometric point cloud features for primitive classification.
- Train a support vector machine (SVM) to learn the corresponding primitive class.
  - For validation use pre-labeled of test data.
- Use the trained SVM to classify new point clouds.



SVM primitive classification (1)

General approach analog to knot-placement.

- Design geometric point cloud features for primitive classification.
- Train a support vector machine (SVM) to learn the corresponding primitive class.
  - For validation use pre-labeled of test data.
- Use the trained SVM to classify new point clouds.



SVM primitive classification (2)

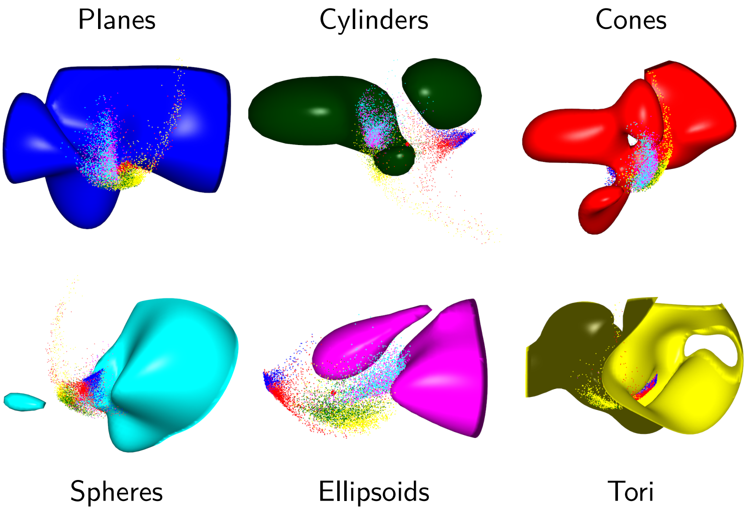
Point clouds are mapped to feature vectors.

See e.g. [Koenderink/van Doorn1992; Osada et al. 2002; Wahl et al. 2003; etc.].

- Point relation features:
  - Point distances, point angles, triangle areas, etc.
- Normal based features:
  - Normal directions, normal angles, etc.
- Curvature based features:
  - Curvature directions, curvature angles, shape index, etc.
- Hybrid features:
  - Surflet pairs, etc.

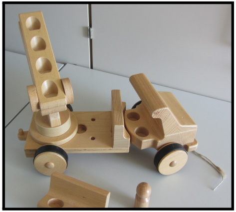
Results of SVM primitive classification (1)

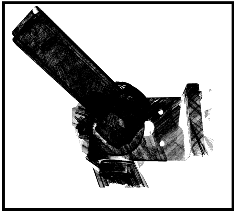
- Separating hyper-surfaces projected to 3d.

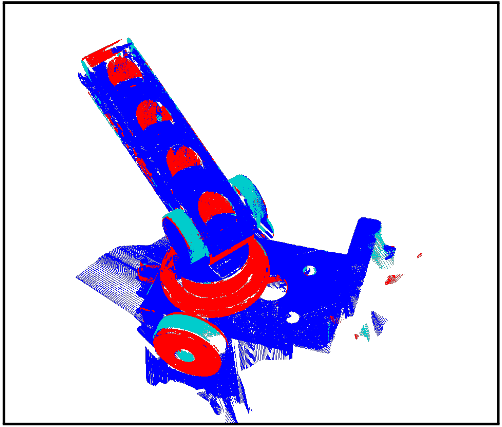


Results of SVM primitive classification (2)

- Classification of a manual 3d-scan.



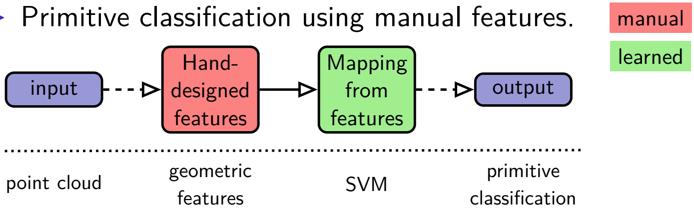




planes, cones, spheres.

Learning geometric features using SAEs

- Primitive classification using manual features.

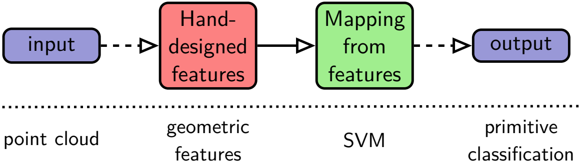


Learning geometric features using SAEs

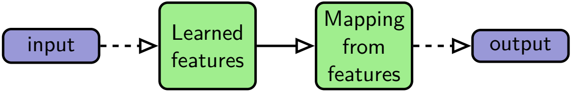
- Primitive classification using manual features.

manual

learned



- Learning geometric features for classification.

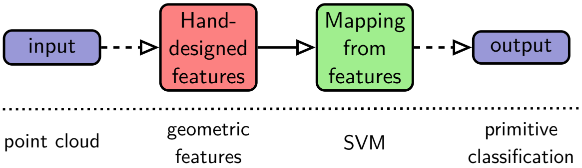


Learning geometric features using SAEs

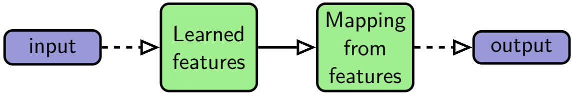
- Primitive classification using manual features.

manual

learned



- Learning geometric features for classification.



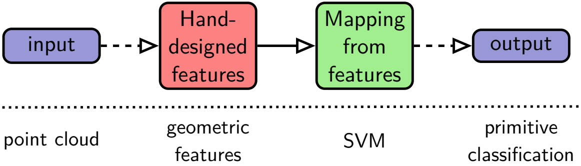
- Stacked auto-encoders (SAEs).

Learning geometric features using SAEs

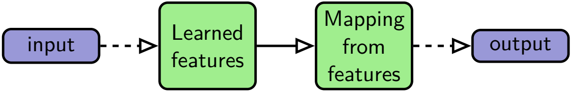
- Primitive classification using manual features.

manual

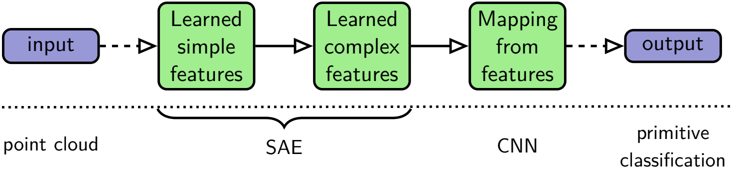
learned



- Learning geometric features for classification.



- Stacked auto-encoders (SAEs).



Feature engineering using SAEs

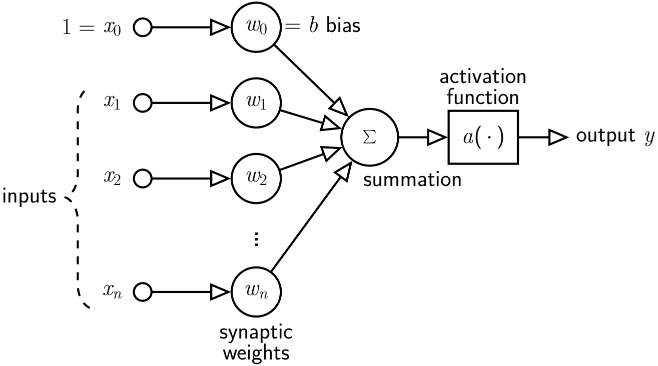
- A stacked auto-encoder (SAE) is
  - a multi-layer perceptron
  - for un-supervised learning
  - of representations.

Feature engineering using SAEs

- A stacked auto-encoder (SAE) is
  - a $\underbrace{\text{multi-layer}}_{2}$ $\underbrace{\text{perceptron}}_{1}$
  - for $\underbrace{\text{un-supervised learning}}_{3}$
  - of $\underbrace{\text{representations}}_{3}$.

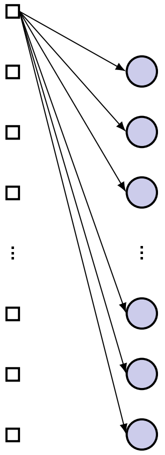
① Perceptron

- A **perceptron** implements for the inputs $x_1, \ldots, x_n$ the function
$$y = a\left(b + \sum_{i=1}^{n} w_i \cdot x_i\right).$$



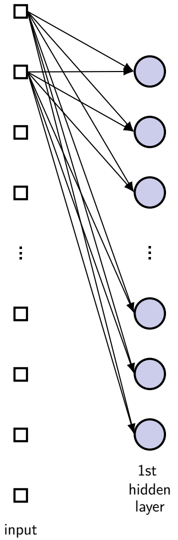
② Deep neural nets (DNNs)

- **Deep neural nets**: multiple layers of perceptrons.



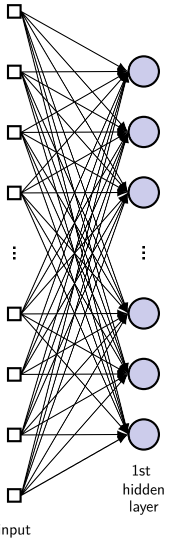
② Deep neural nets (DNNs)

- **Deep neural nets**: multiple layers of perceptrons.



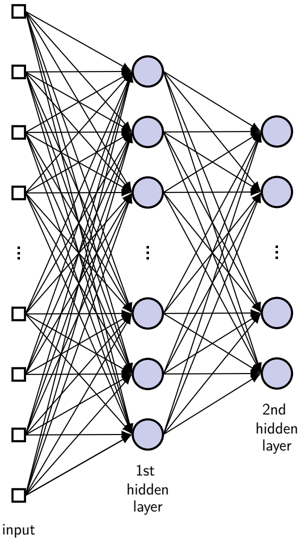
② Deep neural nets (DNNs)

- **Deep neural nets**: multiple layers of perceptrons.



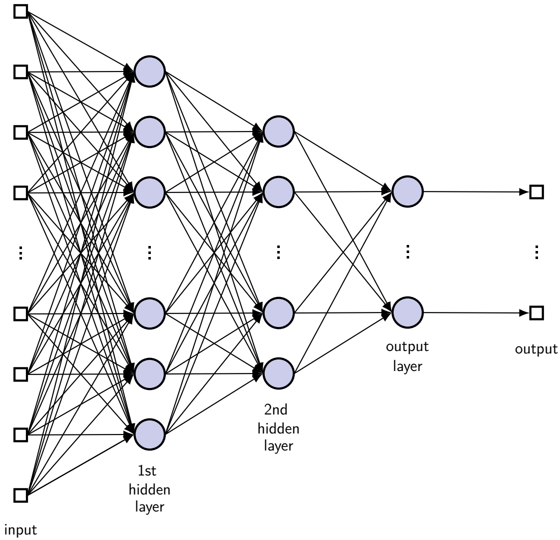
② Deep neural nets (DNNs)

- **Deep neural nets**: multiple layers of perceptrons.



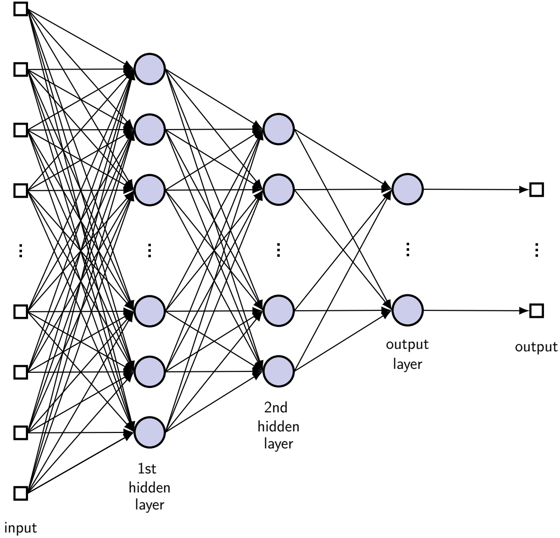
② Deep neural nets (DNNs)

- **Deep neural nets**: multiple layers of perceptrons.



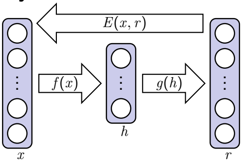
② Deep neural nets (DNNs)

- **Deep neural nets**: multiple layers of perceptrons.
- Training via back-propagation and gradient descent.



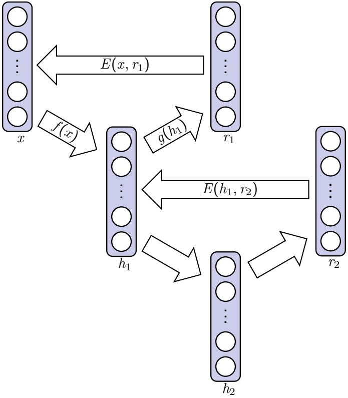
③ Stacked auto-encoders (SAEs)

- A auto-encoder minimizes an error $E(x,r)$ on each layer.



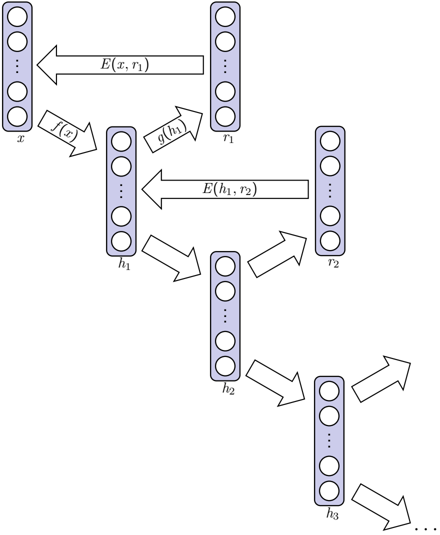
③ Stacked auto-encoders (SAEs)

- A **stacked auto-encoder** minimizes an error $E(x, r)$ on each layer.



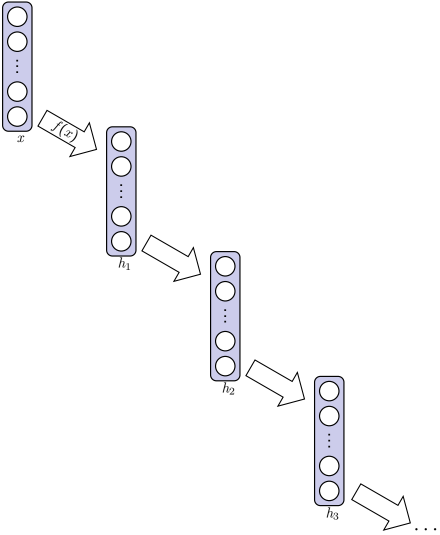
③ Stacked auto-encoders (SAEs)

- A **stacked auto-encoder** minimizes an error $E(x, r)$ on each layer.



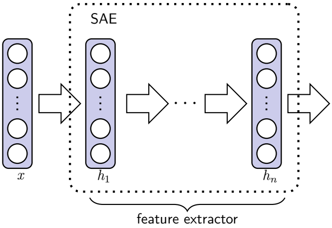
③ Stacked auto-encoders (SAEs)

- A **stacked auto-encoder** minimizes an error $E(x, r)$ on each layer.



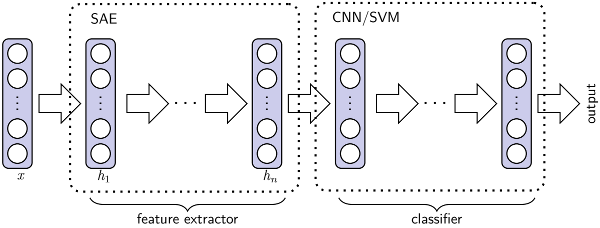
③ Stacked auto-encoders (SAEs)

- A **stacked auto-encoder** minimizes an error $E(x, r)$ on each layer.
- The trained SAE can be used as feature extractor.



③ Stacked auto-encoders (SAEs)

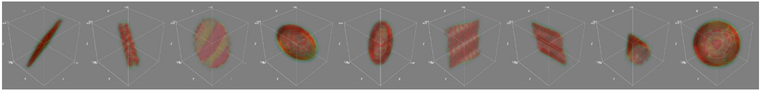
- A **stacked auto-encoder** minimizes an error $E(x, r)$ on each layer.
- The trained SAE can be used as feature extractor.
- The features learned by the SAE are used to train a second DNN to classify the data.



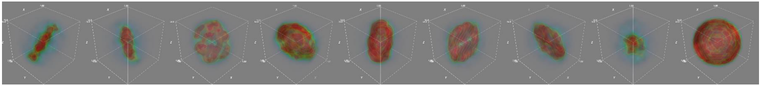
Learned geometric features (1)

- The auto-encoder is capable of representing geometric primitives.

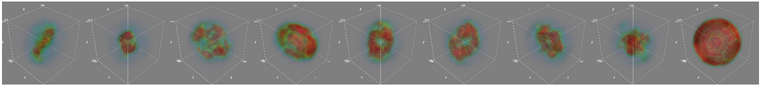
original



DAE, 4100 neurons



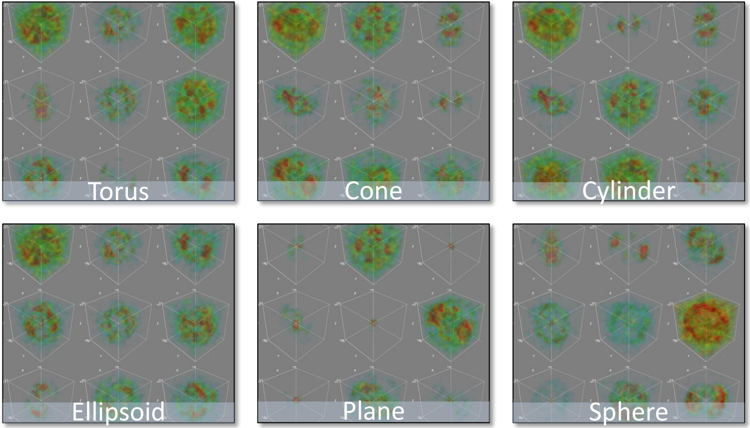
SDAE, 3 layers, 4100 neurons



Learned geometric features (2)

- How can we interpret the un-supervised learned geometric features?

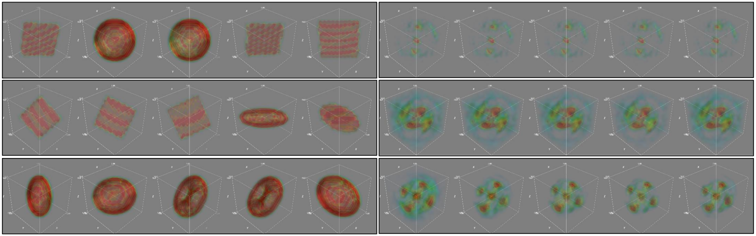
SDAE 3x4100 (Gaussian noise = 0.5)



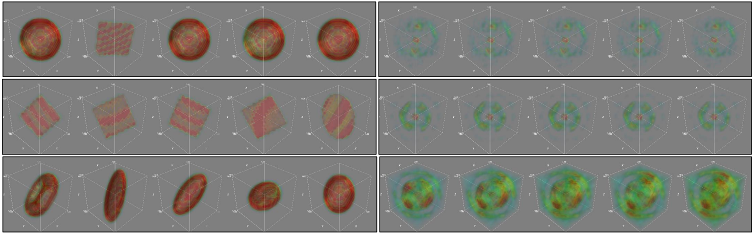
Learned geometric features (2)

- Saliency maps.

SAE 3x4100 (no noise)



SDAE 3x4100 (Gaussian noise = 0.5)



Thank you!

Questions?