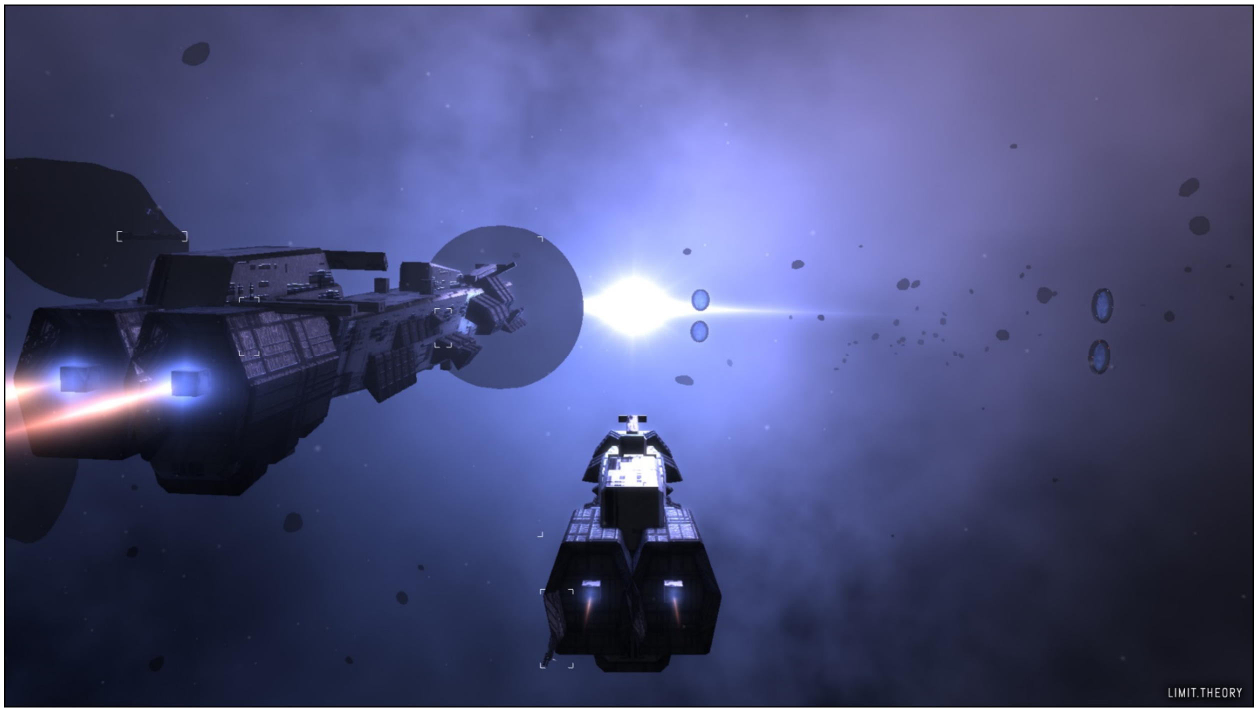
Machine Learning Techniques for Geometric Modeling

Evangelos Kalogerakis

UMASS AMHERST

3D models for digital entertainment



Limit Theory

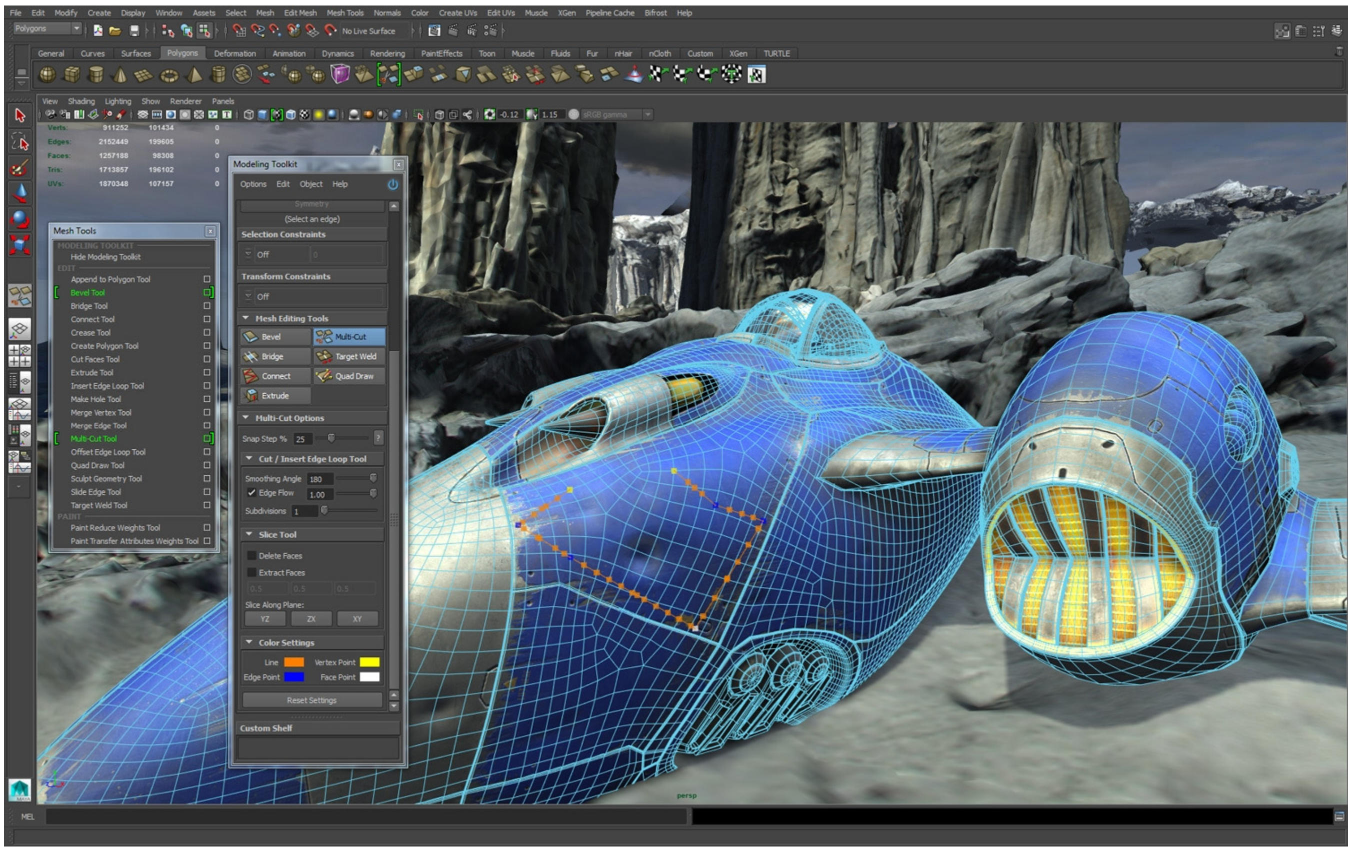
3D models for printing

MakerBot Industries

3D models for architecture

Architect: Thomas Eriksson
Courtesy Industriromantik

Geometric modeling is not easy!



Autodesk Maya 2015

“Traditional” Geometric Modeling

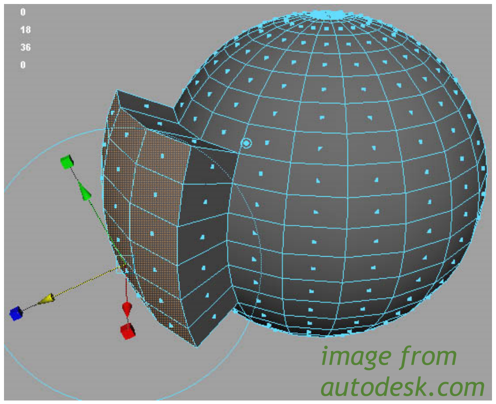


image from autodesk.com

Manipulating polygons

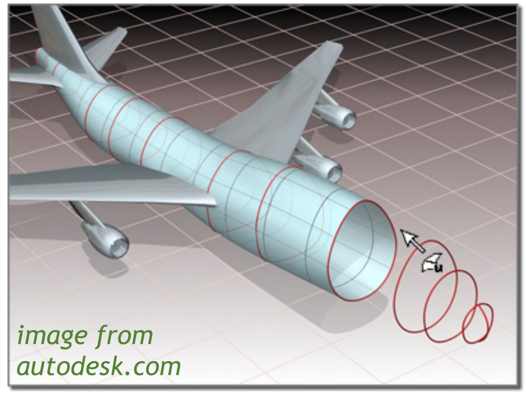


image from autodesk.com

Manipulating curves

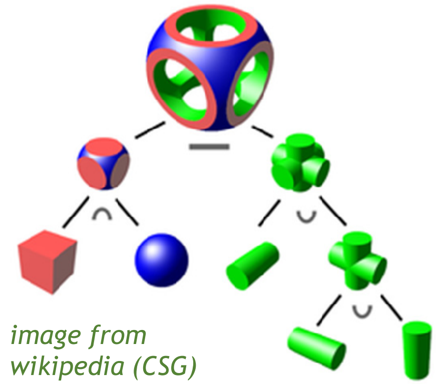


image from wikipedia (CSG)

Manipulating 3D primitives

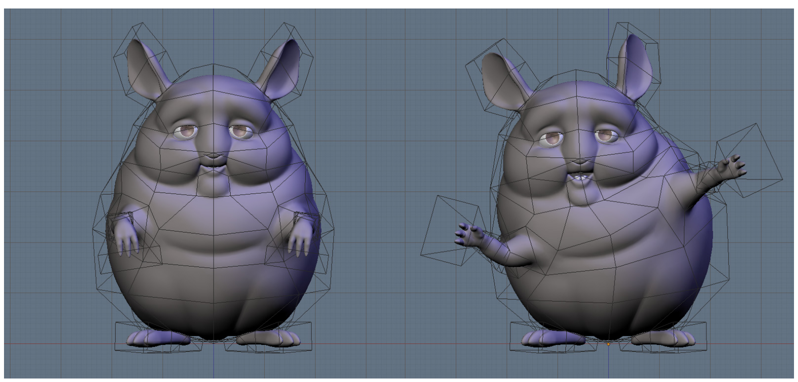

Manipulating control points, cages

image from Blender

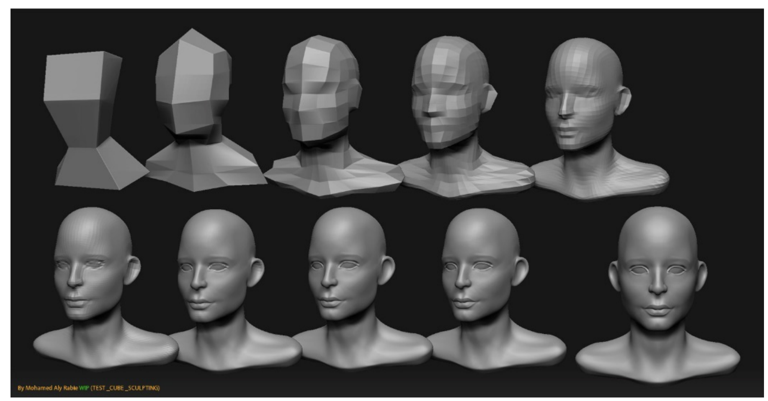
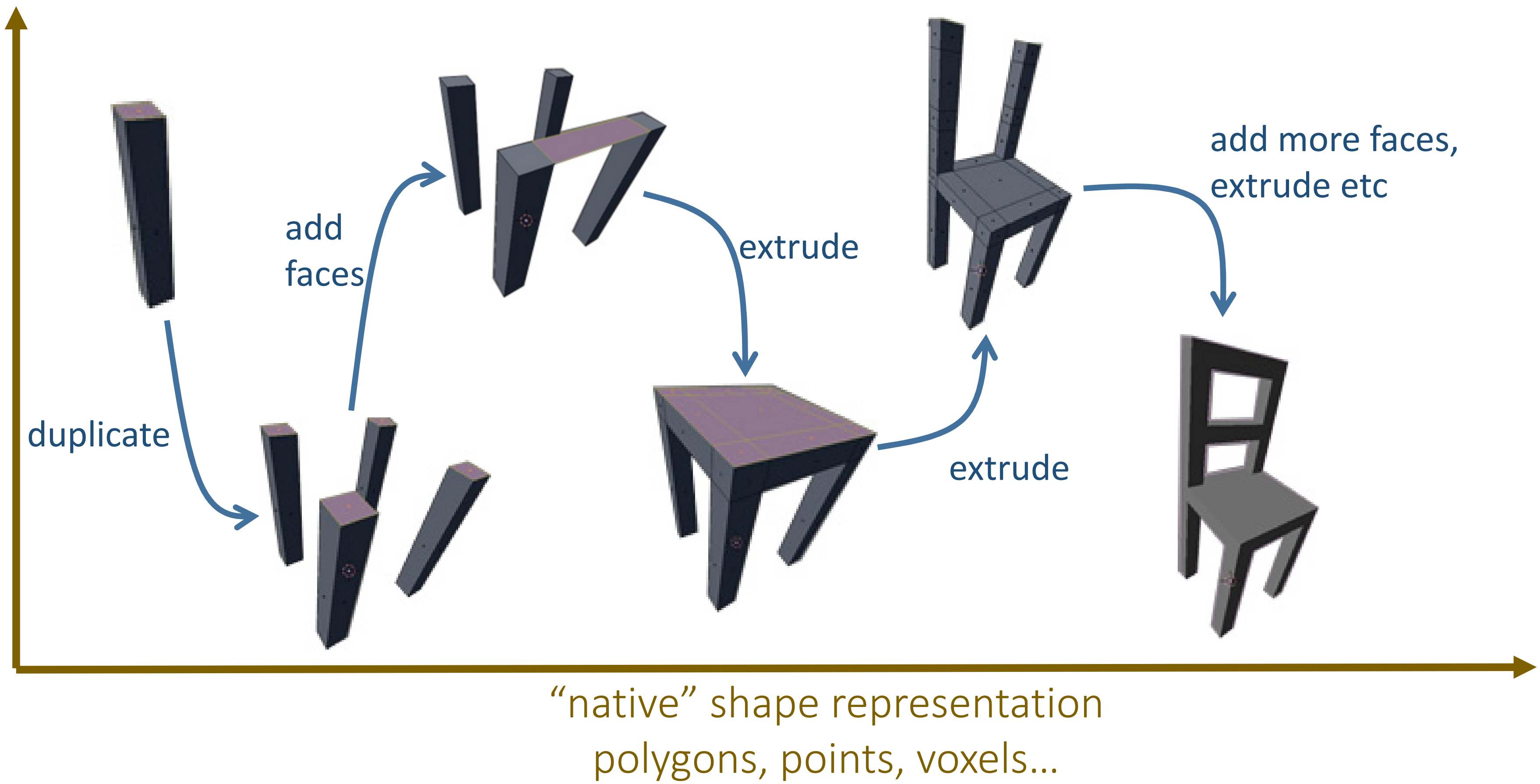

Digital Sculpting

image from Mohamed Aly Rable

Think of a "shape space" traversed by "low-level" operations



Images from flossmanuals.net

“Traditional” Geometric Modeling

Impressive results at the hands of **experienced users**

Operations requires **exact and accurate input**

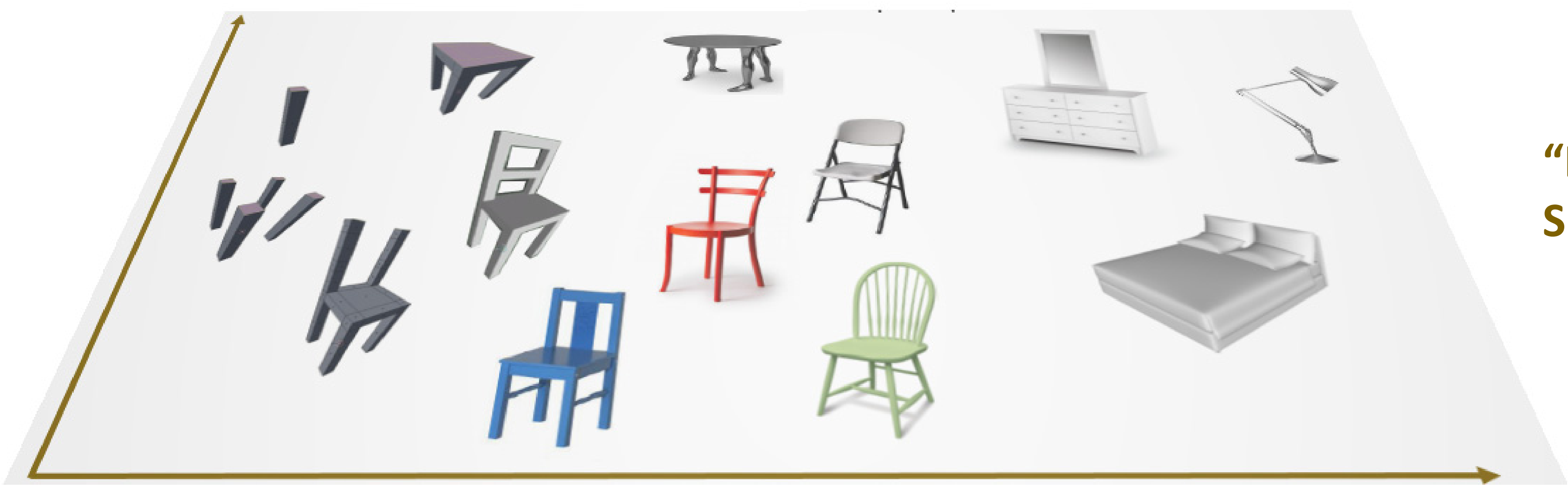
Creating compelling 3D models **takes lots of time**

Tools usually have **steep learning curves**

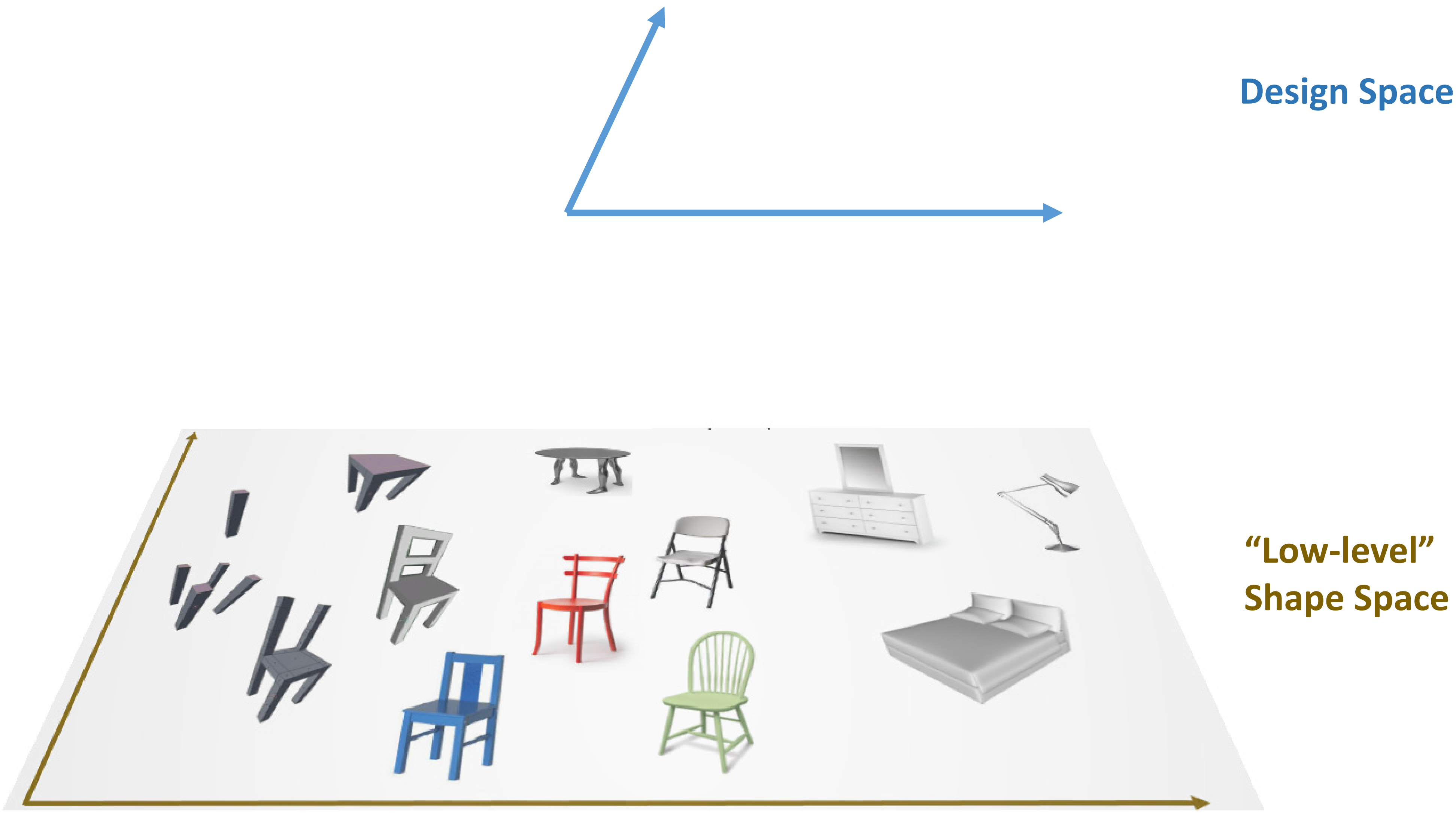
An alternative approach to geometric modeling

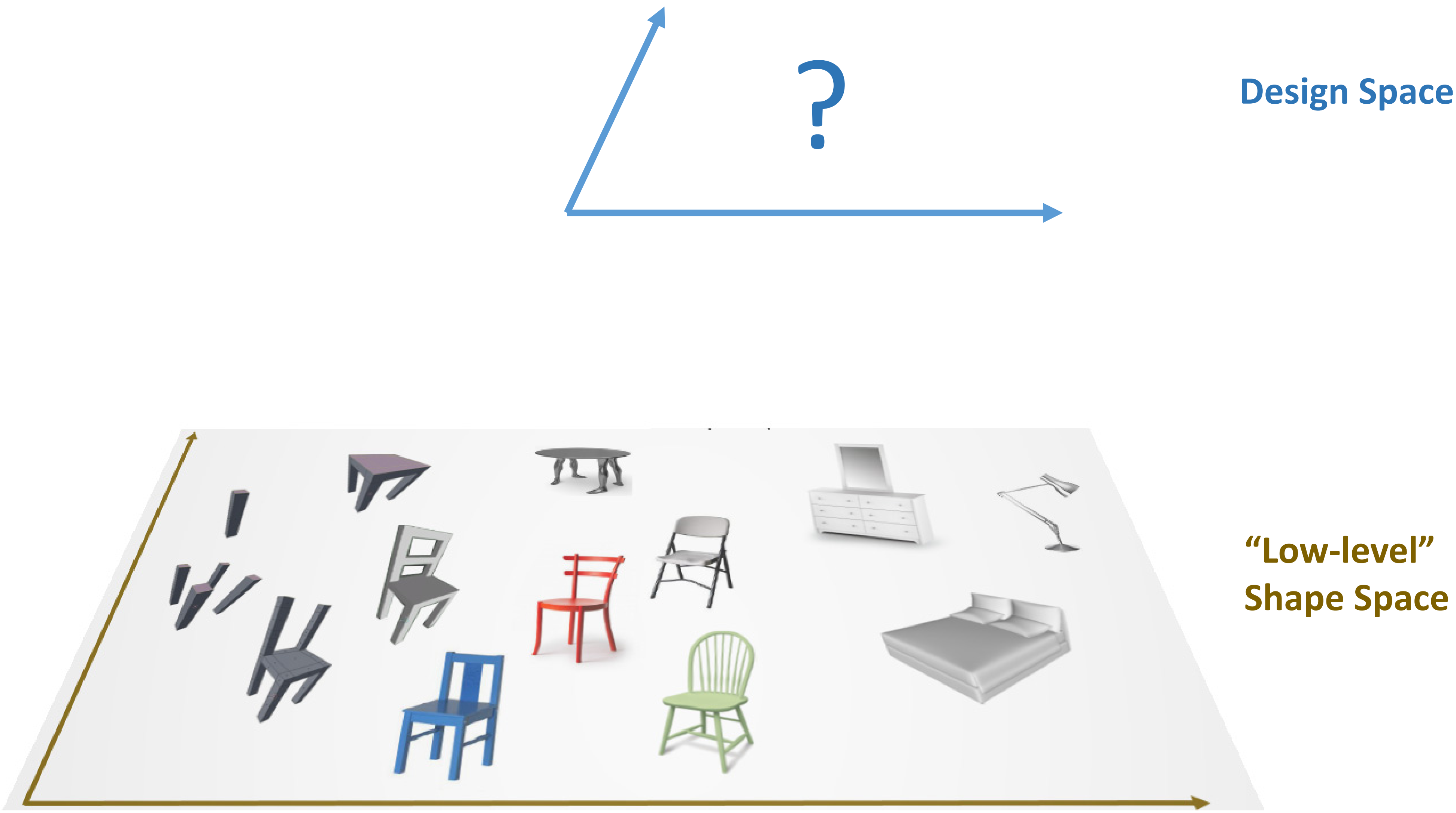
- Users provide high-level, possibly approximate input
- Computers learn to generate low-level, accurate geometry

➢ **Machine learning!**

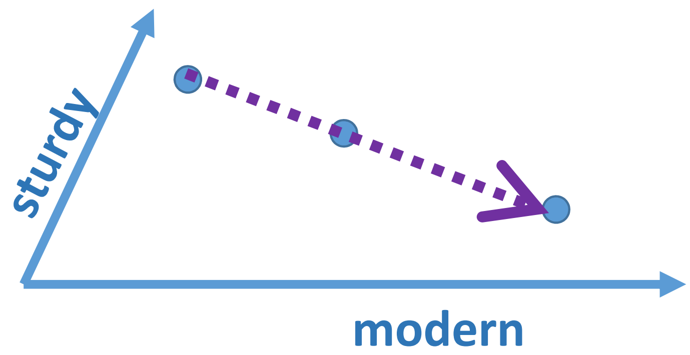


“Low-level”
Shape Space

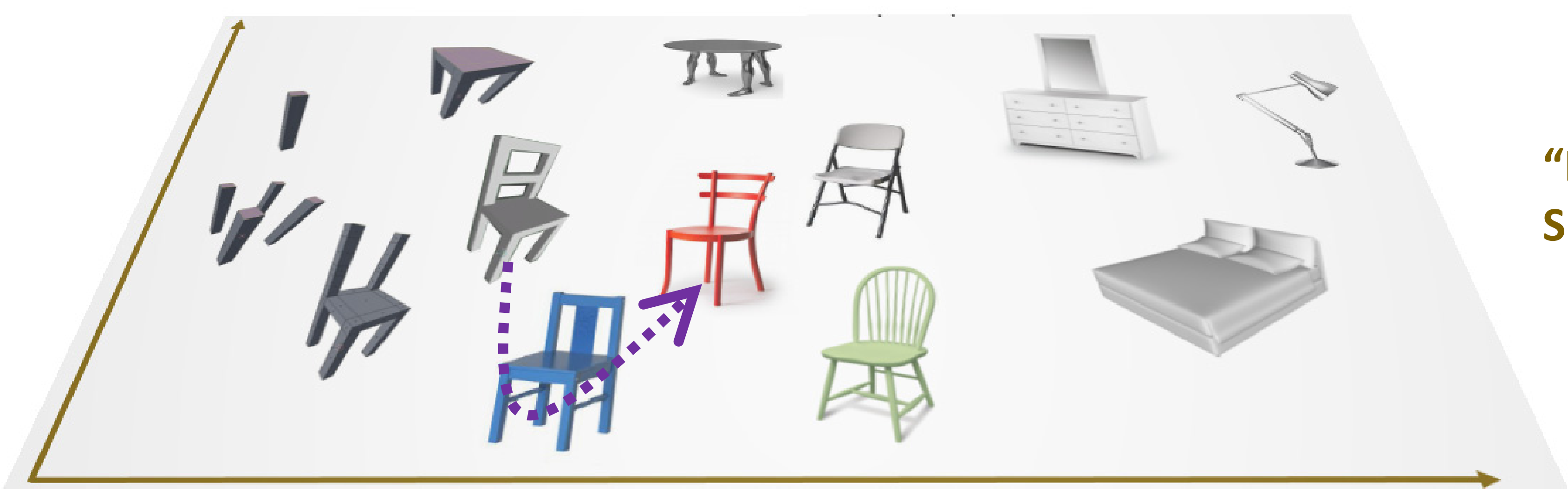

Design Space
“Low-level”
Shape Space


?
Design Space
"Low-level"
Shape Space

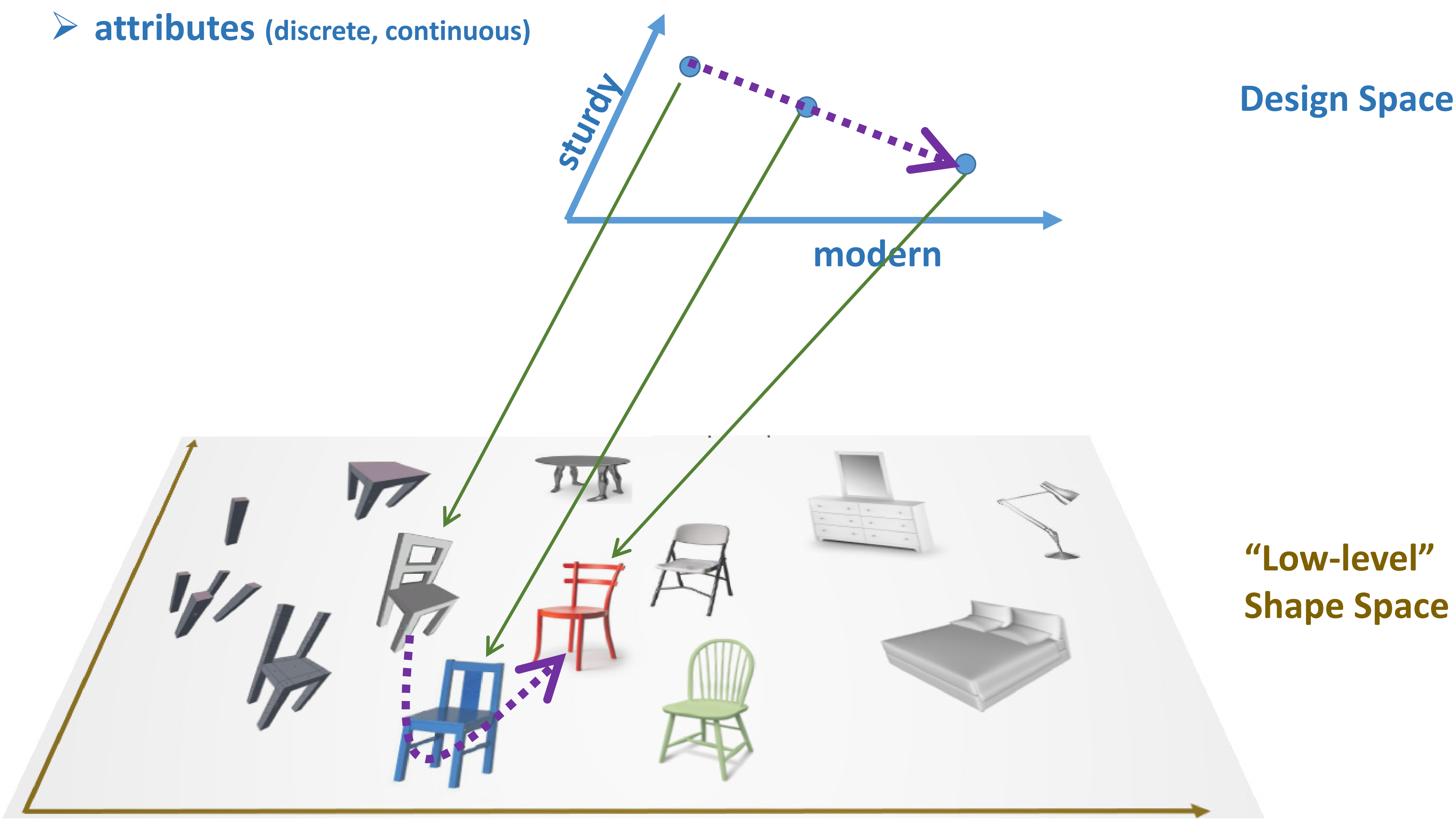
- **attributes** (discrete, continuous)



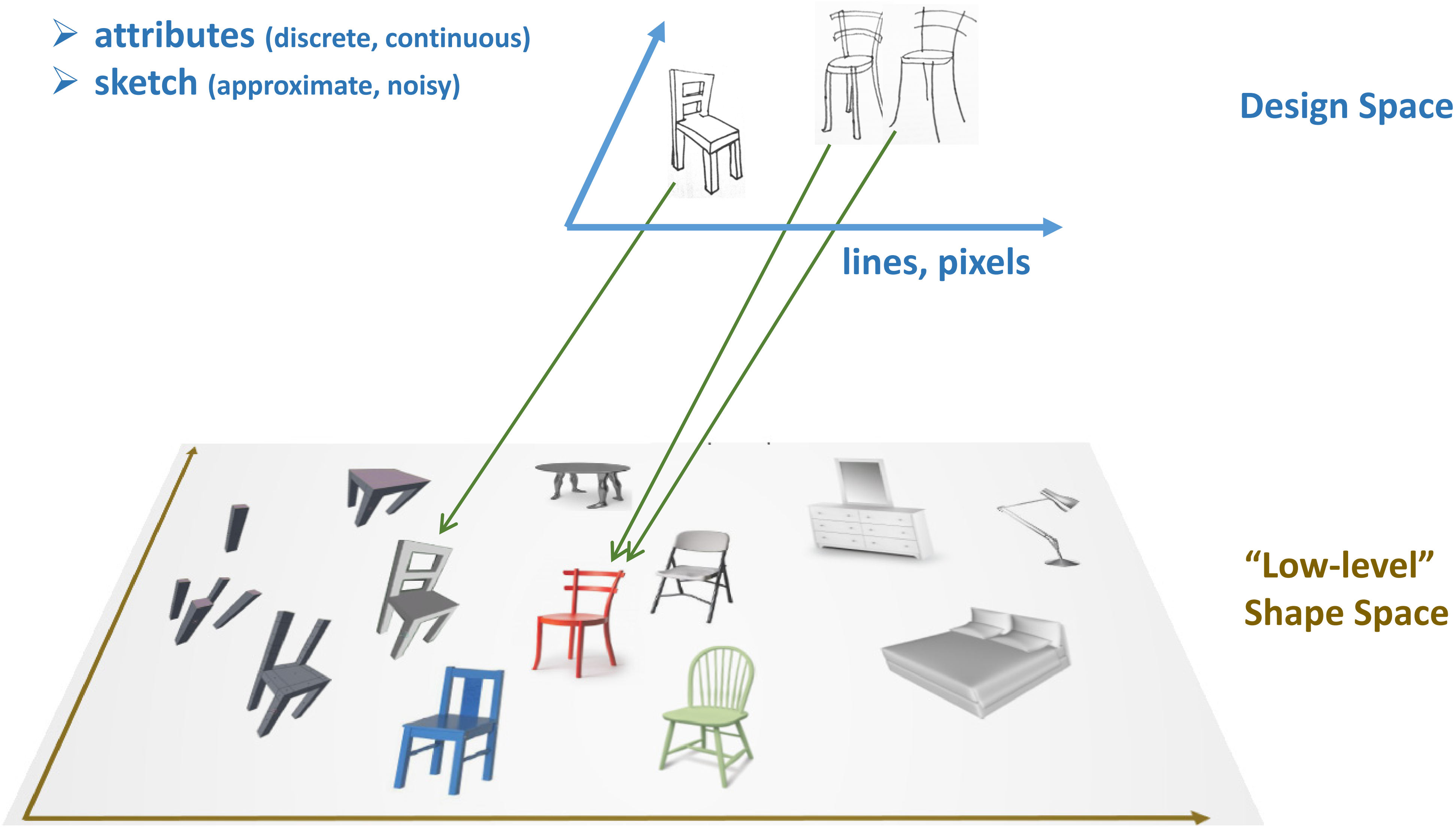
Design Space



"Low-level" Shape Space

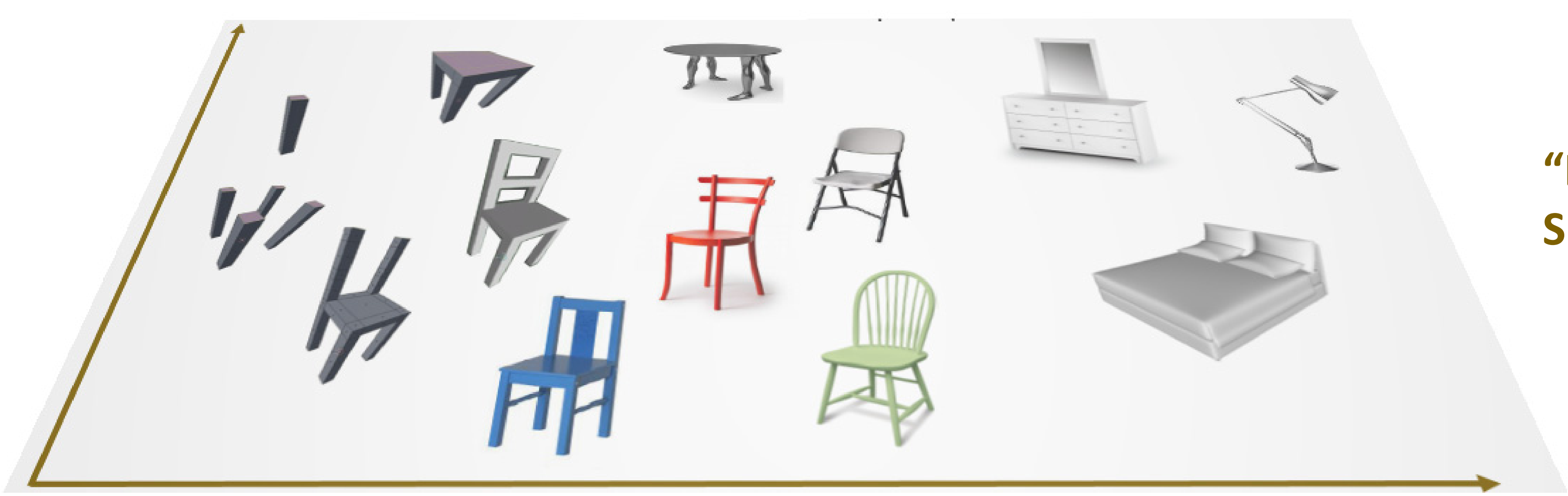


attributes (discrete, continuous)
sturdy
modern
Design Space
"Low-level"
Shape Space


attributes (discrete, continuous)
sketch (approximate, noisy)
Design Space
lines, pixels
"Low-level"
Shape Space

- **attributes** (discrete, continuous)
- **sketch** (approximate, noisy)
- **gestures**
- **natural language**
- **brain signals etc**

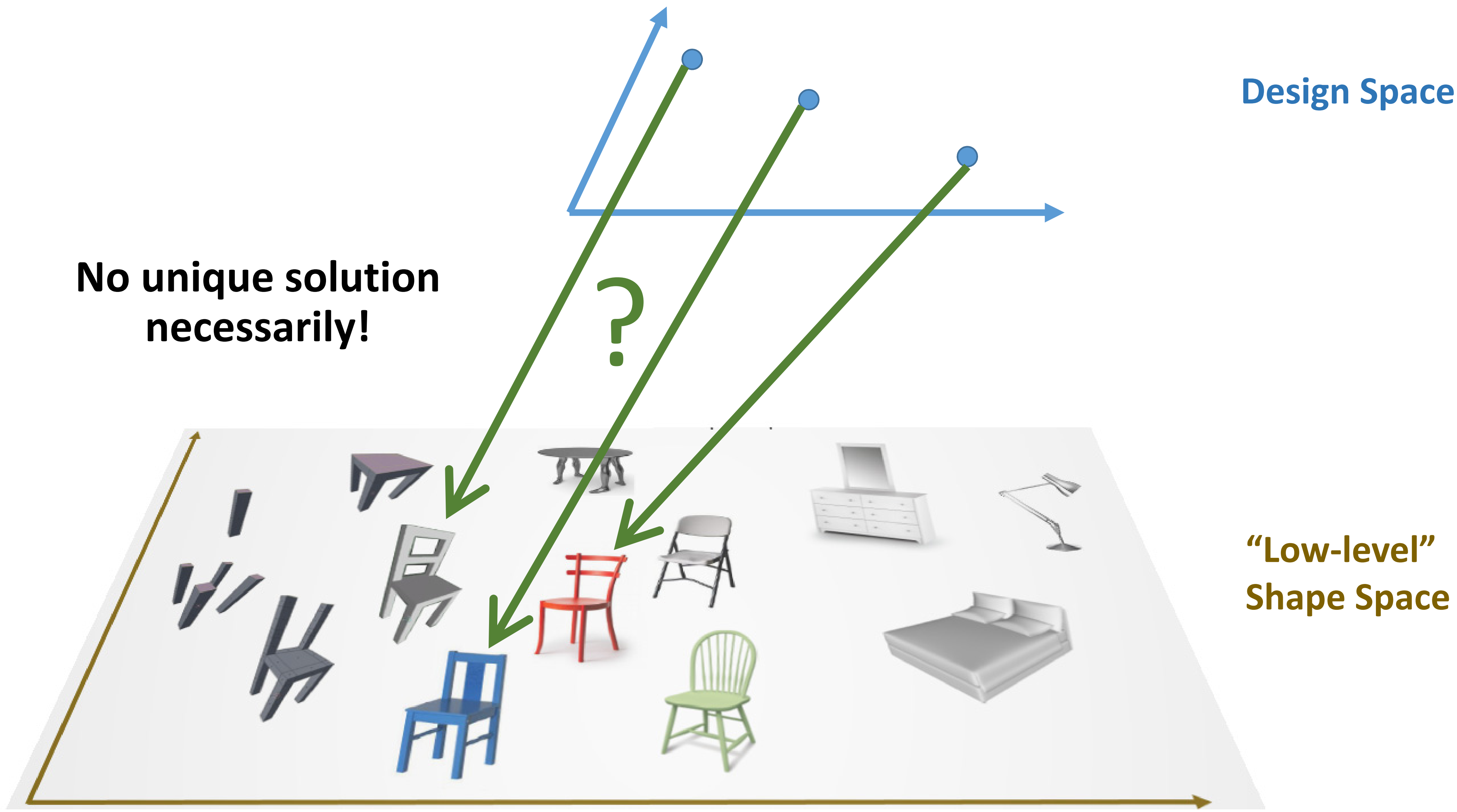
Design Space



"Low-level" Shape Space

Machine learning for Geometric Modeling

- Learn mappings from design to “low-level” space


Design Space
No unique solution necessarily!
?
"Low-level" Shape Space

Machine learning for Geometric Modeling

- Learn mappings from design to “low-level” space

- Learn which shapes are probable (“plausible”) given input

“Plausible” chairs

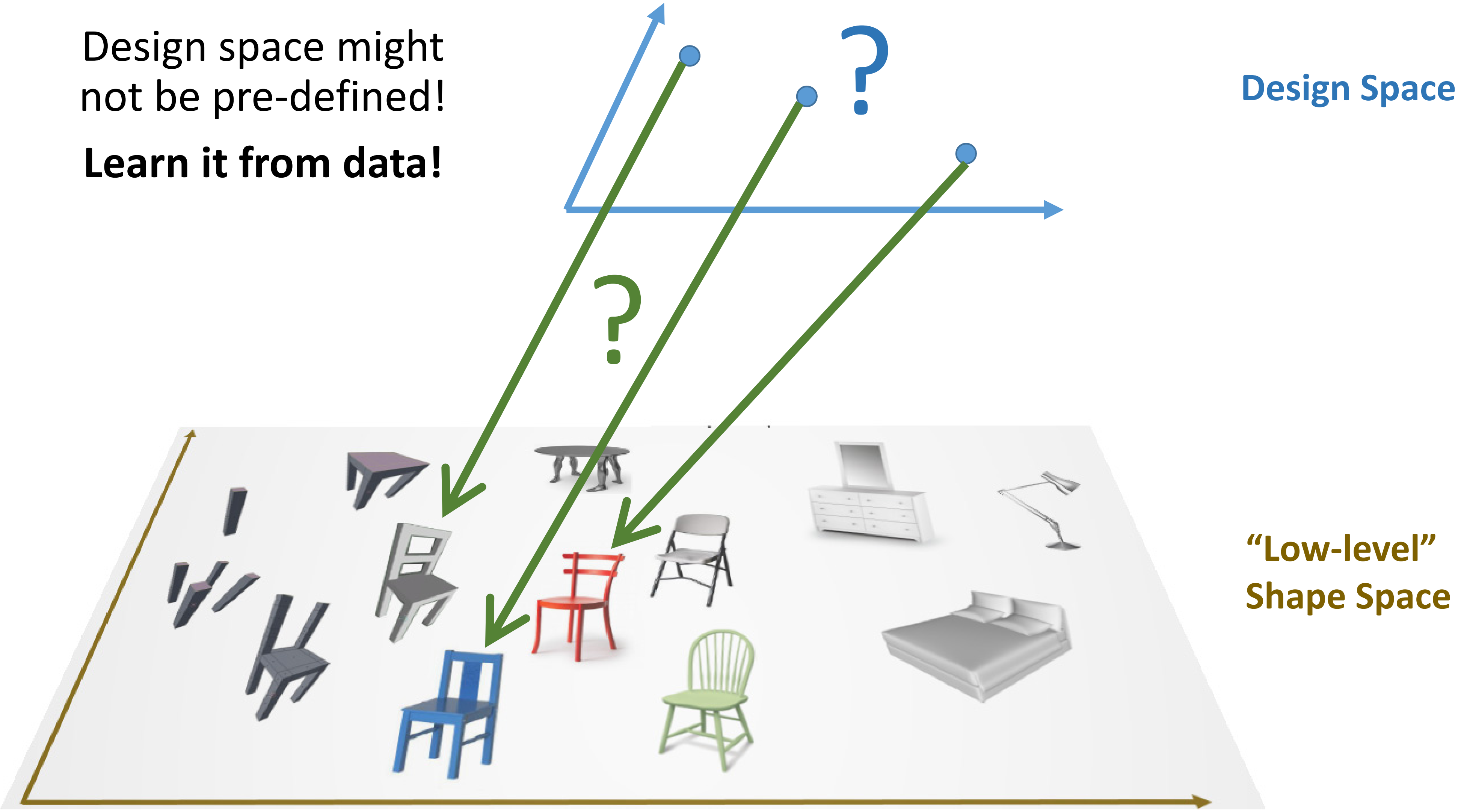
“Plausible” chairs

Machine learning for Geometric Modeling

- Learn mappings from design to “low-level” space
- Learn which shapes are probable (“plausible”) given input
- Learn design space (“high-level” representation)

Design space might
not be pre-defined!

Learn it from data!

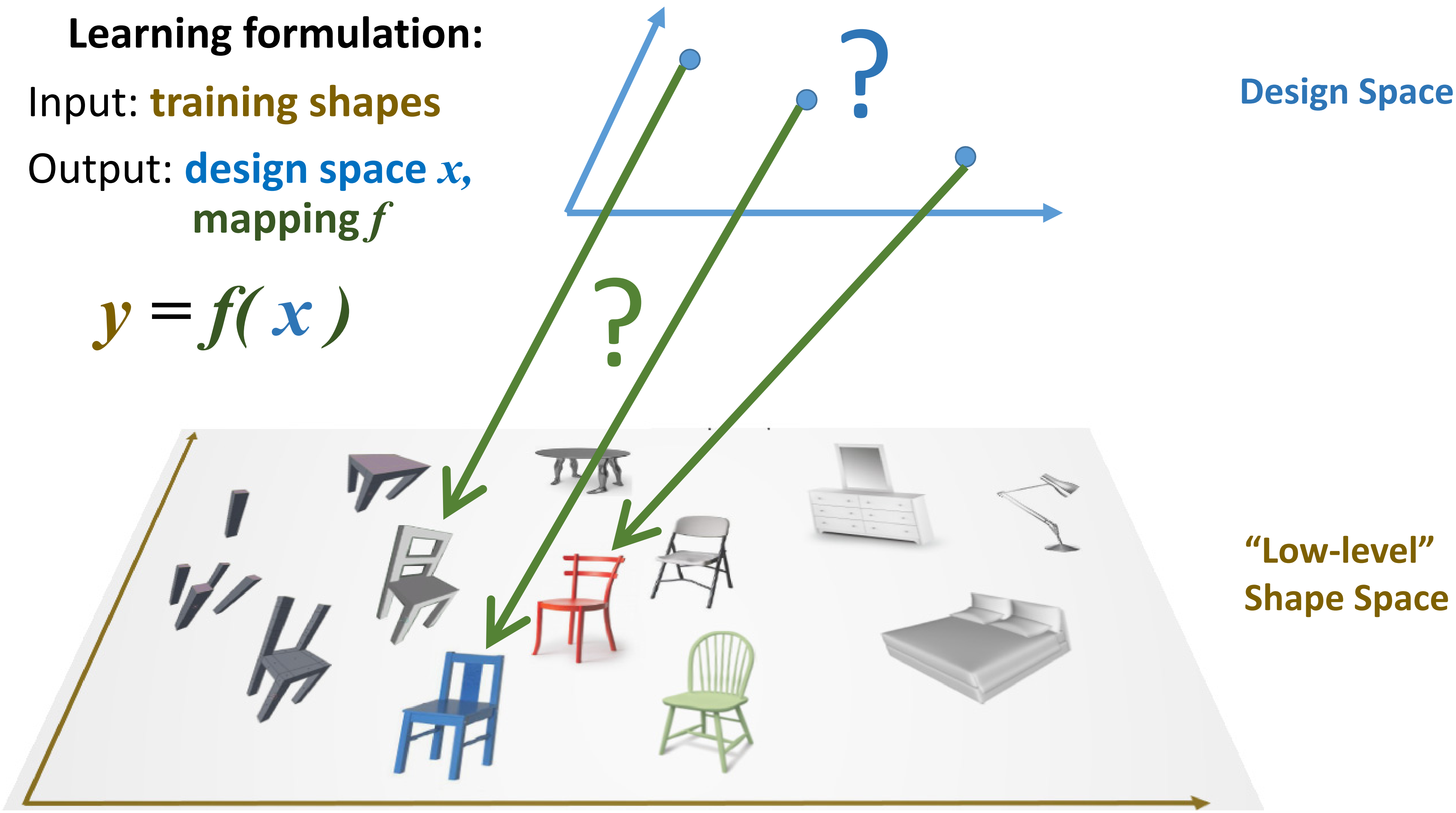


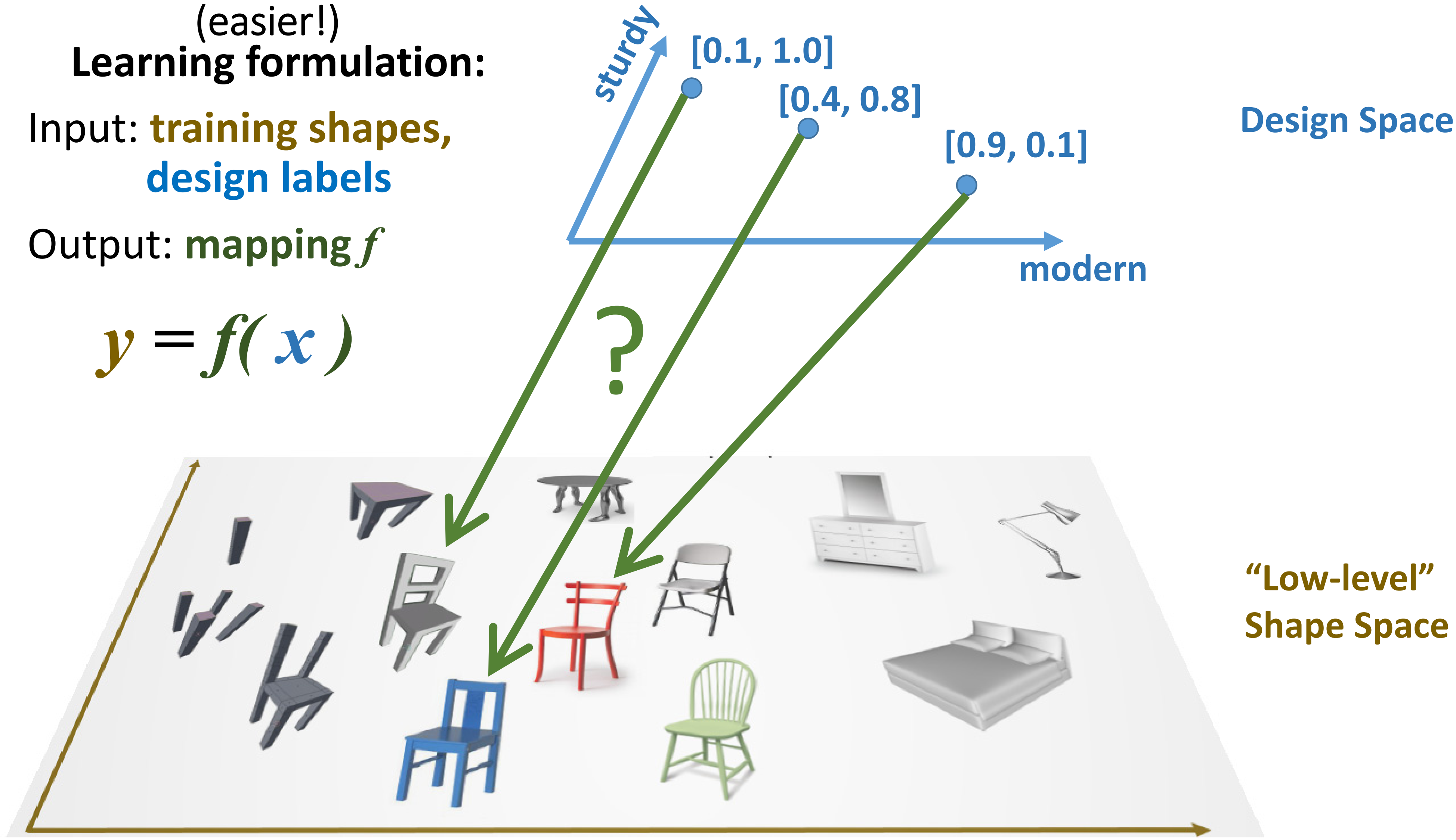
Learning formulation:

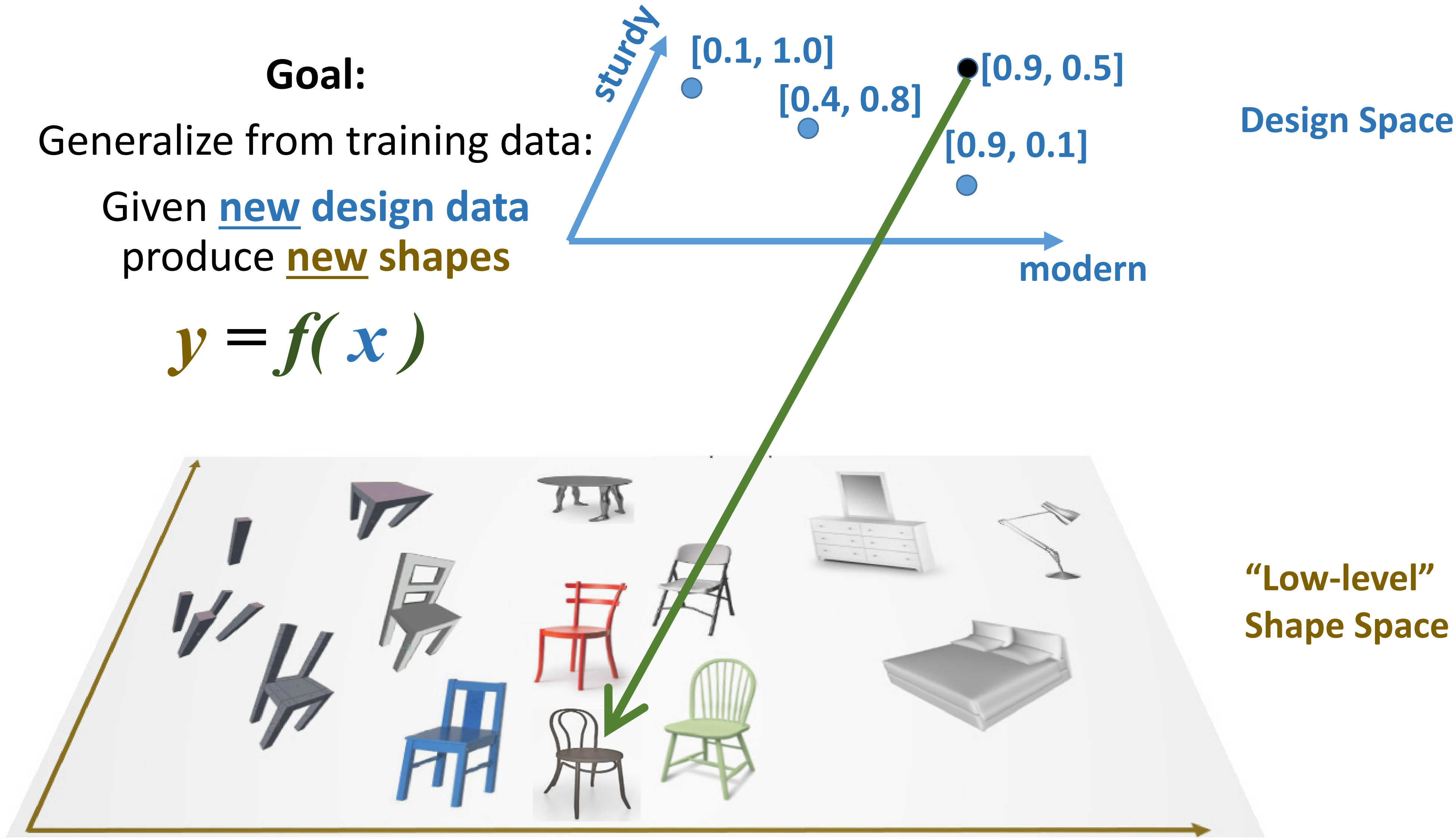
Input: **training shapes**

Output: **design space *x*,**
mapping *f*

$$y = f(x)$$



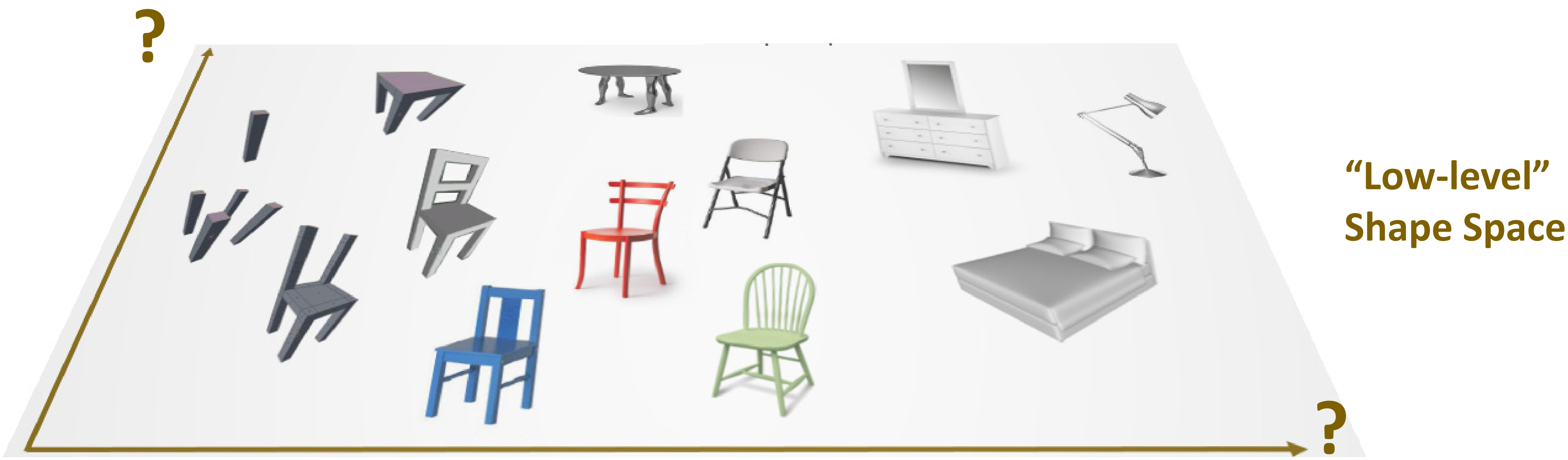

(easier!)
Learning formulation:
Input: training shapes, design labels
Output: mapping f
$y = f(x)$
sturdy
[0.1, 1.0]
[0.4, 0.8]
[0.9, 0.1]
modern
Design Space
?
"Low-level" Shape Space


Goal:
Generalize from training data:
Given new design data
produce new shapes
y = f(x)
sturdy
modern
[0.1, 1.0]
[0.4, 0.8]
[0.9, 0.5]
[0.9, 0.1]
Design Space
"Low-level"
Shape Space

Fundamental challenges

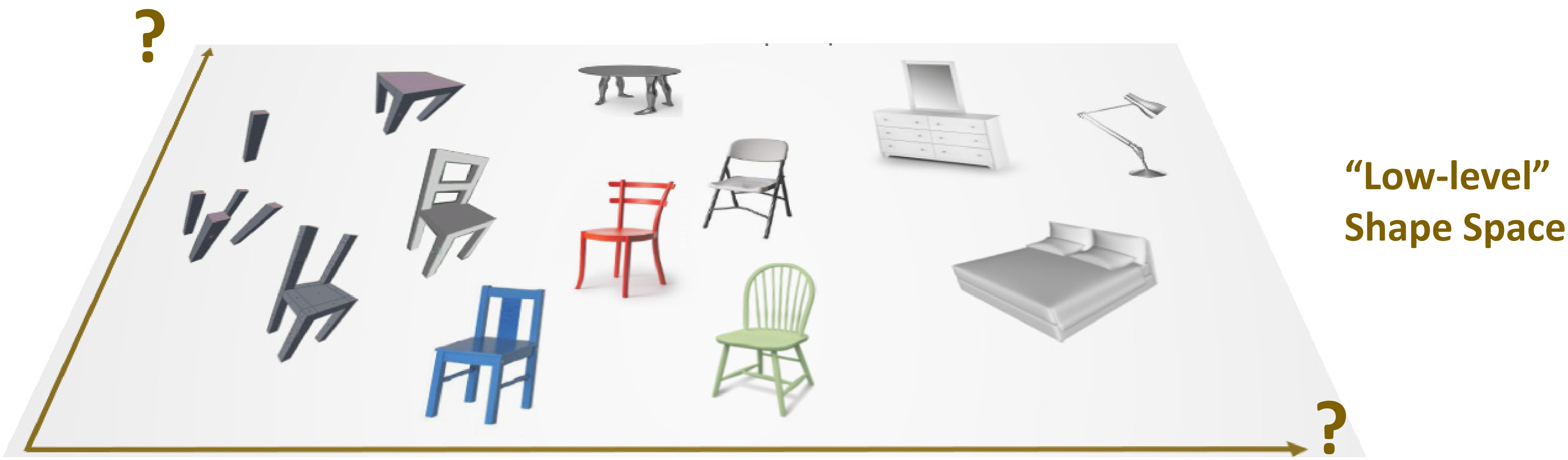
- How do we represent the **shape space**?

“Low-level” shape space representation



“Low-level” shape space representation

Can we use the polygon meshes as-is for our shape space?

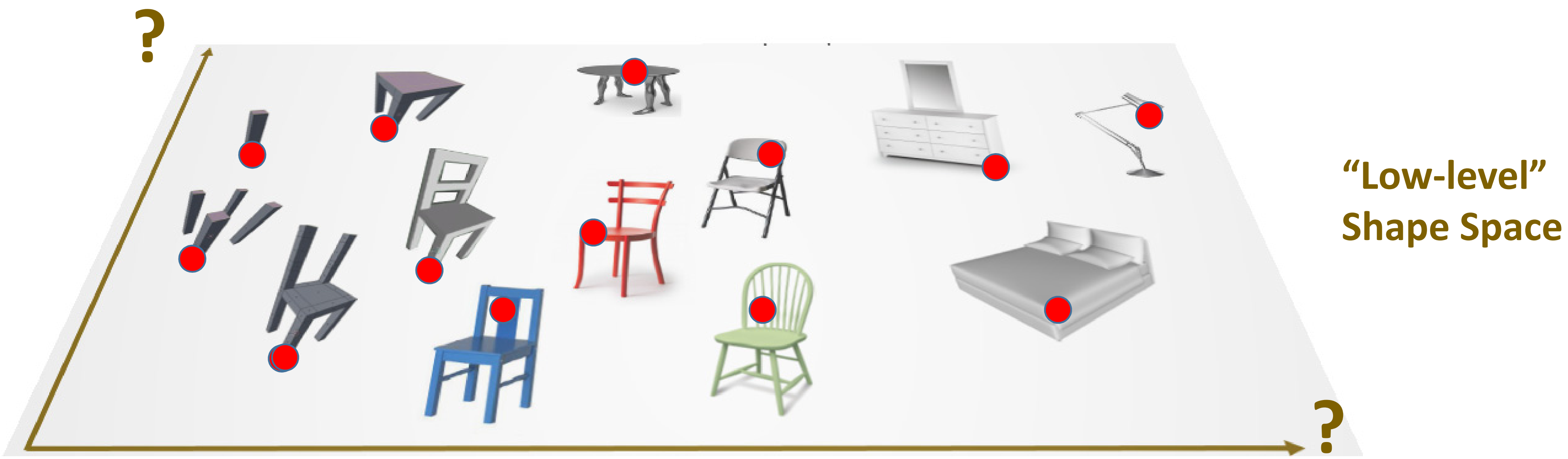


“Low-level” shape space representation

Can we use the polygon meshes as-is for our shape space?

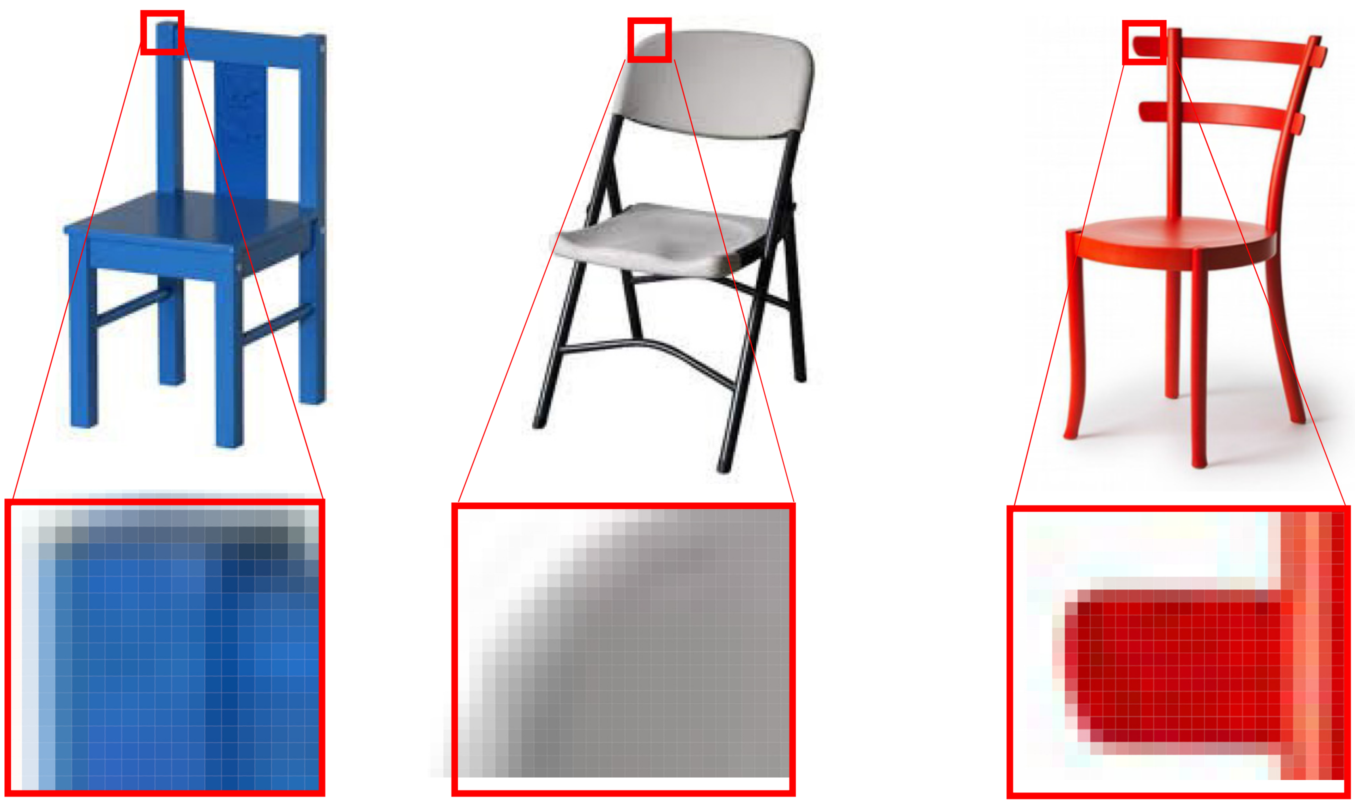
No. Take the first vertex on each mesh. Where is it?

Meshes have different number of vertices, faces etc



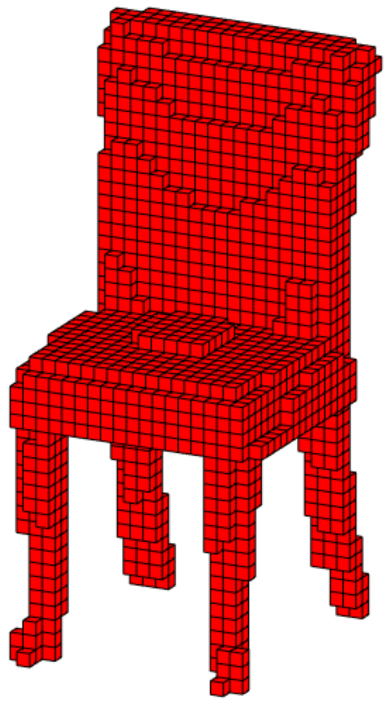
“Low-level” shape space representation – the “computer vision” approach

Learn from pixels & multiple views! Produce pixels! Include view information?



“Low-level” shape space representation – another “computer vision” approach

Learn from voxels! Produce voxels! Include orientation information?



“Low-level” shape space representation – correspondences

Find point correspondences between 3D surface points. Can do aligment.
Can we always have dense correspondences?

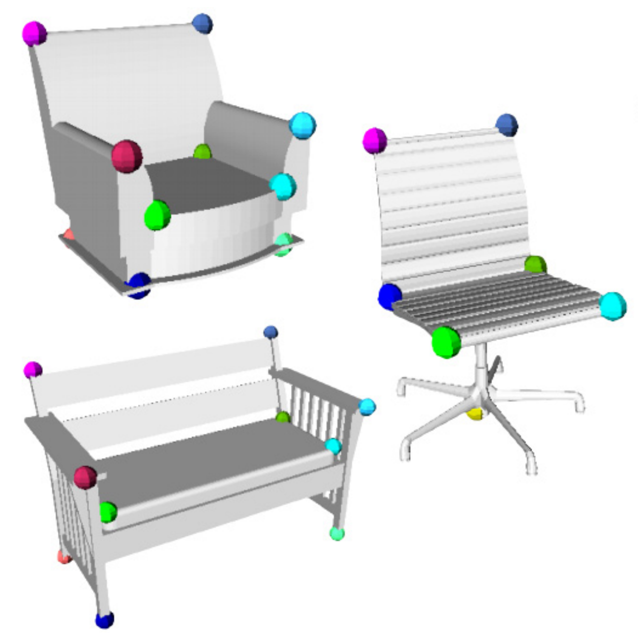


Image from Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser, “Learning Part-based Templates from Large Collections of 3D Shapes”, 2013

“Low-level” shape space representation – abstractions

Parameterize shapes with primitives (cuboids, cylinders etc)
How can we produce surface detail?

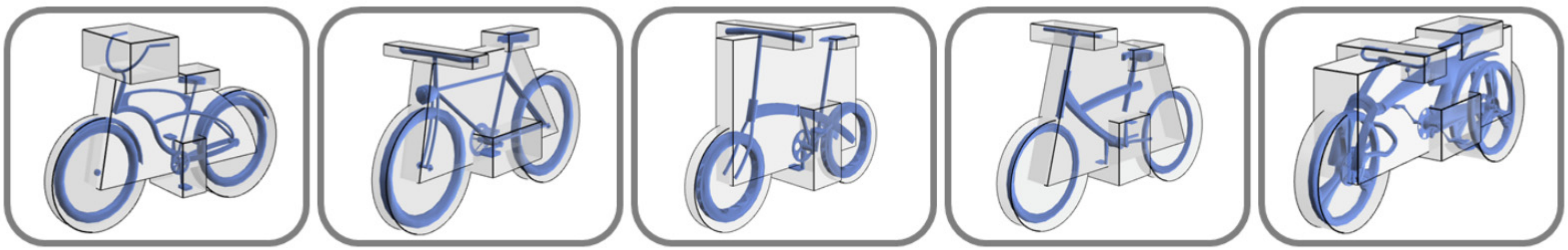
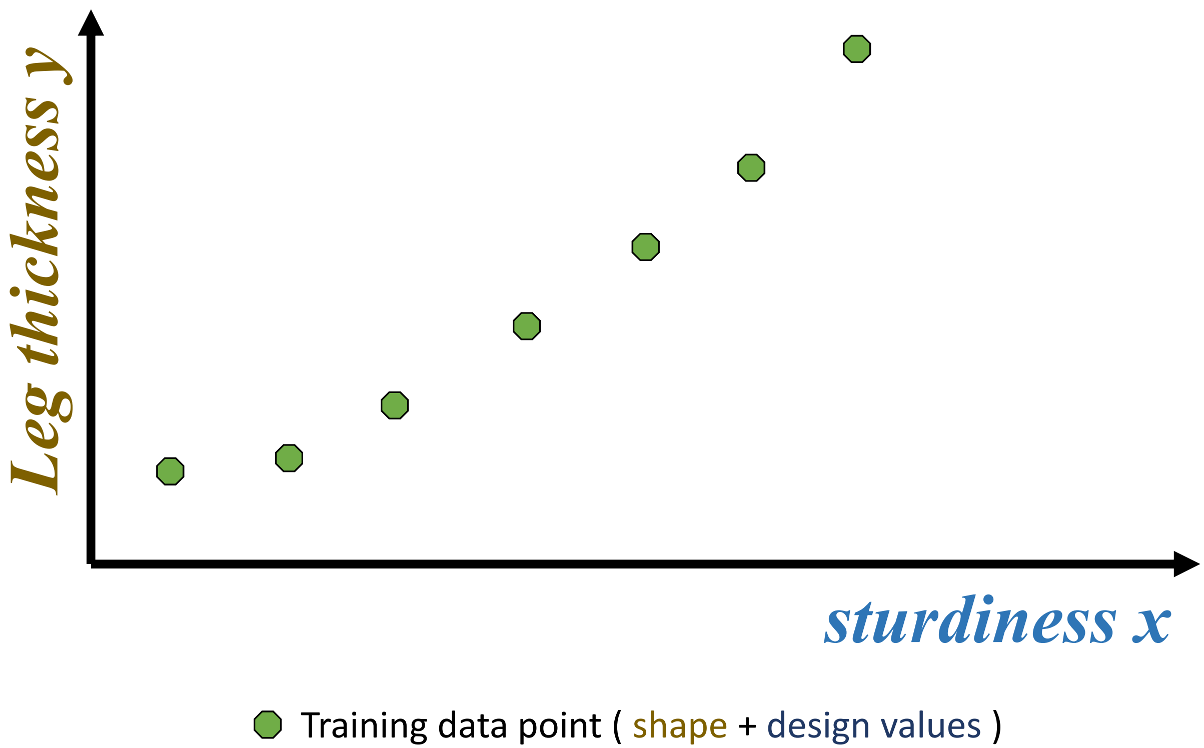


Image from E. Yumer., L. Kara, Co-Constrained Handles for Deformation in Shape Collections, 2014

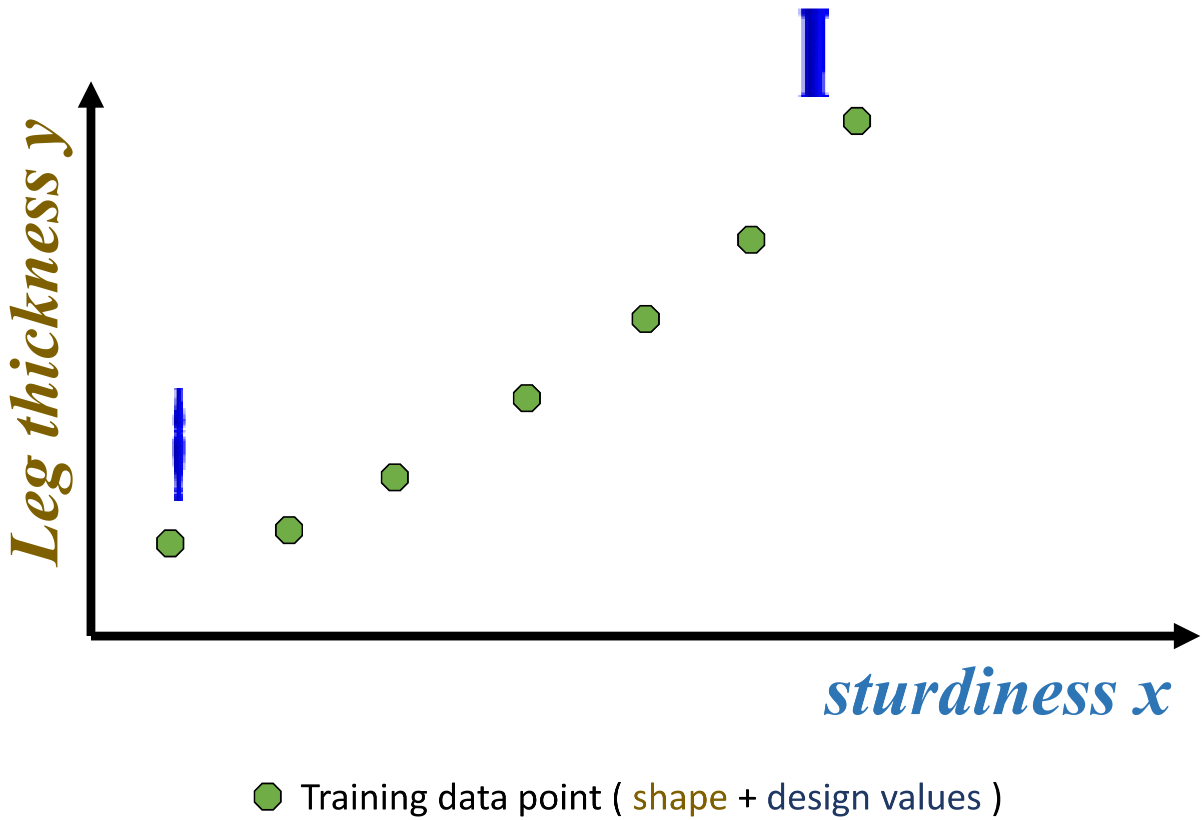
Fundamental challenges

- How do we represent the **shape space**?
- What is the form of the **mapping**? **How is it learned**?

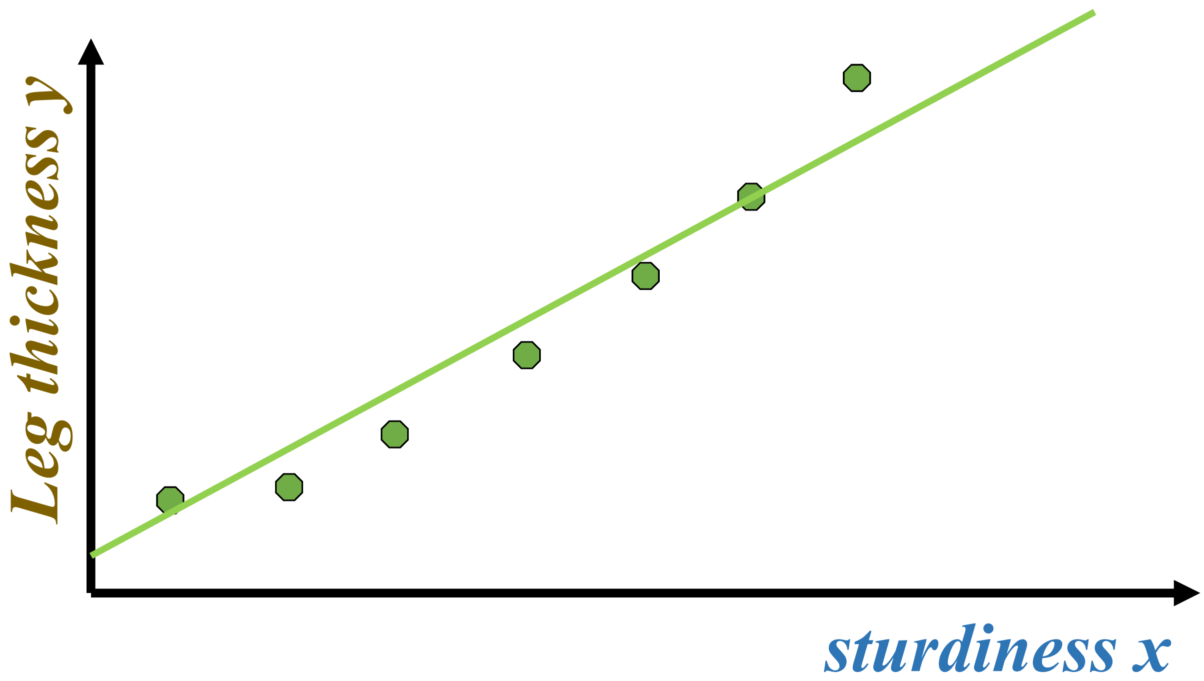
Regression example (simplistic)



Regression example (simplistic)

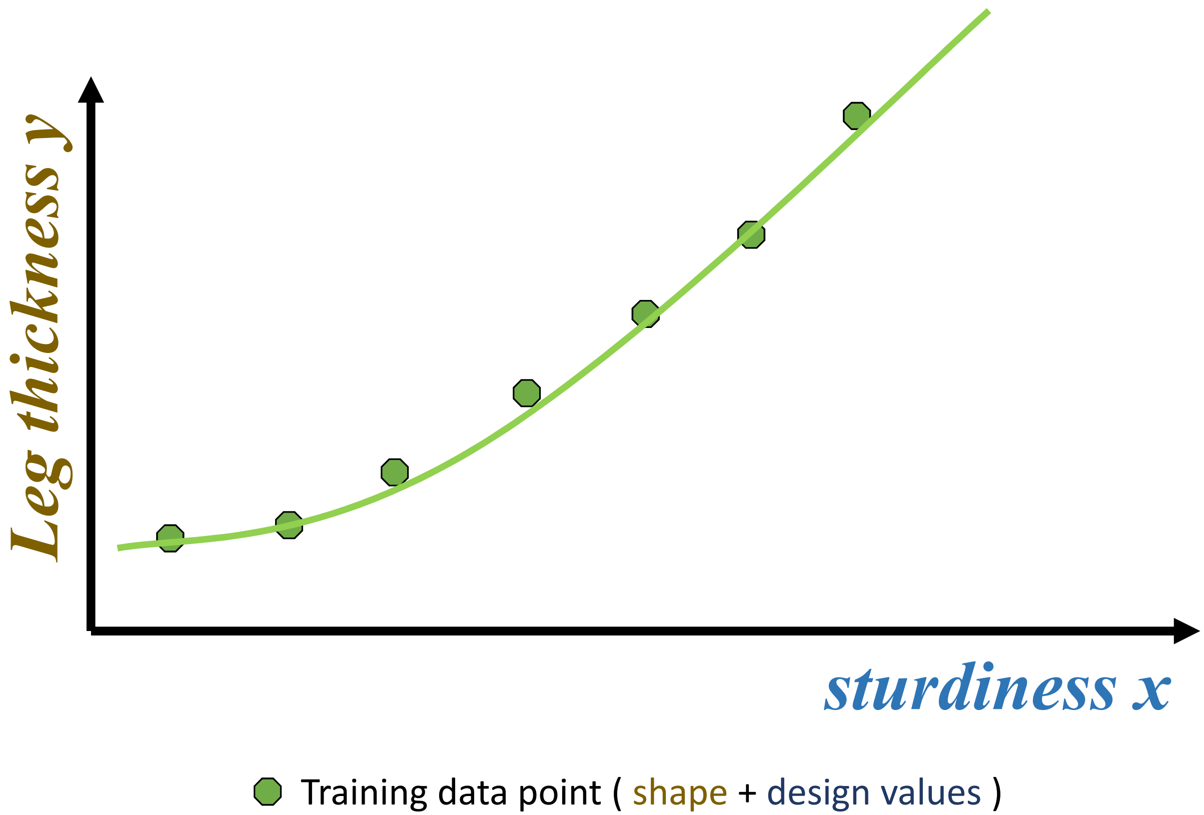


Regression example (simplistic)

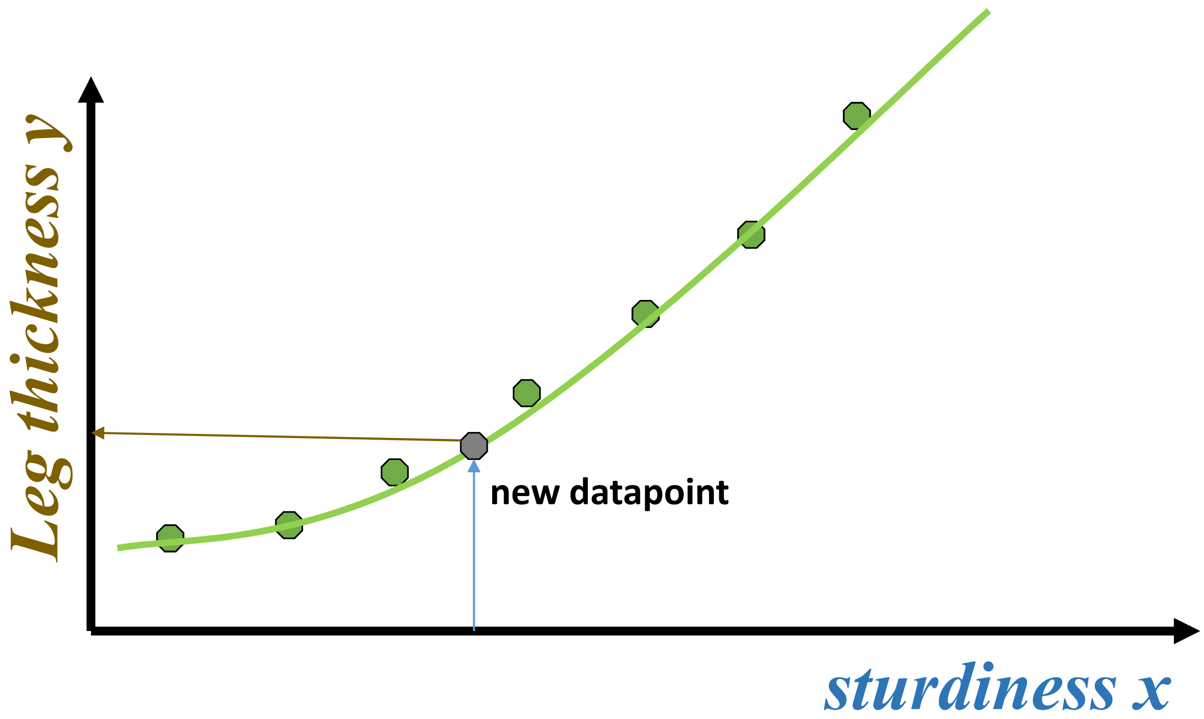


Training data point (shape + design values)

Regression example (simplistic)

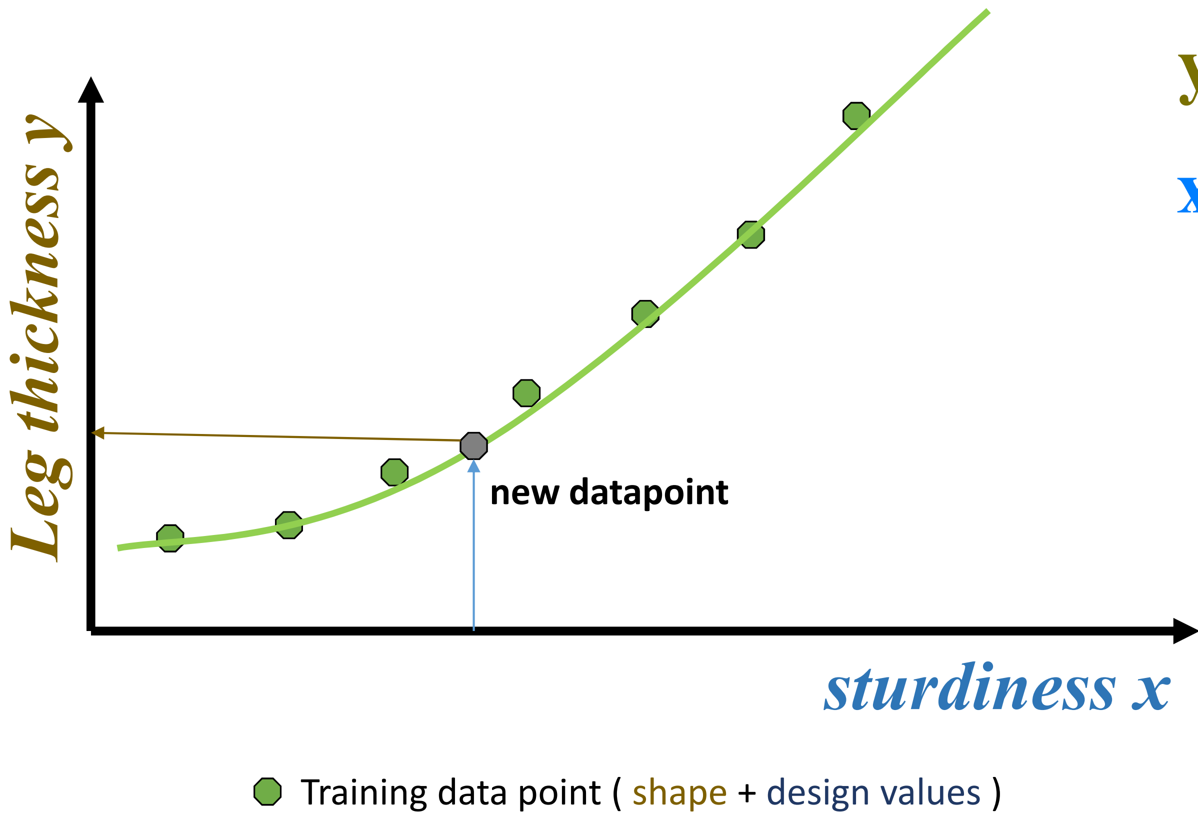


Regression example (simplistic)



Training data point (shape + design values)

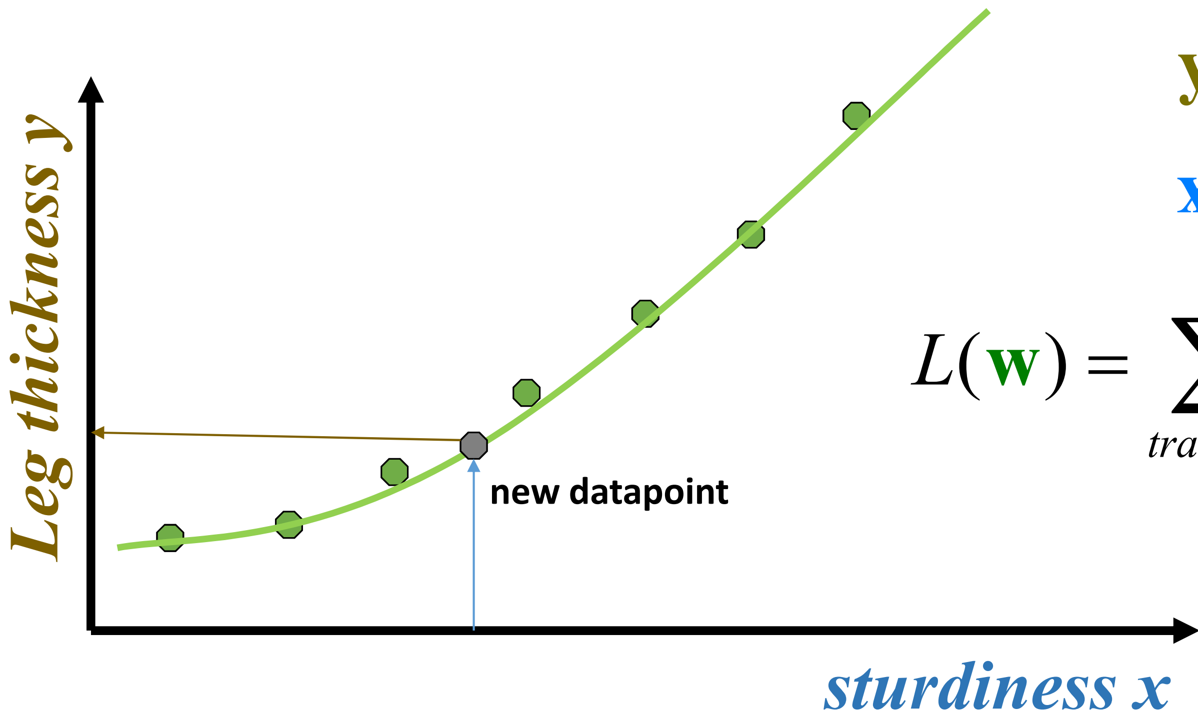
Regression example (simplistic)



$$\mathbf{y} = \mathbf{w} \cdot \mathbf{x}'$$

$$\mathbf{x}' = [\mathbf{x}^2 \;\; \mathbf{x} \;\; 1]$$

Regression example (simplistic)



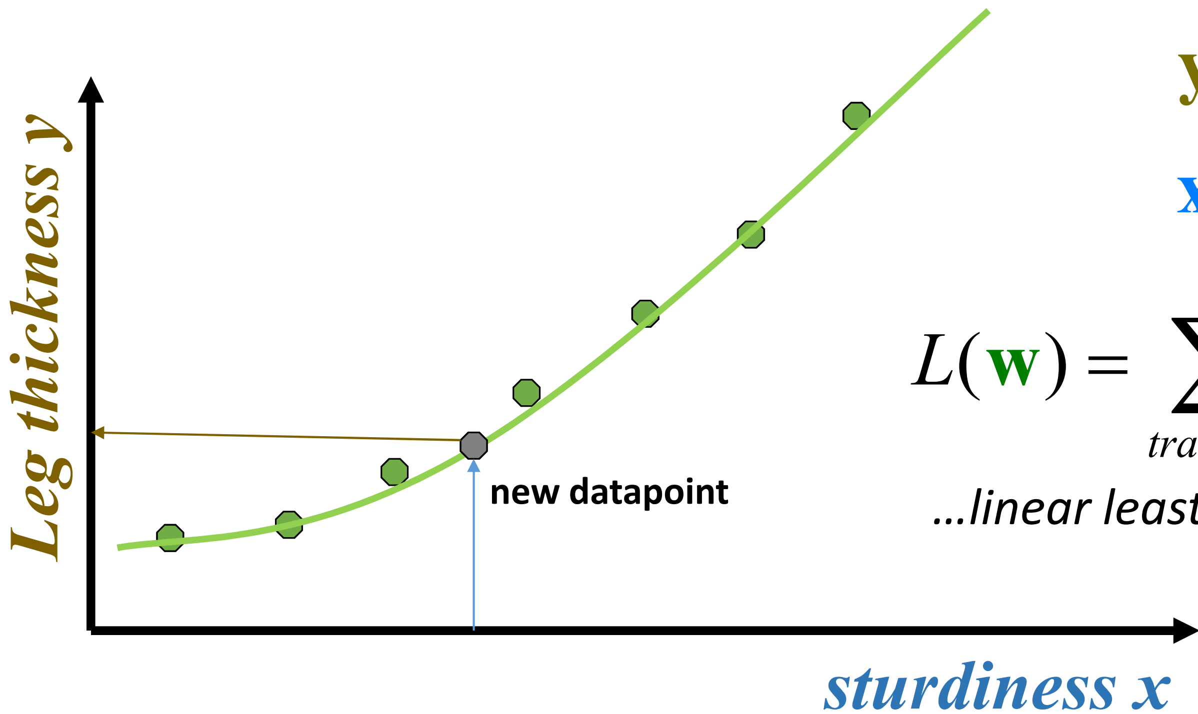
$$\mathbf{y} = \mathbf{w} \cdot \mathbf{x}'$$

$$\mathbf{x}' = [\mathbf{x}^2 \;\; \mathbf{x} \;\; 1]$$

$$L(\mathbf{w}) = \sum_{train.\, i} [\mathbf{y}_i - \mathbf{w} \cdot \mathbf{x}_i']^2$$

Training data point (shape + design values)

Regression example (simplistic)



$$\mathbf{y} = \mathbf{w} \cdot \mathbf{x}'$$

$$\mathbf{x}' = [\mathbf{x}^2 \;\; \mathbf{x} \;\; 1]$$

$$L(\mathbf{w}) = \sum_{train.\, i} [\mathbf{y}_i - \mathbf{w} \cdot \mathbf{x}_i']^2$$

...linear least-squares solution...

Training data point (shape + design values)

Overfitting

Important to select a function that would **avoid overfitting** & **generalize** (produce reasonable outputs for inputs not encountered during training)

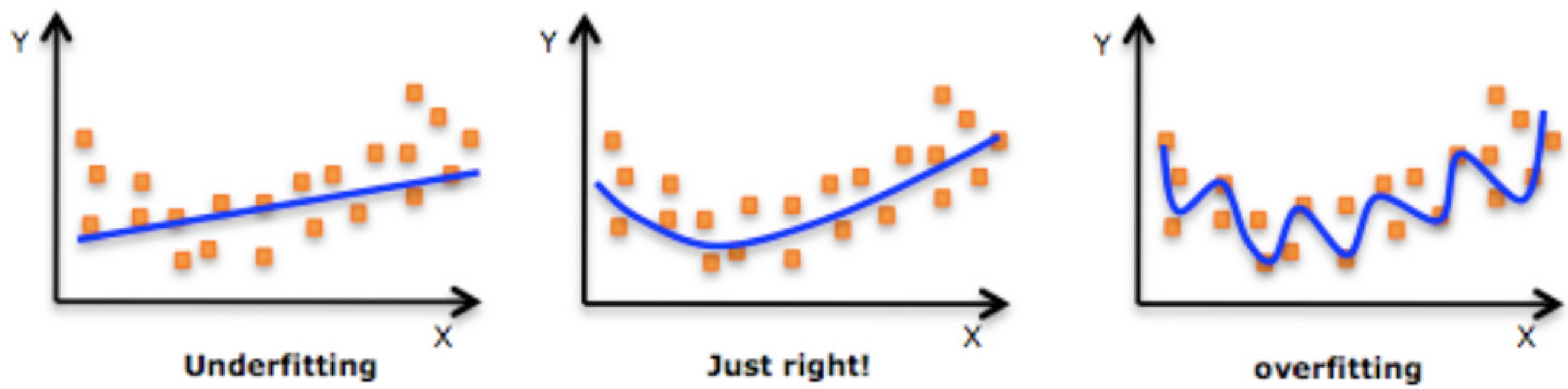


image from Andrew Ng's ML class (?)

Classification example (Logistic Regression)

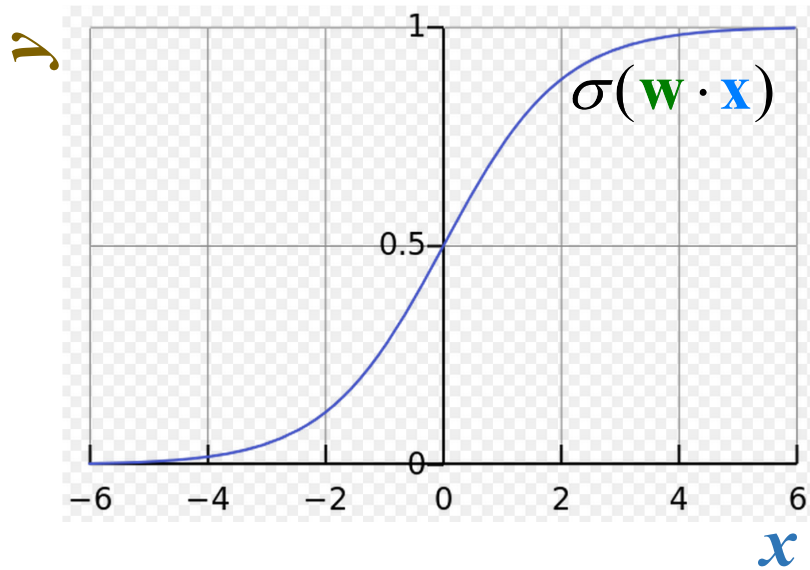
Suppose you want to predict pixels or voxels (on or off).

Probabilistic classification function:

$$P(\mathbf{y} = \mathbf{1} \mid \mathbf{x}) = \mathbf{f}(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

$$where:$$

$$\sigma(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})}$$



Logistic regression: training

Need to estimate parameters *w* from training data.

Find parameters that **maximize probability of training data**

$$\max_{\mathbf{w}} \prod_{i=1}^{N} P(\mathbf{y} = 1 \mid \mathbf{x}_i)^{[\mathbf{y_i} == 1]} [1 - P(\mathbf{y} = 1 \mid \mathbf{x}_i)]^{[\mathbf{y_i} == 0]}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data.

Find parameters that **maximize probability of training data**

$$\max_{\mathbf{w}} \prod_{i=1}^{N} \sigma(\mathbf{w} \cdot \mathbf{x}_i)^{[\mathbf{y_i}==1]}[1-\sigma(\mathbf{w} \cdot \mathbf{x}_i)]^{[\mathbf{y_i}==0]}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data.

Find parameters that **maximize log probability of training data**

$$\max_{\mathbf{w}} \log\{\prod_{i=1}^{N} \sigma(\mathbf{w}\cdot\mathbf{x}_i)^{[\mathbf{y_i}==1]}[1-\sigma(\mathbf{w}\cdot\mathbf{x}_i)]^{[\mathbf{y_i}==0]}\}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data.

Find parameters that **maximize log probability of training data**

$$\max_{\mathbf{w}} \sum_{i=1}^{N} [\mathbf{y}_{\mathbf{i}} == 1] \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + [\mathbf{y}_{\mathbf{i}} == 0] \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data.

Find parameters that **minimize negative log probability of training data**

$$\min_{\mathbf{w}} - \sum_{i=1}^{N} [\mathbf{y}_{\mathbf{i}} == 1] \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + [\mathbf{y}_{\mathbf{i}} == 0] \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data.

In other words, find parameters that **minimize the negative log likelihood function**

$$\min_{\mathbf{w}} -\sum_{i=1}^{N} [\mathbf{y}_i == 1] \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + [\mathbf{y}_i == 0] \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))$$

$\boldsymbol{L(w)}$

Logistic regression: training

Need to estimate parameters w from training data.

In other words, find parameters that **minimize the negative log likelihood function**

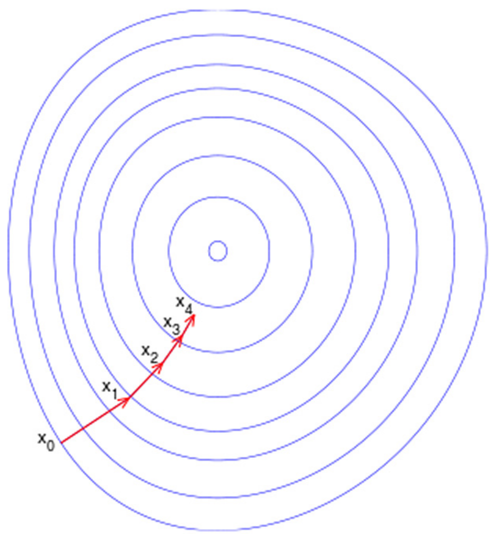
$$\min_{\mathbf{w}} -\sum_{i=1}^{N}[\mathbf{y}_i == 1]\log\sigma(\mathbf{w}\cdot\mathbf{x}_i) + [\mathbf{y}_i == 0]\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}_i))$$

$$L(w)$$

$$\frac{\partial L(\mathbf{w})}{\partial w_d} = \sum_i x_{i,d}[\sigma(\mathbf{w}\cdot\mathbf{x}_i) - y_i]$$

(partial derivative for d^{th} parameter)

How can you minimize/maximize a function?



Gradient descent: Given a random initialization of parameters and a step rate η, update them according to:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \eta \nabla L(\mathbf{w})$$

See also **quasi-Newton** and **IRLS** methods

Regularization

Overfitting: few training data and number of parameters is large!

Penalize large weights - shrink weights:

$$\min_{\mathbf{w}} L(\mathbf{w}) + \lambda \sum_{d} \mathrm{w}_{d}{}^{2}$$

Called **ridge regression (or L2 regularization)**

Regularization

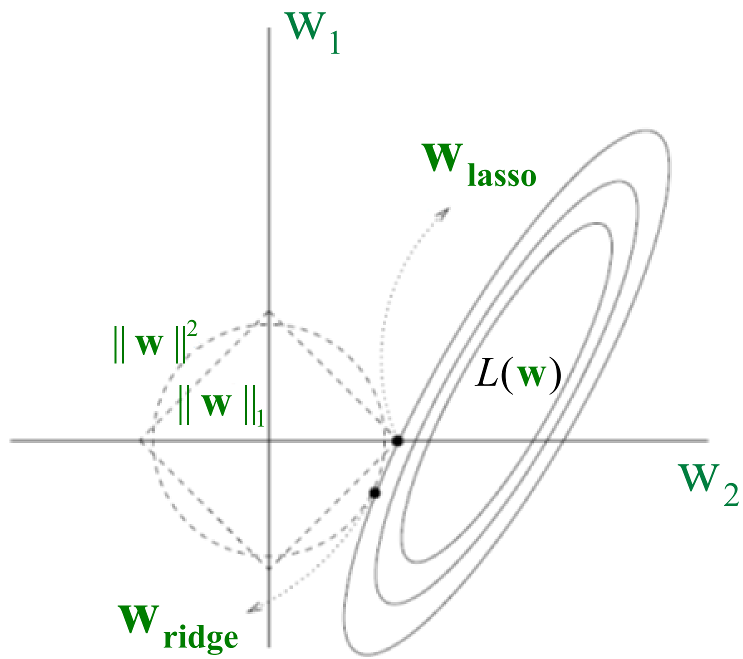
Overfitting: few training data and number of parameters is large!

Penalize non-zero weights - push as many as possible to **0**:

$$\min_{\mathbf{w}} L(\mathbf{w}) + \lambda \sum_{d} | \mathrm{w}_d |$$

Called **Lasso (or L1 regularization)**

Lasso vs Ridge Regression



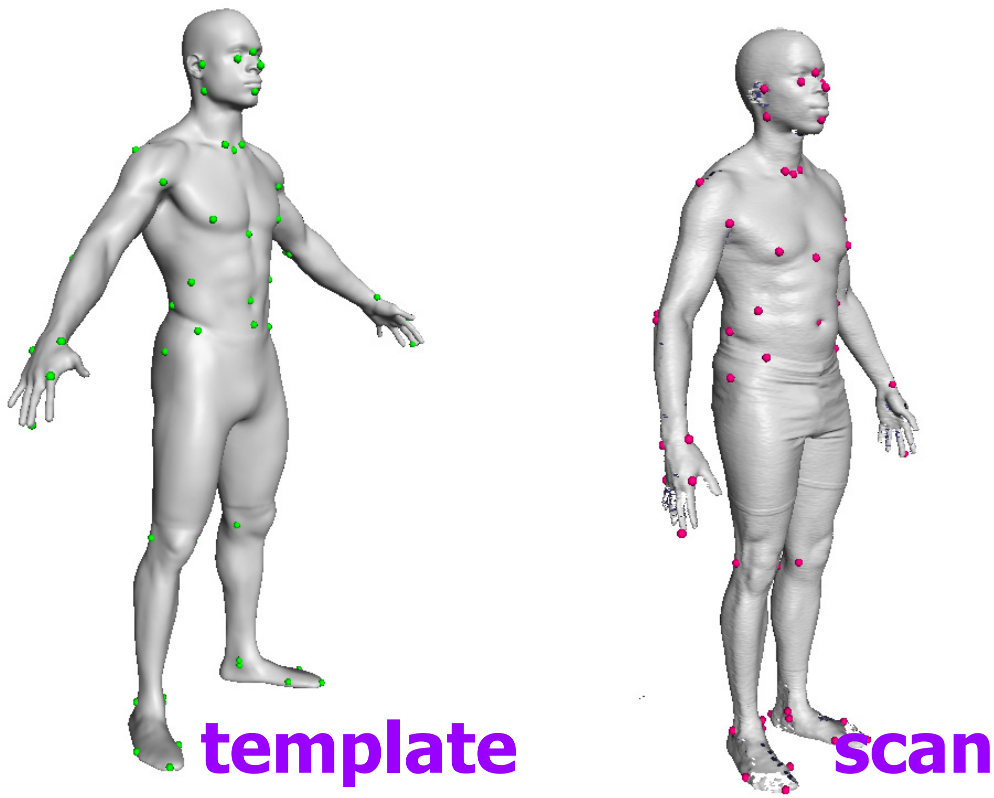
Modified image from Robert Tibsirani, Regression shrinkage and selection via the lasso, 1996

Case study: the space of human bodies

Training shapes: 125 male + 125 female scanned bodies

Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

Matching algorithm

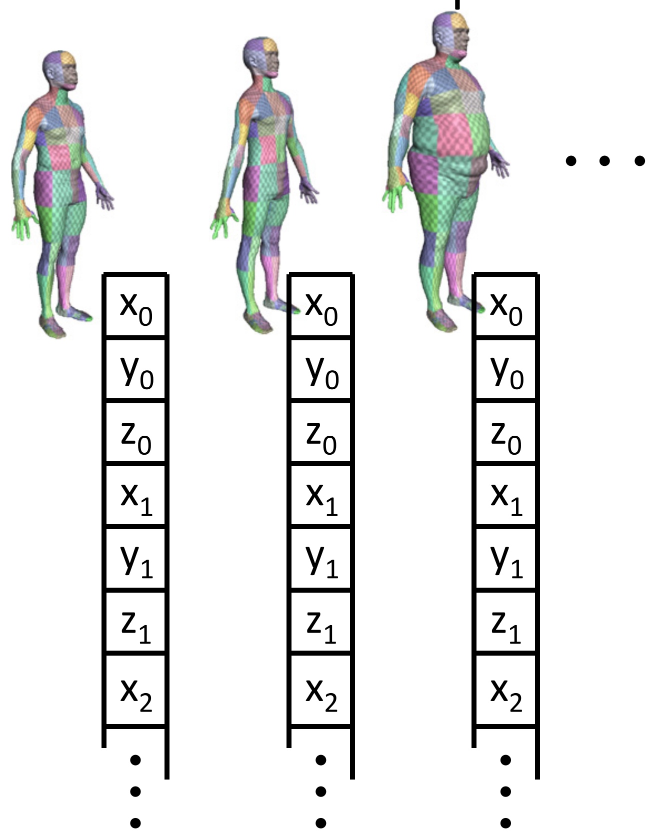


Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

Matching algorithm

Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

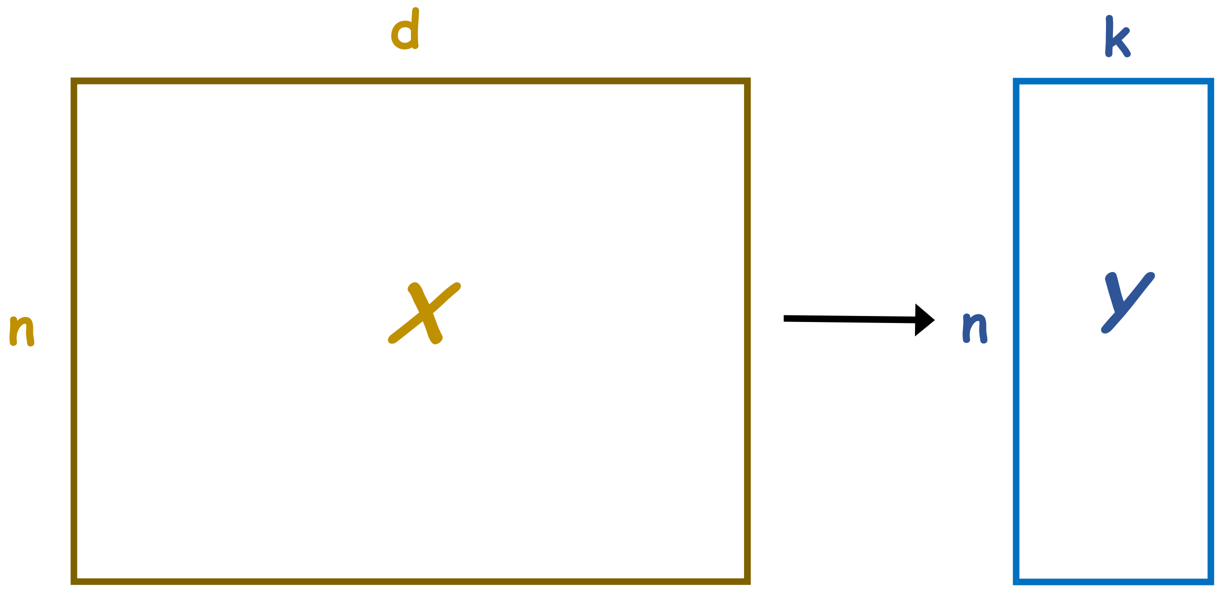
Principal Component Analysis



Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

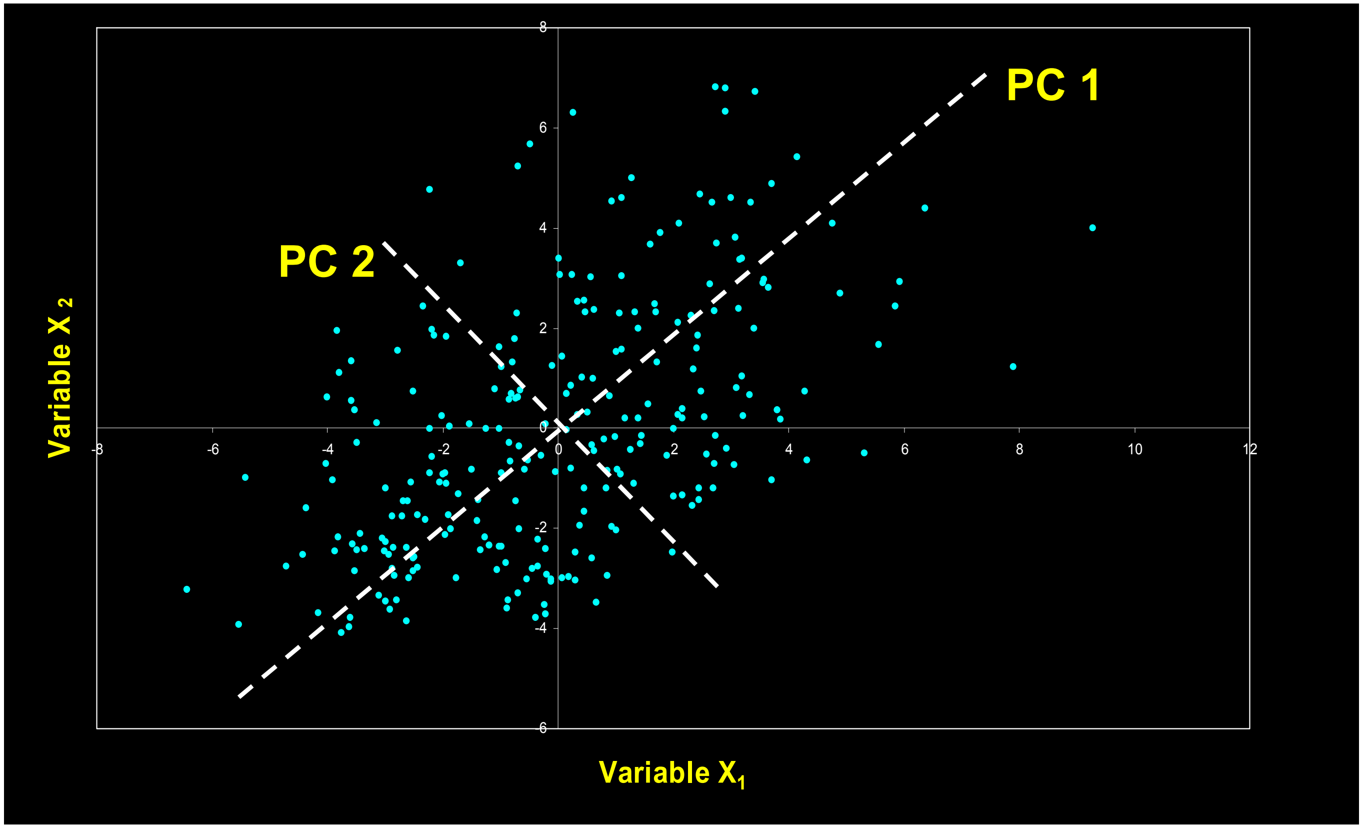
Dimensionality Reduction

Summarization of data with many (d) variables by a smaller set of (k) derived (synthetic, composite) variables.

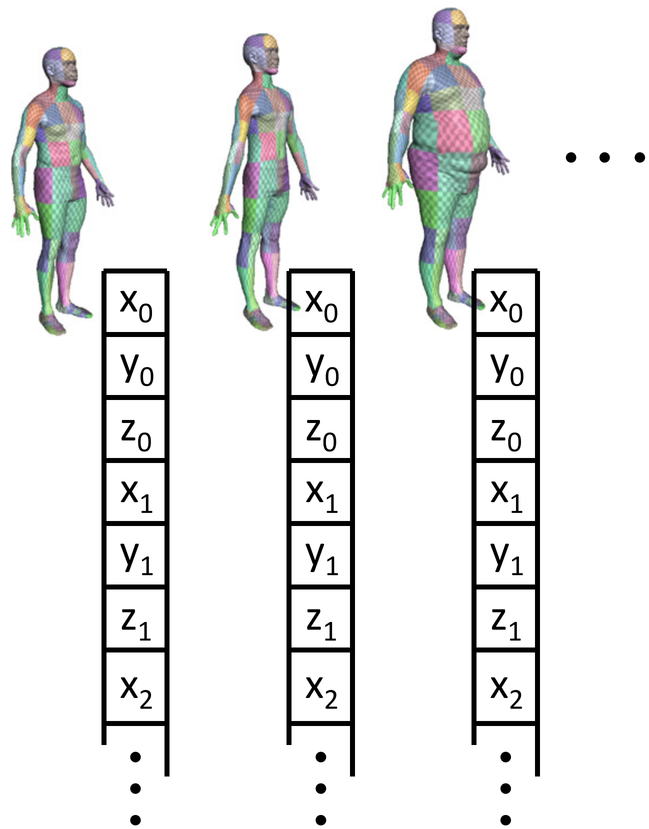


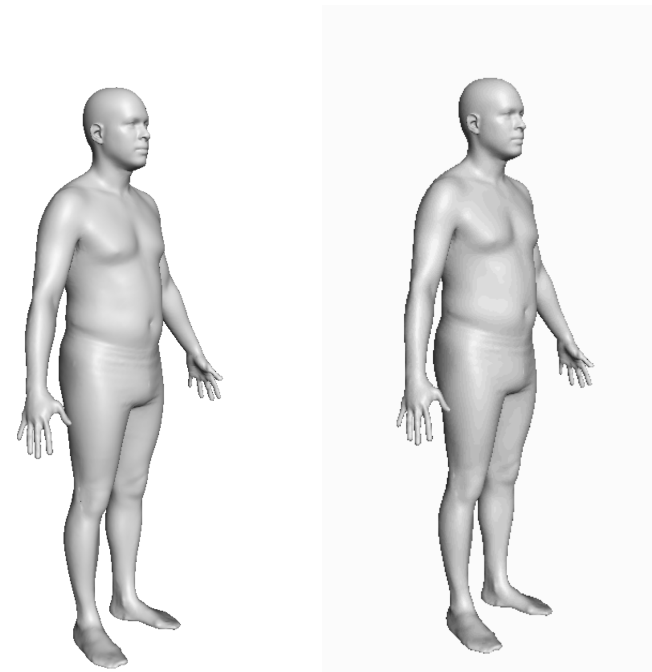
Principal Component Analysis

Each principal axis is a linear combination of the original variables



Principal Component Analysis



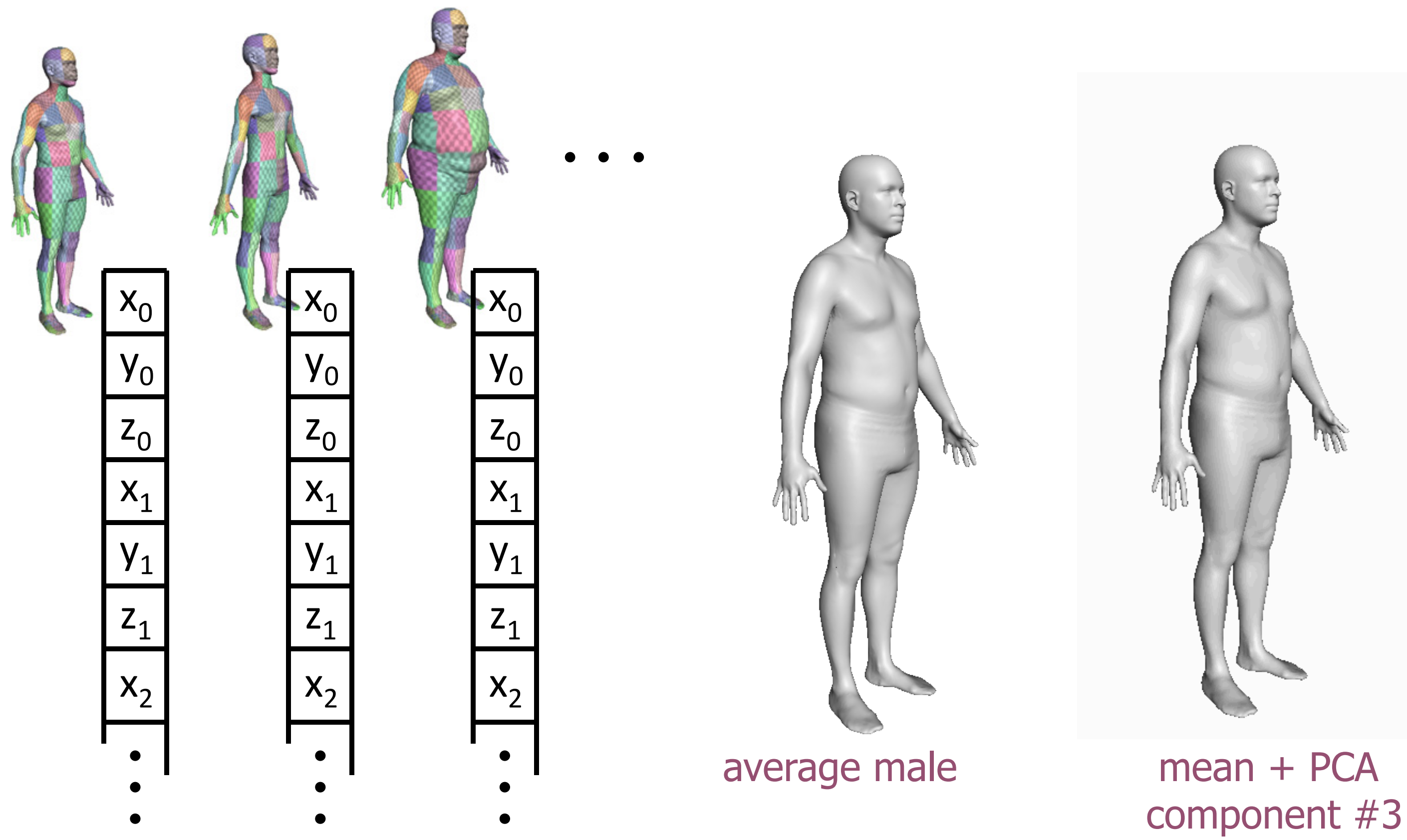


average male

mean + PCA component #1

Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

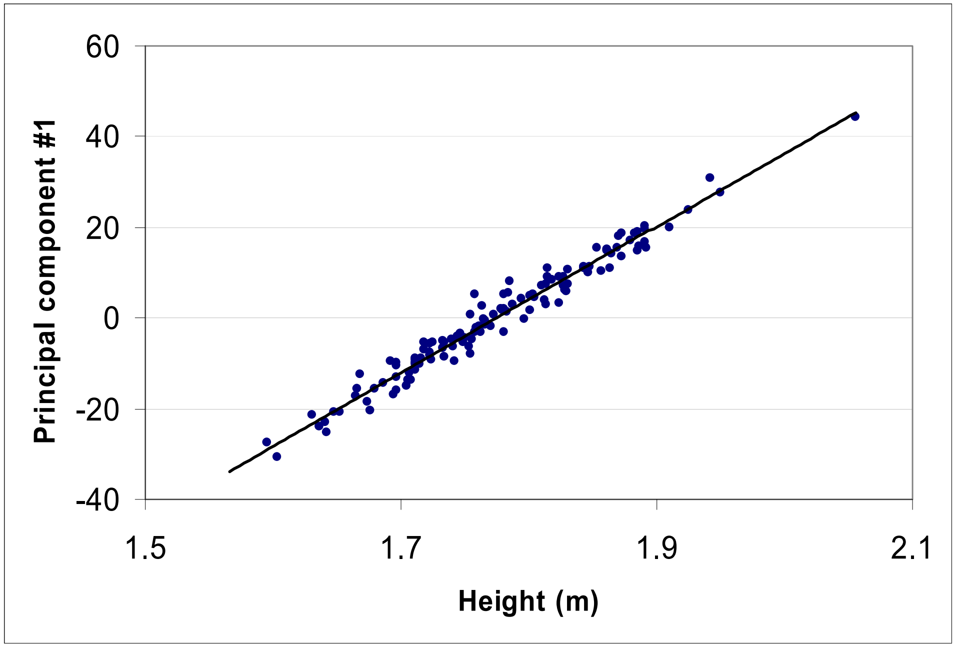
Principal Component Analysis



Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

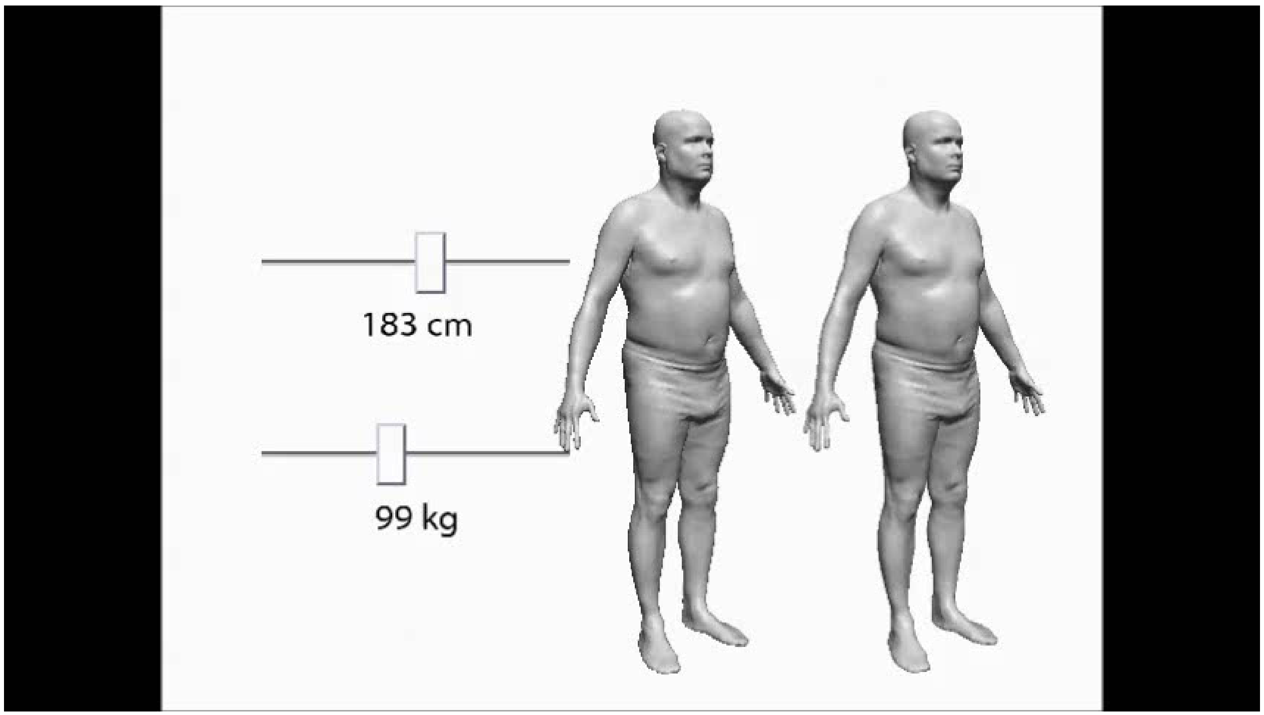
Fitting to attributes

Correlate PCA space with known attributes:



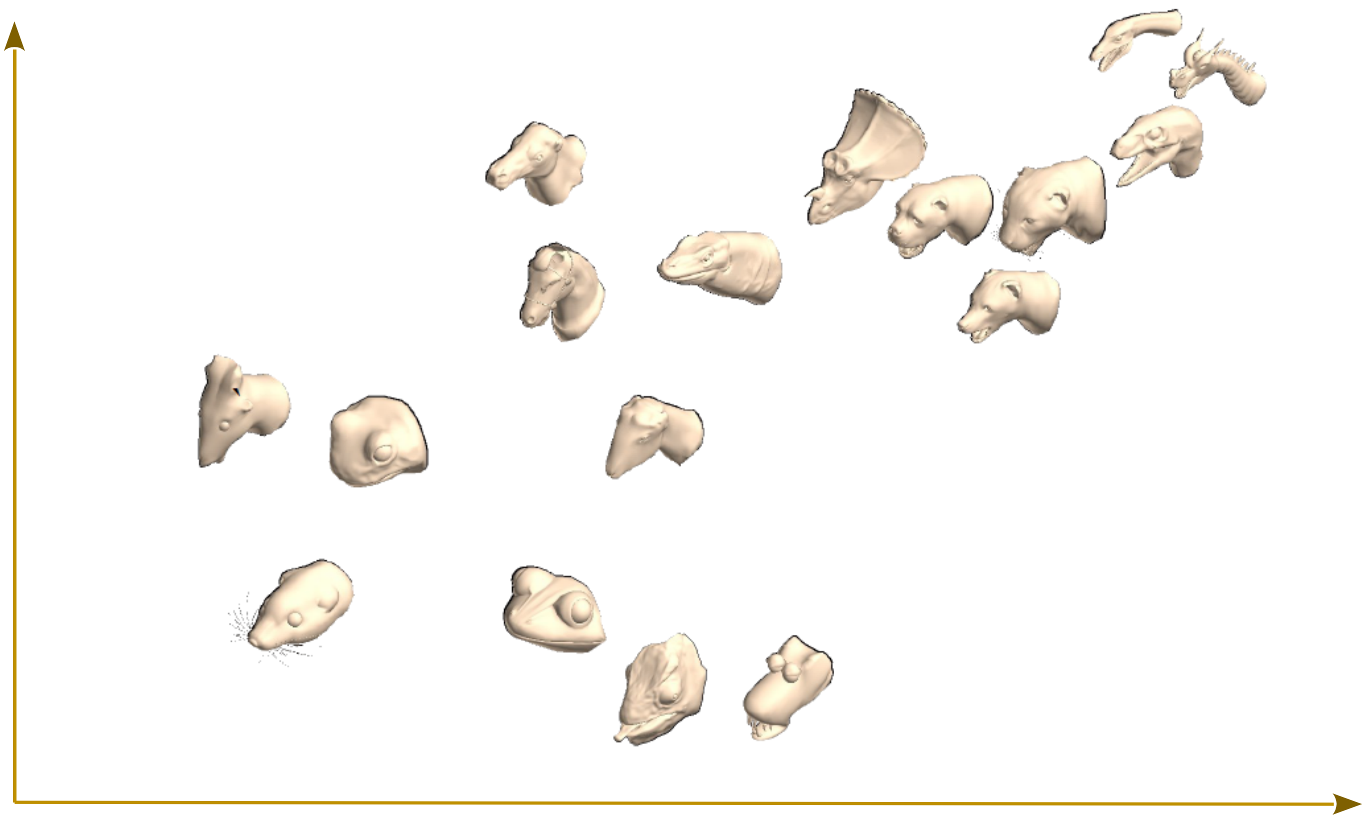
Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

Fitting to attributes



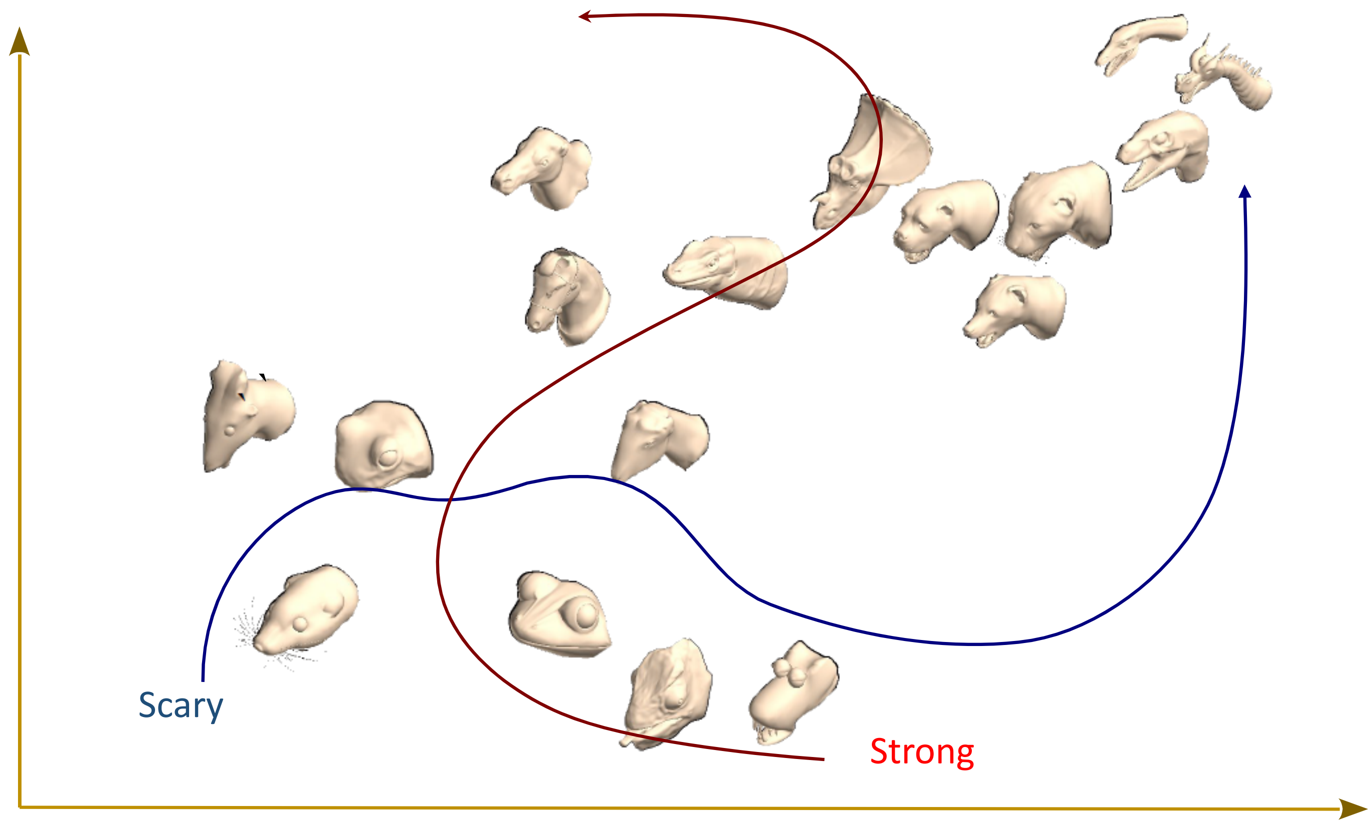
Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

Case study: content creation with semantic attributes



Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

Case study: content creation with semantic attributes



Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

Less aerodynamic ⟷ More aerodynamic

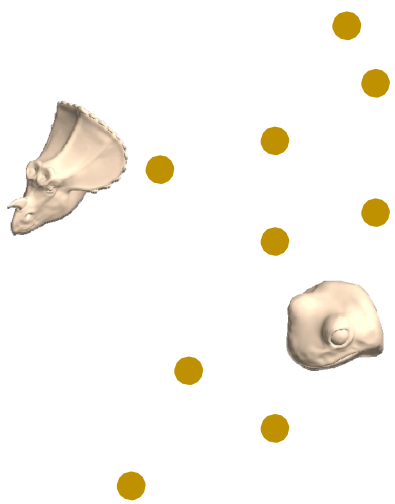
Less scary ⟷ More scary

Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

to access the video: https://www.youtube.com/watch?v=U_XfYzy2c9w

Ranking

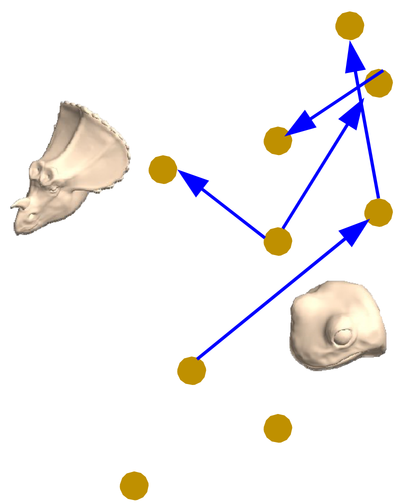
Rank-SVM: Project shape space onto a subspace that best preserves pairwise orderings



Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

Ranking

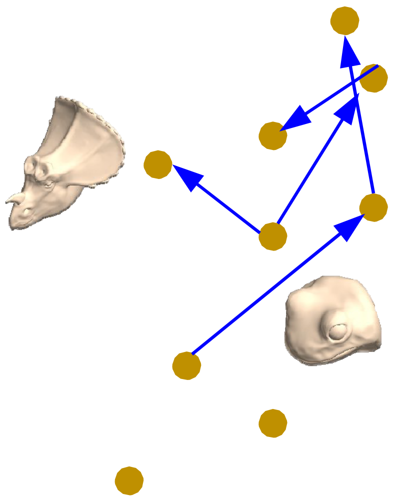
Rank-SVM: Project shape space onto a subspace that best preserves pairwise orderings



Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

Ranking

Rank-SVM: Project shape space onto a subspace that best preserves pairwise orderings



Learn attribute strength:

$$r_m(\mathbf{x}) = \mathbf{w}_m \cdot \mathbf{x}$$

subject to crowdsourced constraints:

$$\forall (i,j) \in O_m : \ \mathbf{w}_m \cdot \mathbf{x}_i > \mathbf{w}_m \cdot \mathbf{x}_j$$
$$\forall (i,j) \in S_m : \ \mathbf{w}_m \cdot \mathbf{x}_i = \mathbf{w}_m \cdot \mathbf{x}_j$$

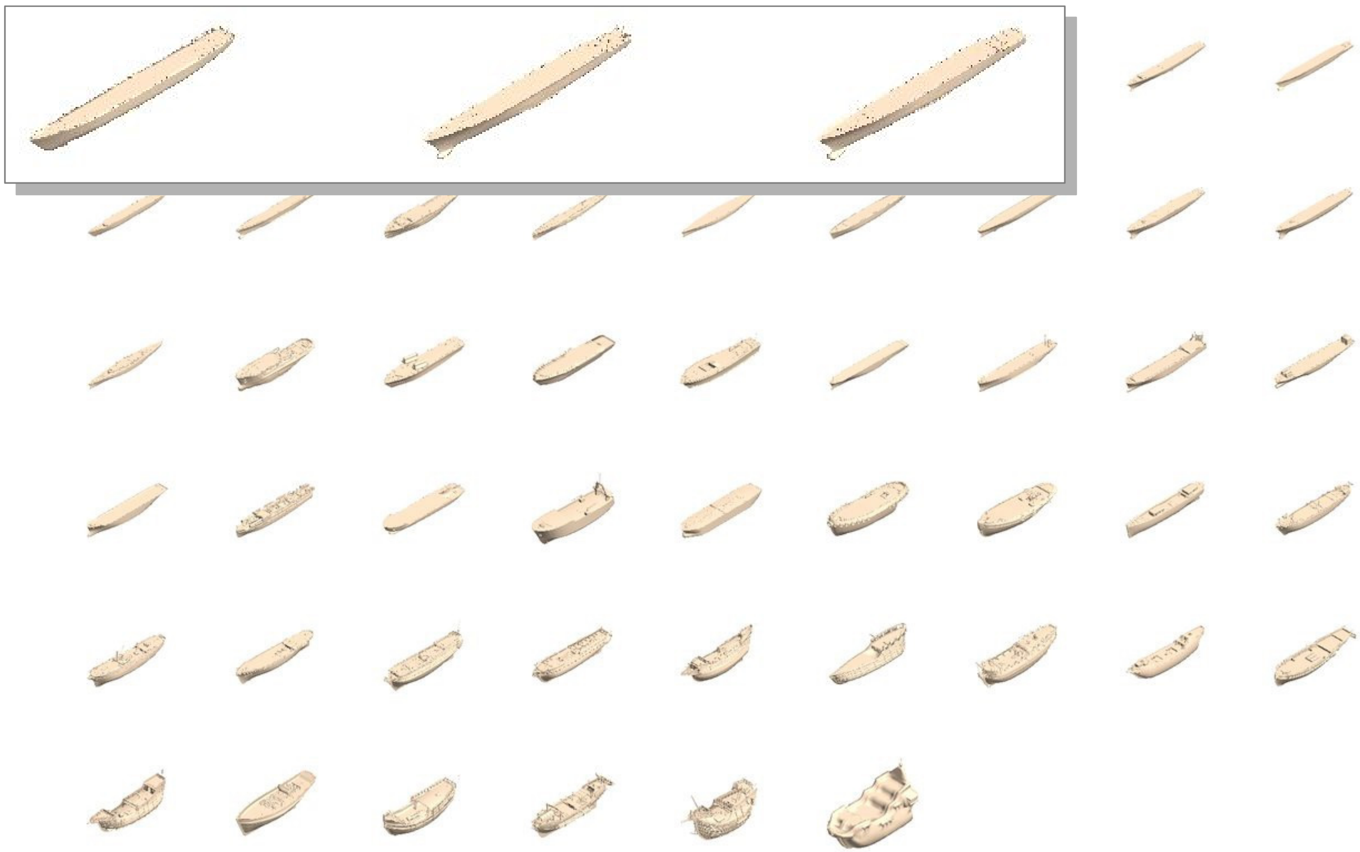
$$\text{minimize} \ \ \|\mathbf{w}_m\|_2^2 + \mu \sum_{i,j \in O_m} c_{ij}(1 - \sigma(\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j))) + \nu \sum_{i,j \in S_m} c_{ij}\sigma(|\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)|)$$

Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

“Old-Fashioned”

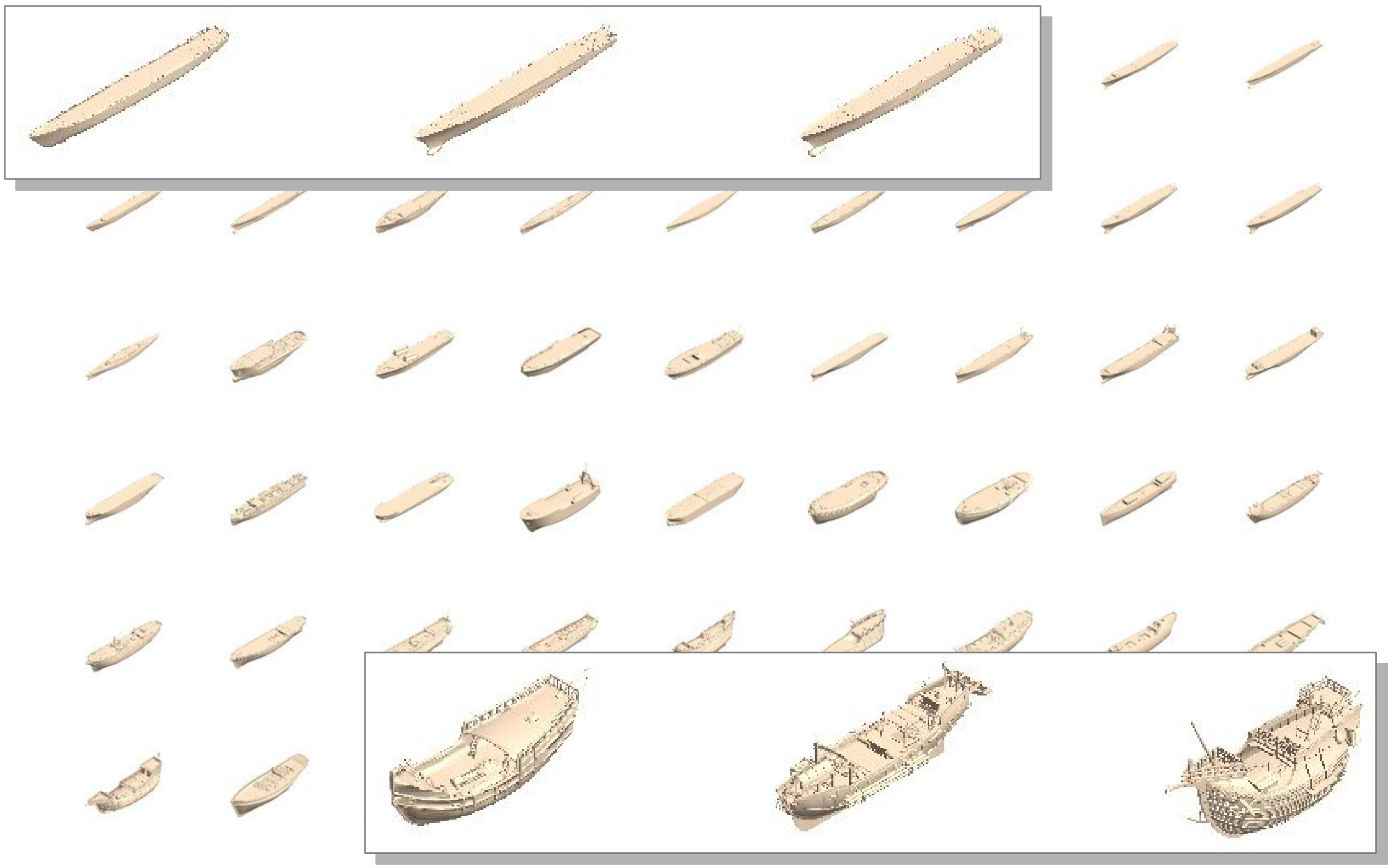
Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

“Old-Fashioned”



Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

"Old-Fashioned"



Slides from Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, Thomas Funkhouser, Content Creation with Semantic Attributes, 2013

Case study: a probabilictic model for component-based synthesis

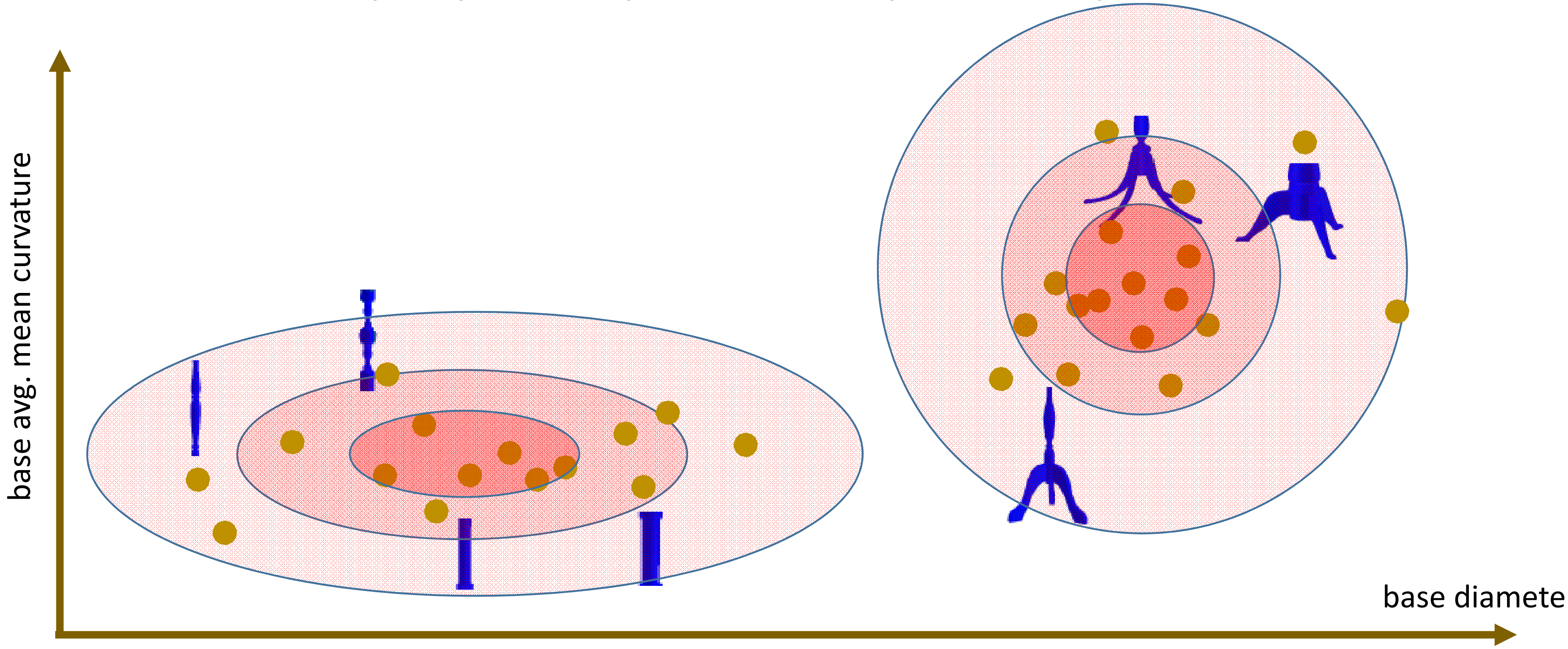
Given some training segmented shapes:

… and more ….

Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Case study: a probabilictic model for component-based synthesis

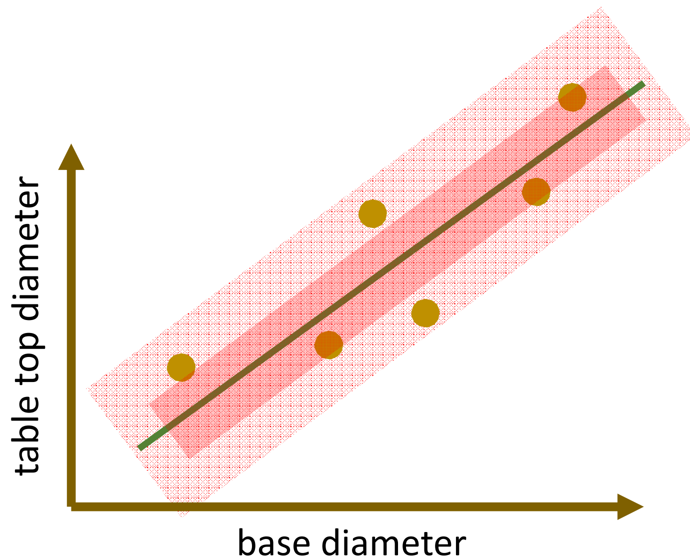
Describe shape space of parts with a probability distribution



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Case study: a probabilictic model for component-based synthesis

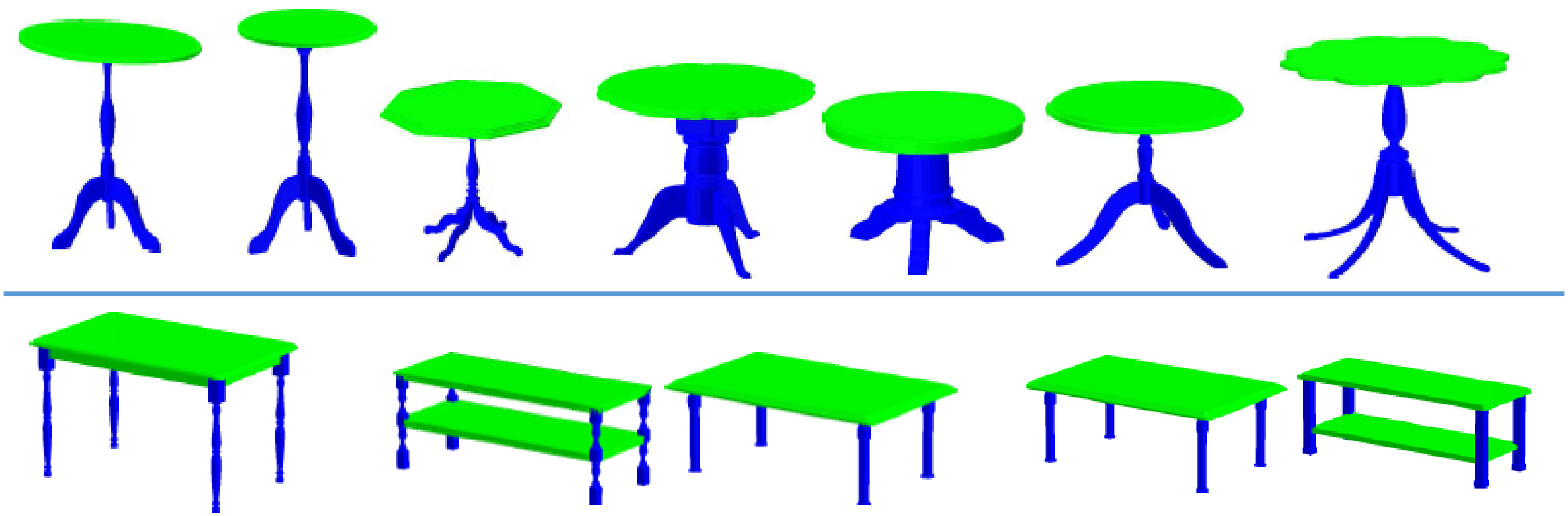
Learn relationships between different part parameters within each cluster
e.g. diameter of table top is related to scale of base plus some uncertainty



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Case study: a probabilictic model for component-based synthesis

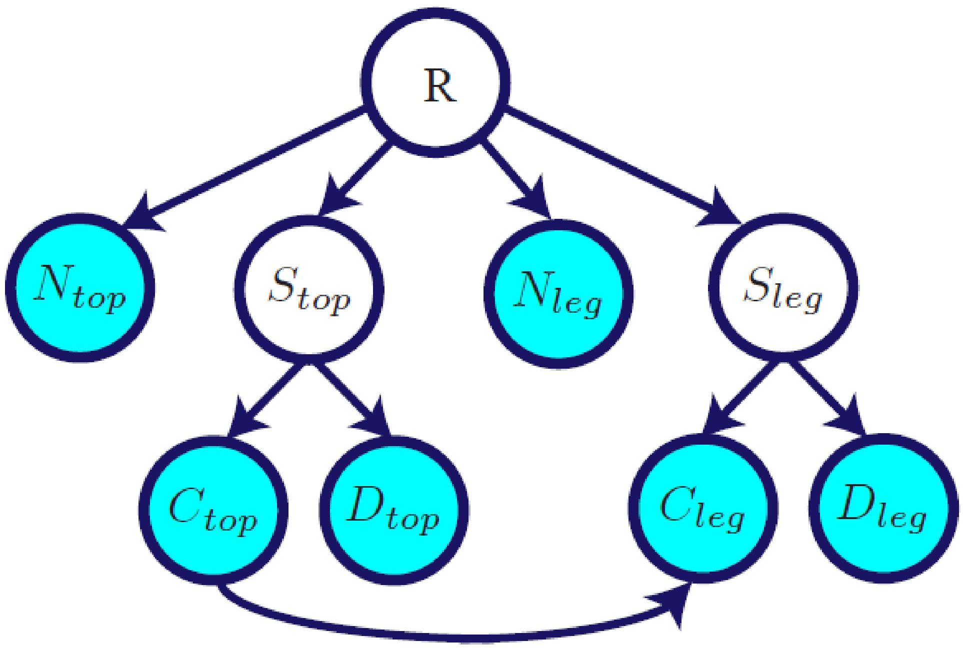
Learn relationships between part clusters e.g. circular table tops are associated with bases with split legs



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

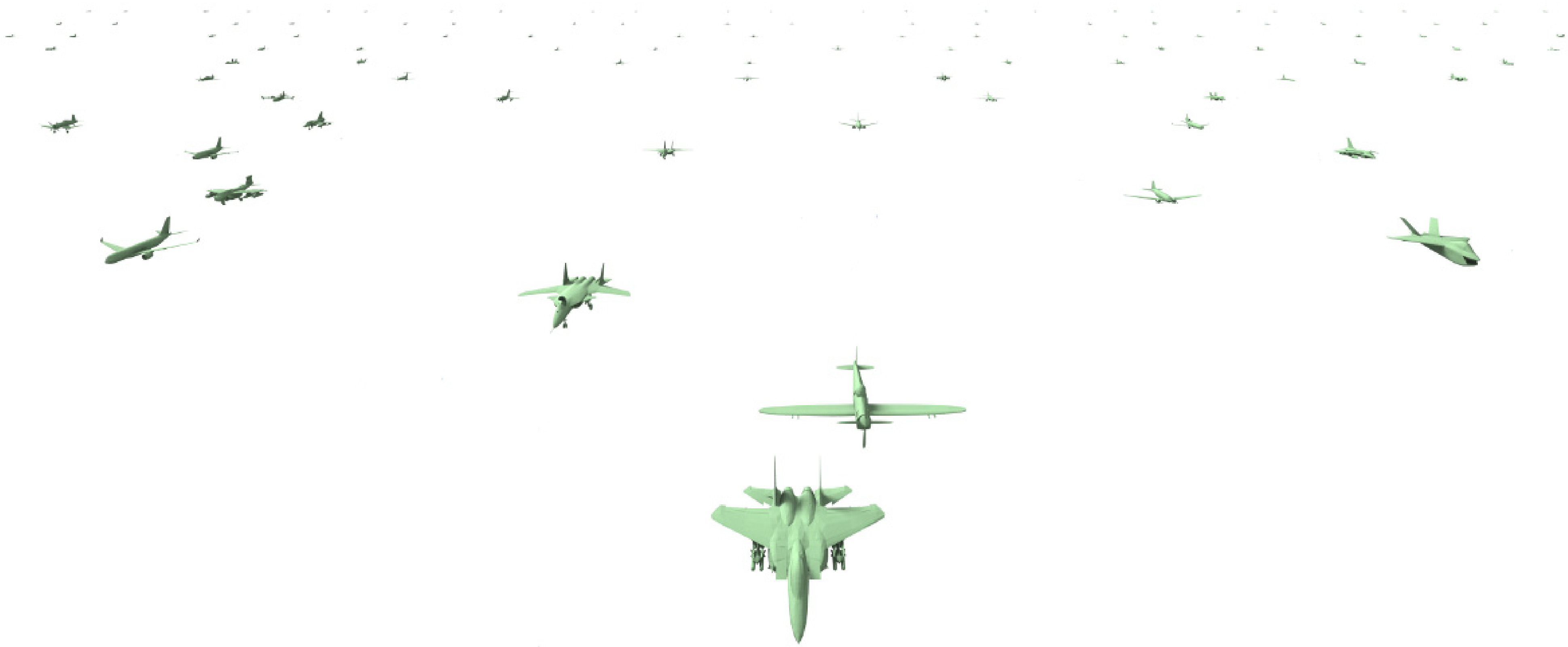
Case study: a probabilictic model for component-based synthesis

Represent all these relationships within a structured probability distribution (probabilistic graphical model)



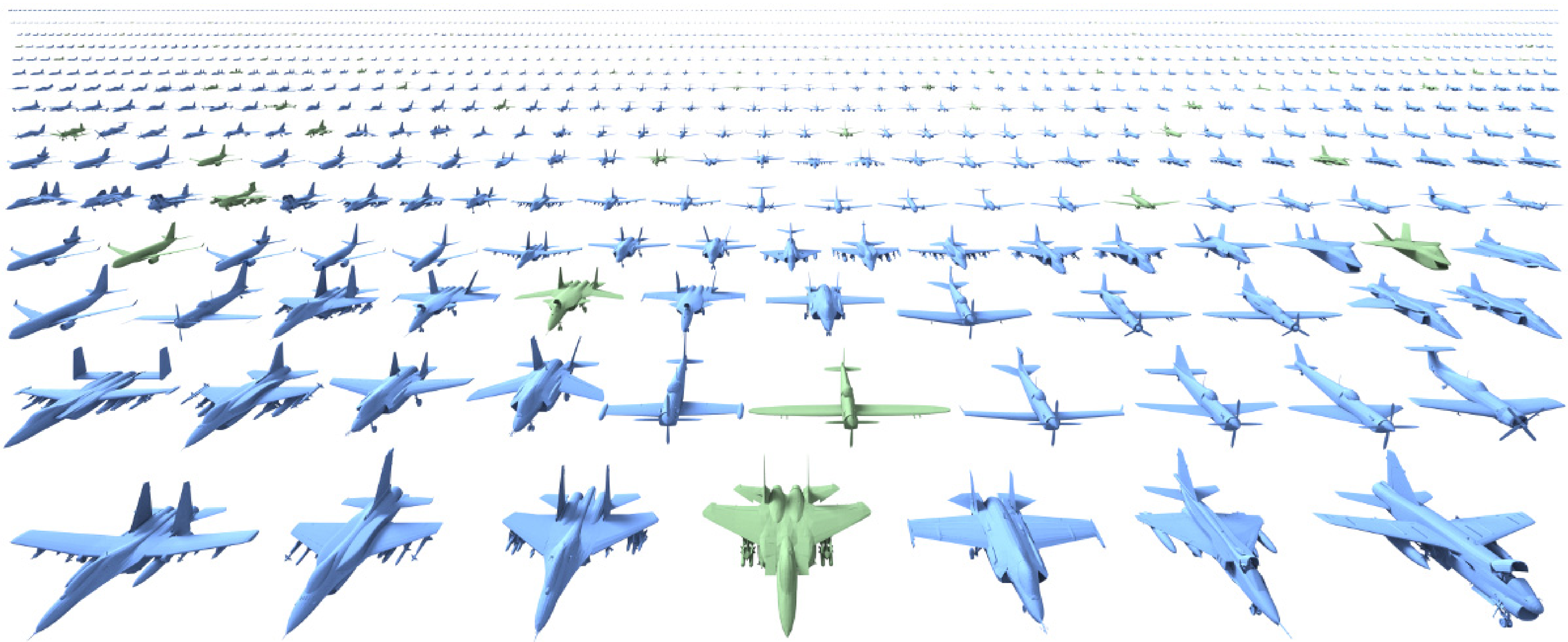
Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Airplanes



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Airplanes

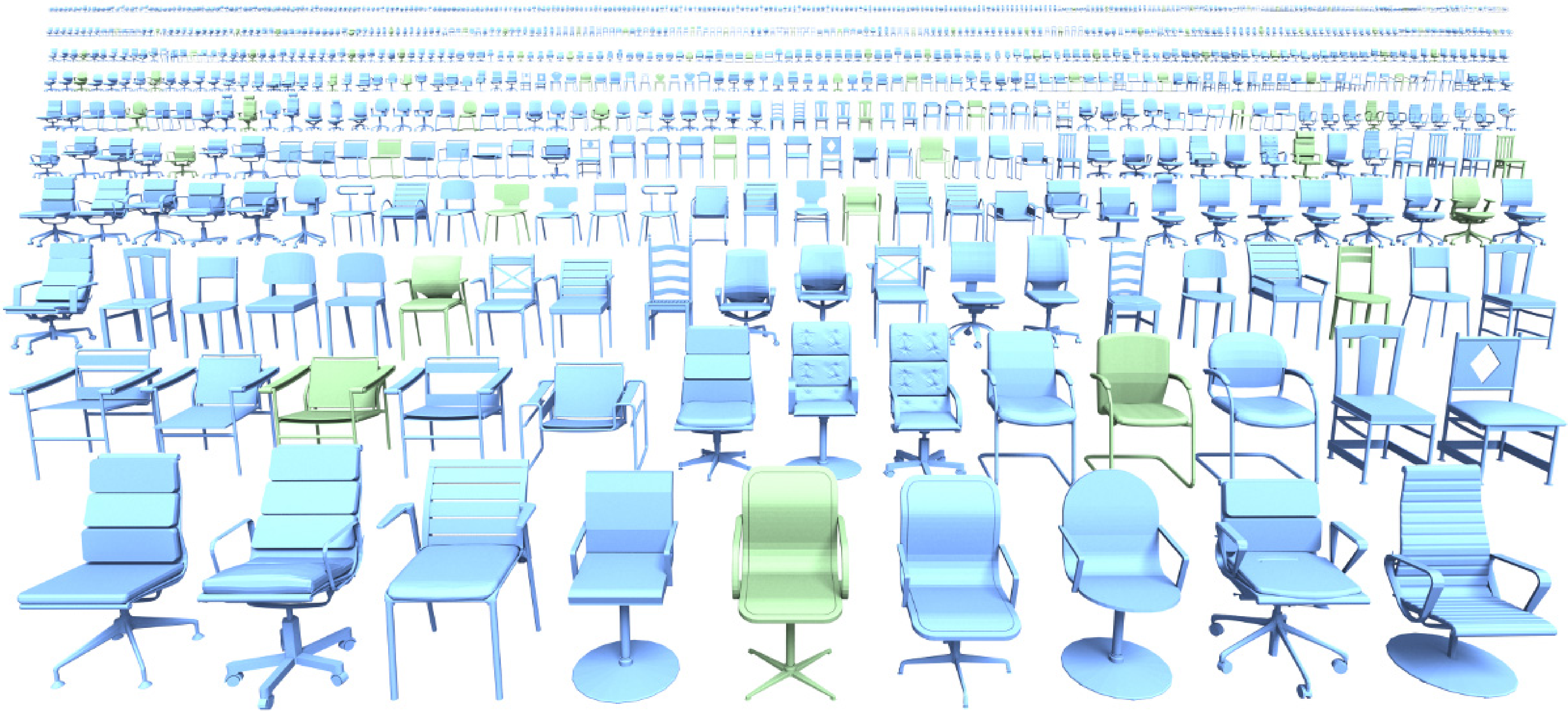


Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Chairs

Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun, A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Chairs

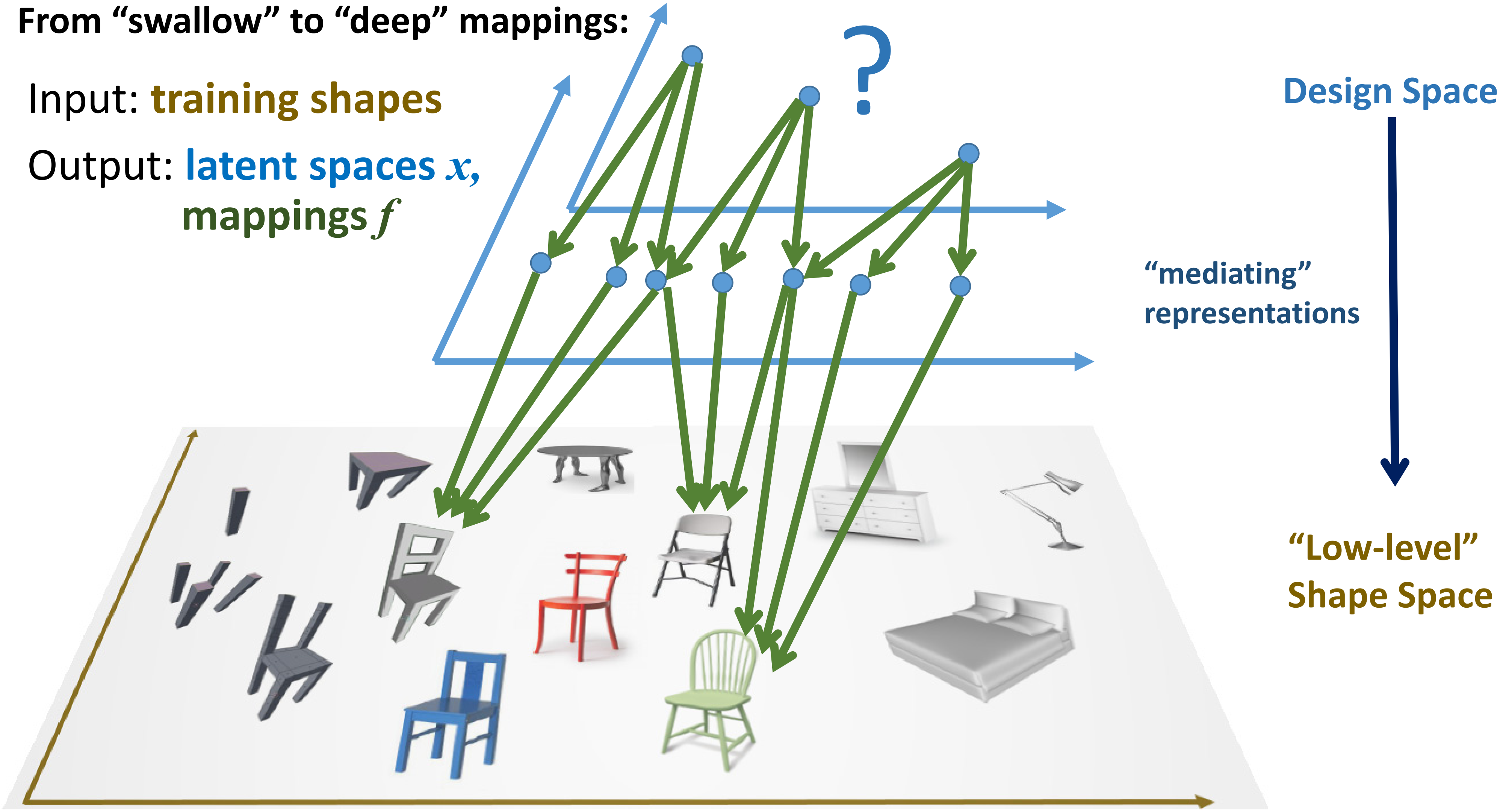


Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun, A Probabilistic Model for Component-Based Synthesis, 2012

From “swallow” to “deep” mappings:

Input: training shapes

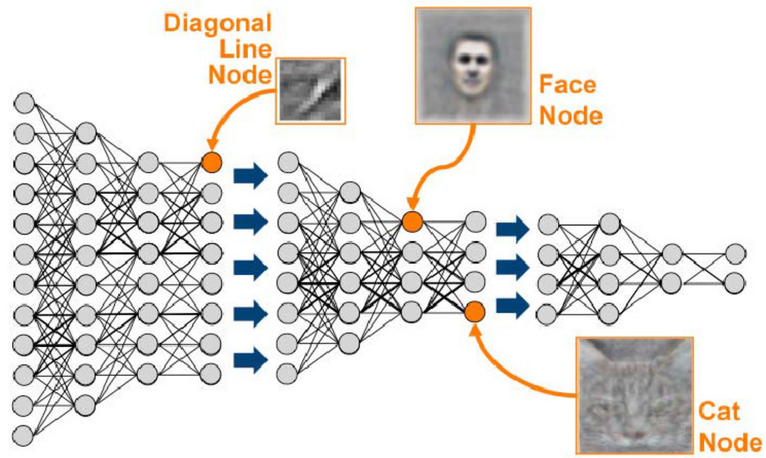
Output: latent spaces x,
mappings f



From “swallow” to “deep” mappings (networks)

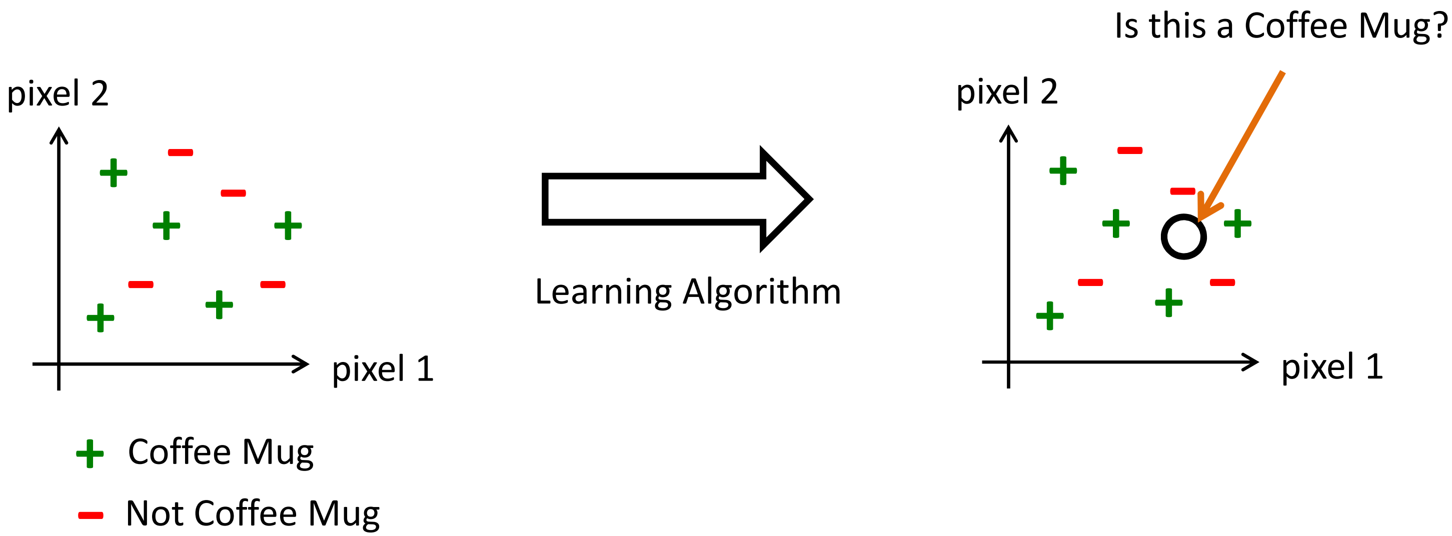
Images, shapes, natural language have **compositional structure**

Deep neural networks!



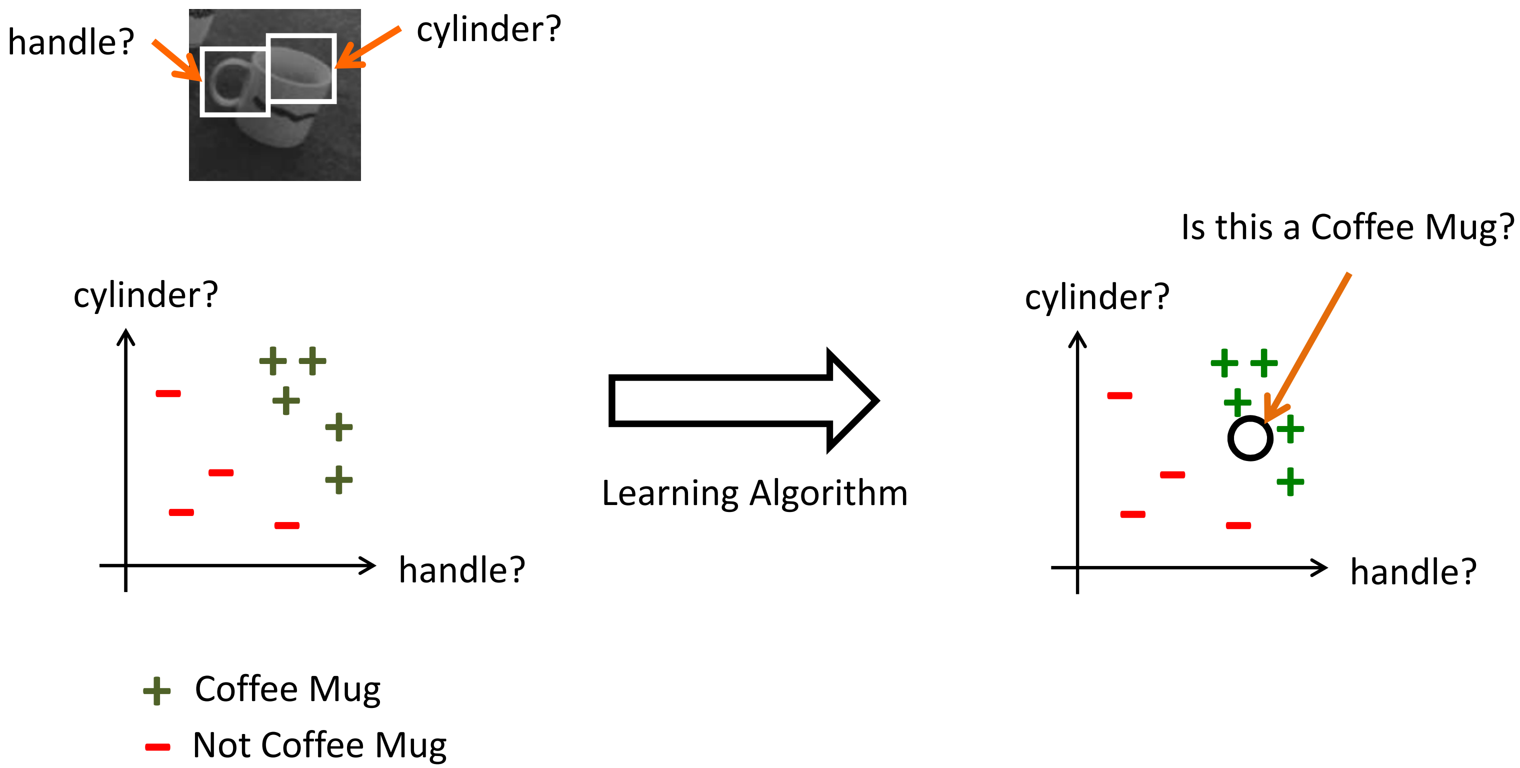
Note: Let’s discuss them in the case of 2D images for now! Also let’s map from images to high-level representations. We’ll see how this can be reversed later.

Motivation



modified slides originally by Adam Coates

Motivation



modified slides originally by Adam Coates

"Traditional" recognition pipeline

Fixed/engineered descriptors + **trained** classifier/regressor

"Hand-engineered" Descriptor Extractor e.g. SIFT, bags-of-words → **Trained classifier/regressor** → **car?**

modified slides originally by Adam Coates

"New" recognition pipeline

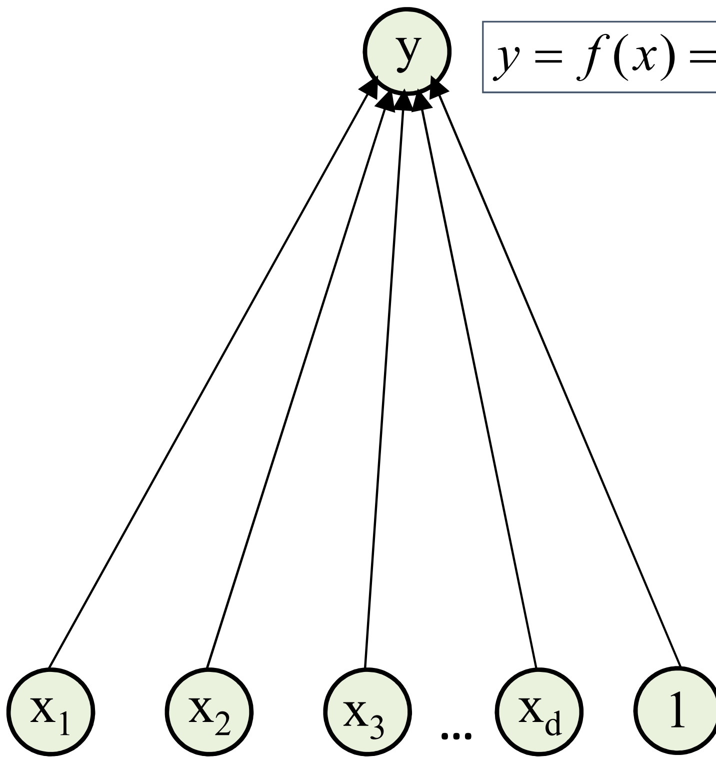
Trained descriptors + **trained** classifier/regressor

modified slides originally by Adam Coates

From “swallow” to “deep” mappings (networks)

In logistic regression, output was a direct function of inputs. Conceptually, this can be thought of as a network:

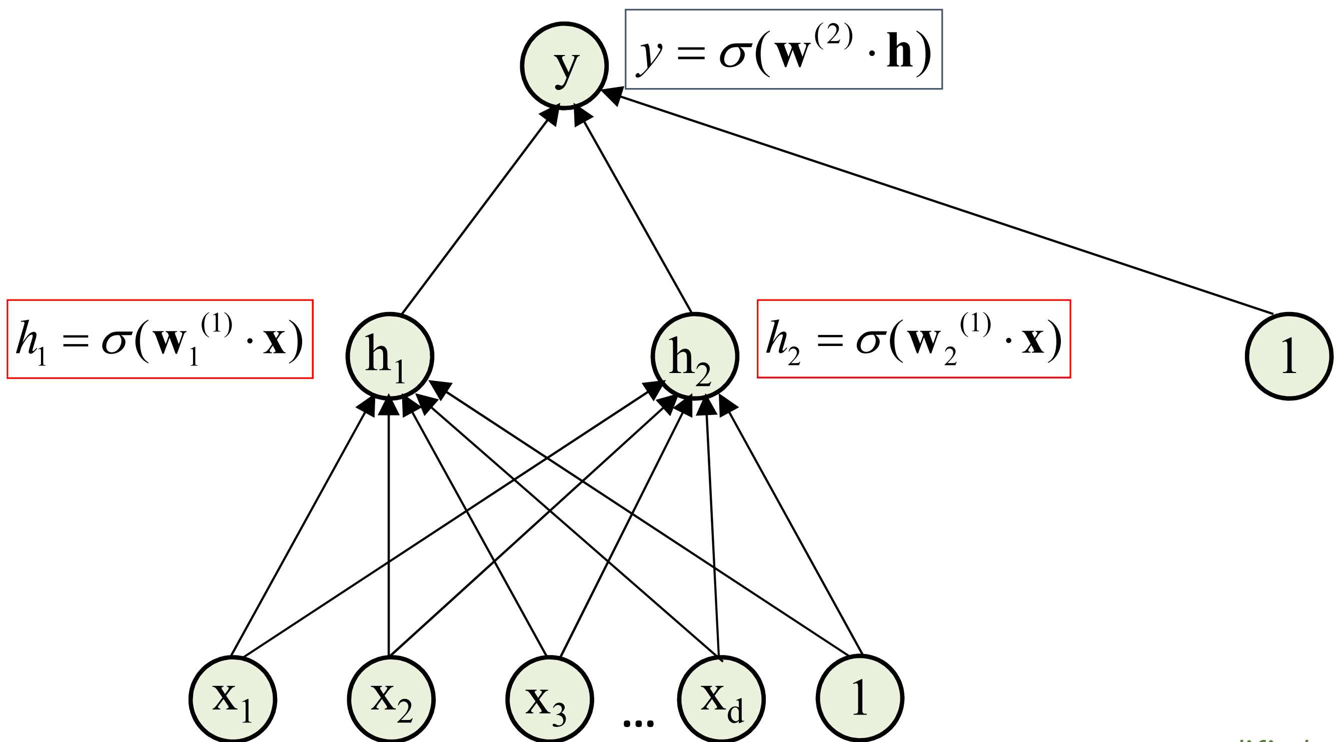
$$y = f(x) = \sigma(\mathbf{w} \cdot \mathbf{x})$$



modified slides originally by Adam Coates

Basic idea

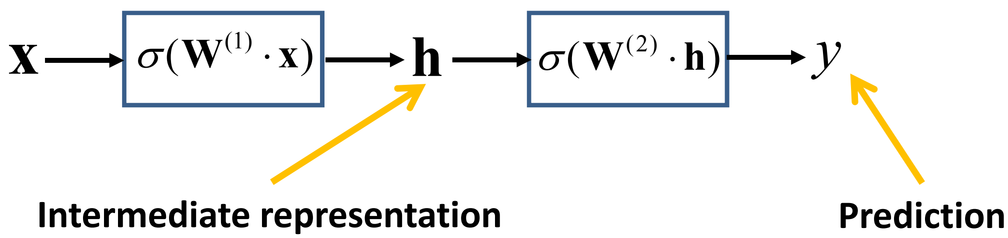
Introduce latent nodes that will play the role of **learned representations.**



modified slides originally by Adam Coates

Neural network

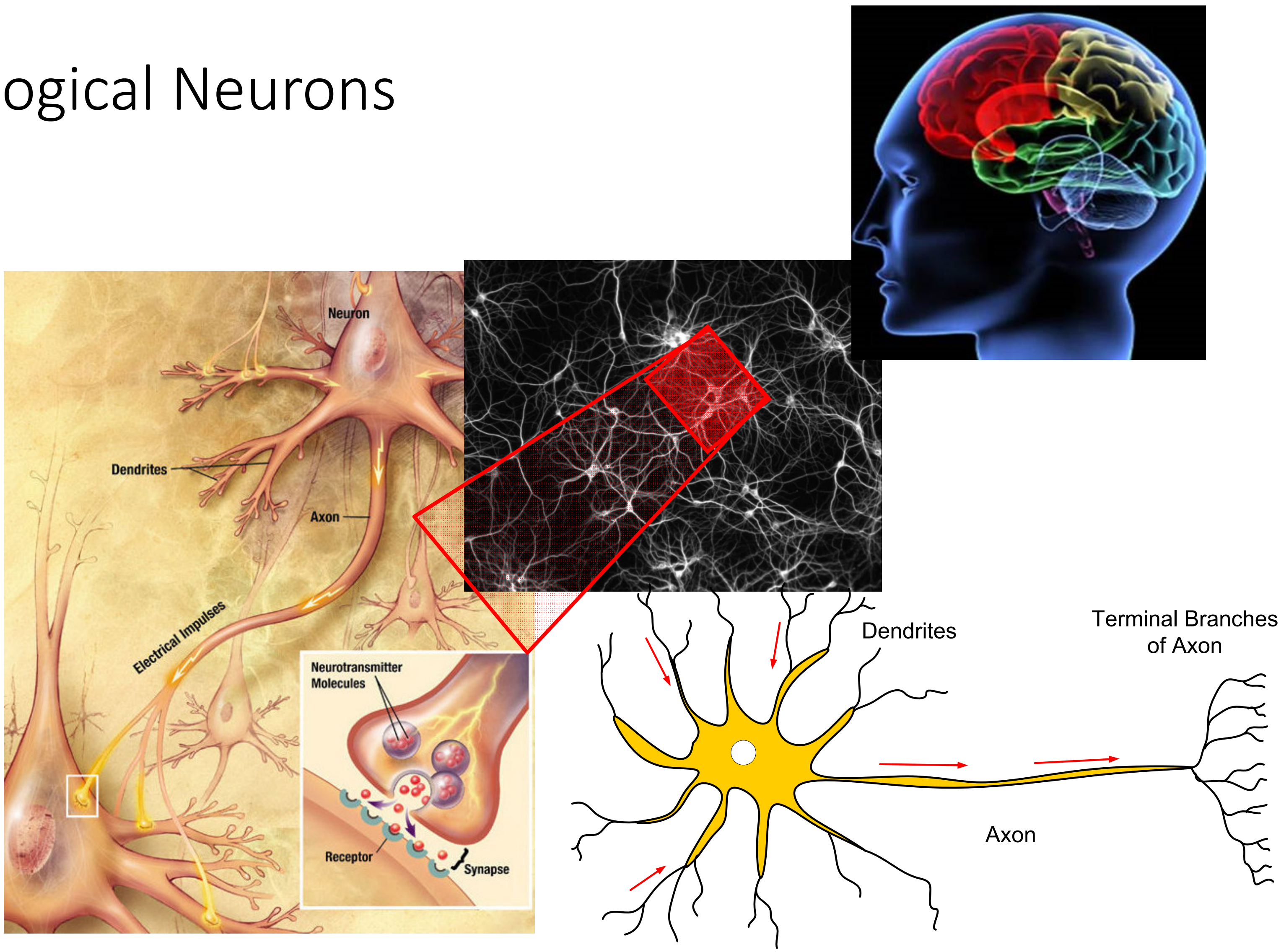
Same as logistic regression but now our output function has multiple stages ("layers", "modules").



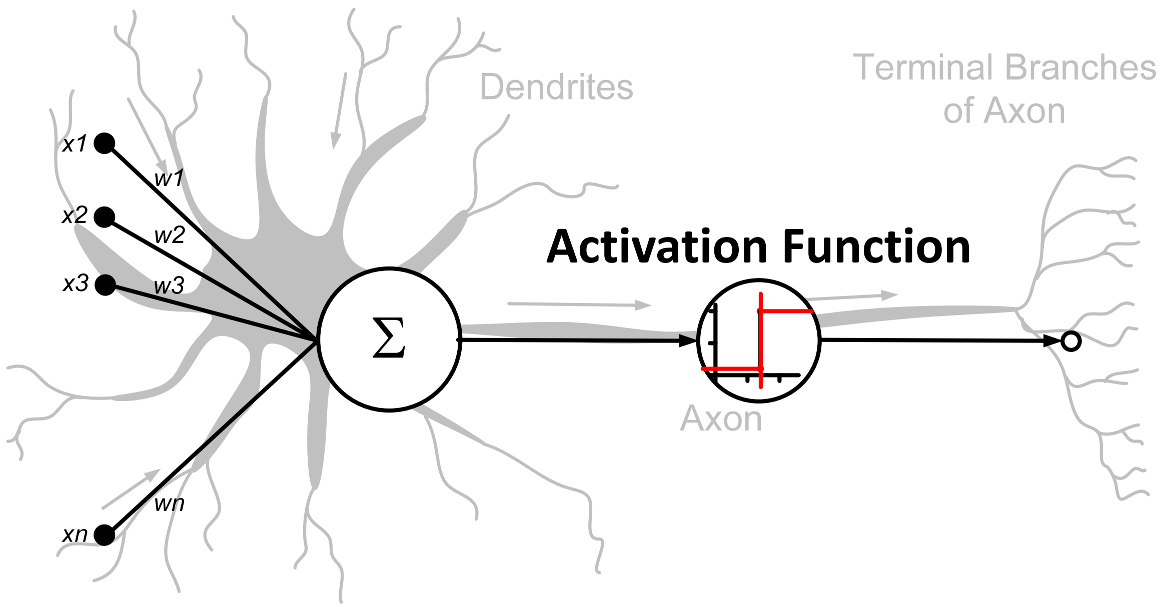
$$\textit{where} \quad \mathbf{W}^{(\cdot)} = \begin{bmatrix} \mathbf{w_1}^{(\cdot)} \\ \mathbf{w_2}^{(\cdot)} \\ \cdots \\ \mathbf{w}_m^{(\cdot)} \end{bmatrix}$$

modified slides originally by Adam Coates

Biological Neurons



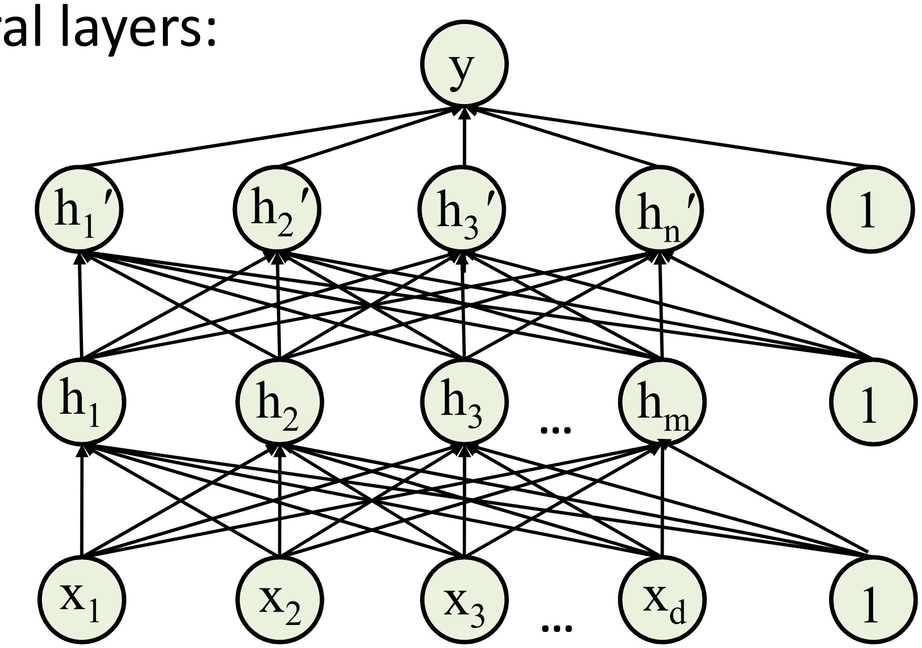
Analogy with biological networks



Slide credit : Andrew L. Nelson

Neural network

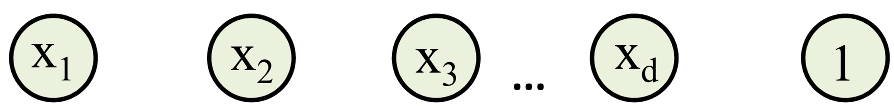
Stack up several layers:



modified slides originally by Adam Coates

Forward propagation

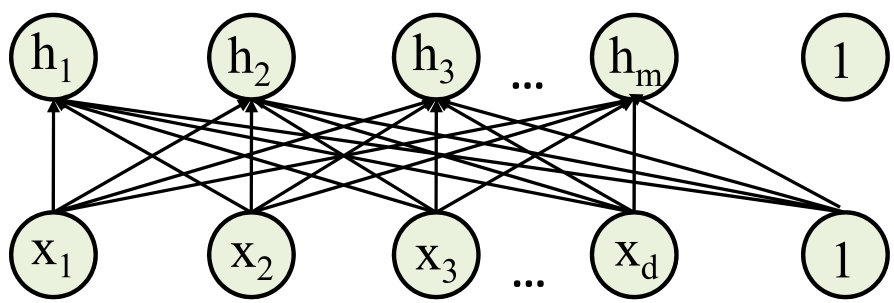
Process to compute output:



modified slides originally by Adam Coates

Forward propagation

Process to compute output:

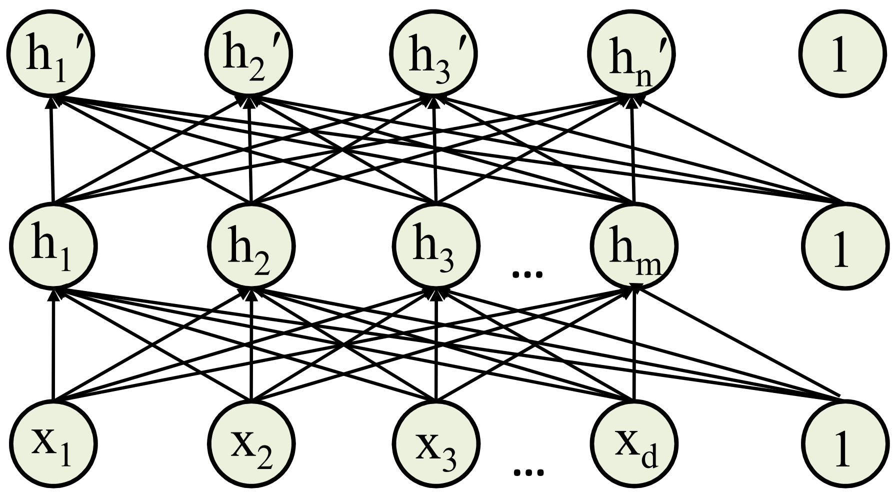


$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h}$$

modified slides originally by Adam Coates

Forward propagation

Process to compute output:

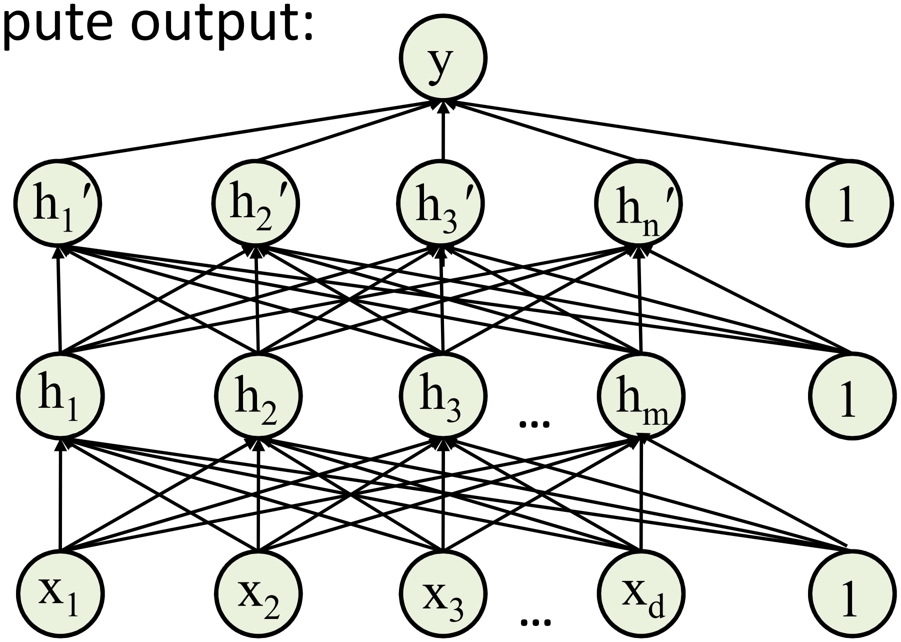


$$\mathbf{x} \rightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \rightarrow \mathbf{h} \rightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \rightarrow \mathbf{h}'$$

modified slides originally by Adam Coates

Forward propagation

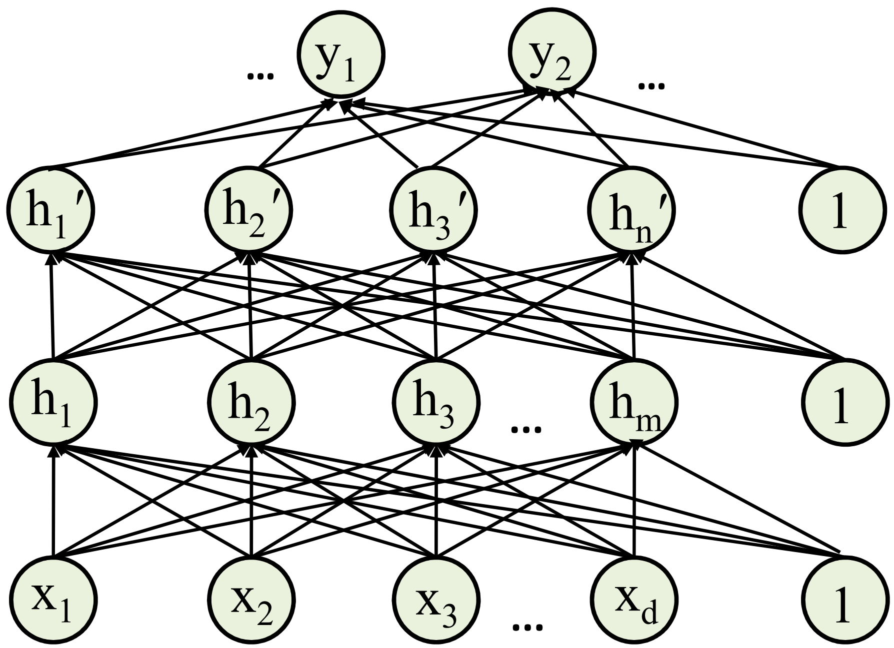
Process to compute output:



$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h} \longrightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \longrightarrow \mathbf{h}' \longrightarrow \boxed{\sigma(\mathbf{W}^{(3)} \cdot \mathbf{h}')} \longrightarrow y$$

modified slides originally by Adam Coates

Multiple outputs



$$\mathbf{x} \rightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \rightarrow \mathbf{h} \rightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \rightarrow \mathbf{h}' \rightarrow \boxed{\sigma(\mathbf{W}^{(3)} \cdot \mathbf{h}')} \rightarrow \mathbf{y}$$

modified slides originally by Adam Coates

How can you learn the parameters?

Use a loss function e.g., for classification:

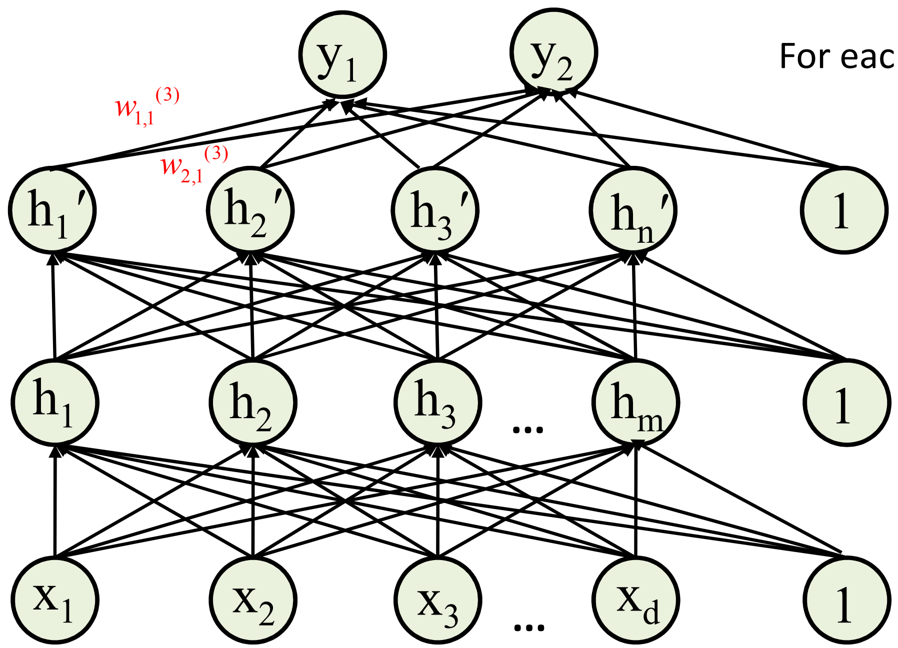
$$L(\mathbf{w}) = -\sum_{i=1} \sum_{output\ t} [\mathbf{y}_{\mathbf{i,t}} == 1] \log f_t(\mathbf{x}_i) + [\mathbf{y}_{\mathbf{i,t}} == 0] \log(1 - f_t(\mathbf{x}_i))$$

For regression:

$$L(\mathbf{w}) = \sum_{i} \sum_{output\ t} [\mathbf{y}_{i,t} - f_t(\mathbf{x}_i)]^2$$

Backpropagation

For each training example i (omit index i for clarity):

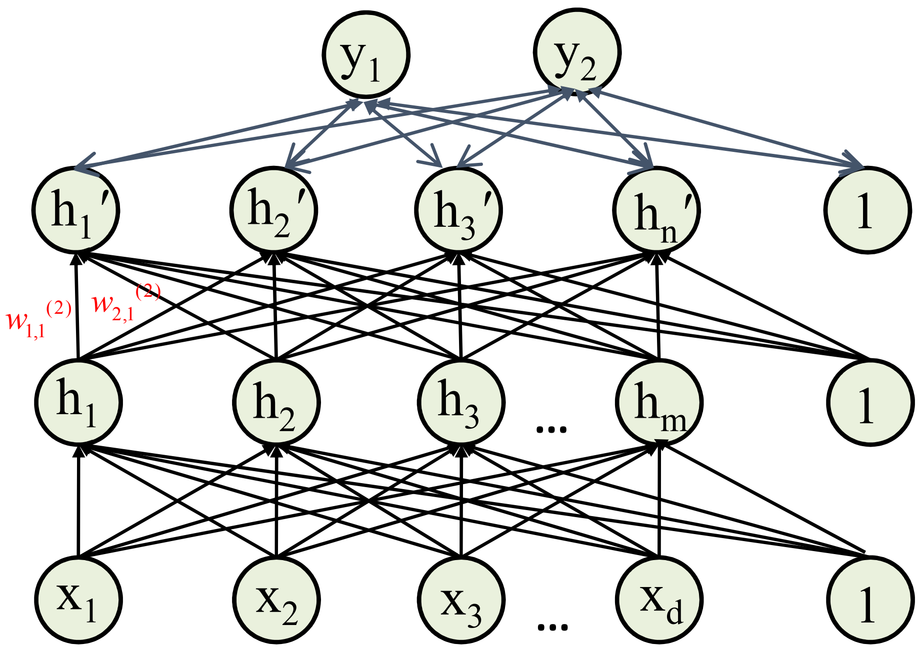


For each output: $\delta_t^{(3)} = f_t(\mathbf{x}) - y_t$

$$\frac{\partial L(\mathbf{w})}{\partial w_{t,n}^{(3)}} = \delta_t^{(3)} h_n$$

Backpropagation

For each training example i (omit index i for clarity):



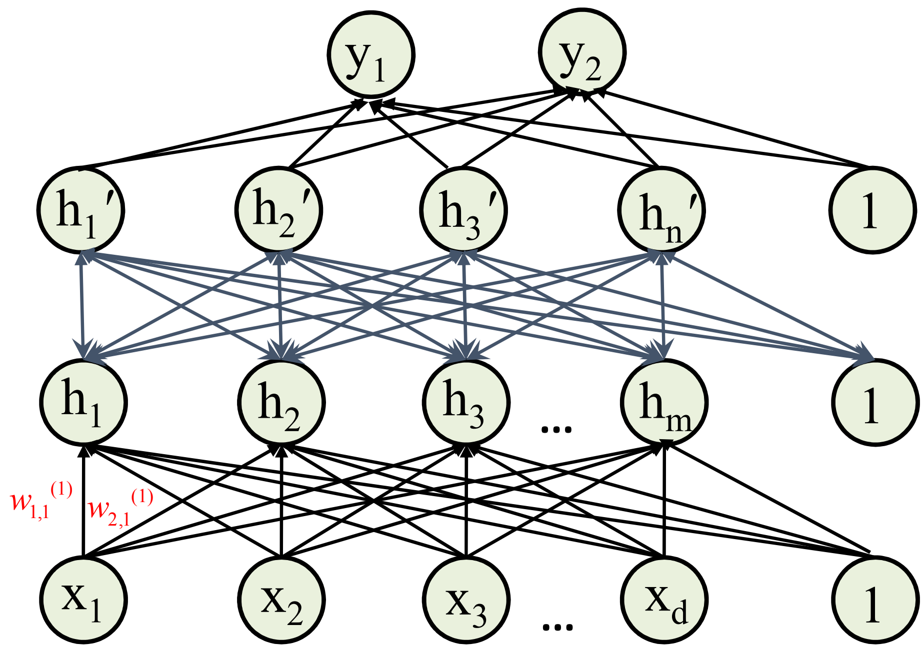
$$\delta_n^{(2)} = \sigma'(\mathbf{w_n}^{(2)} \cdot \mathbf{h}) \sum_t w_{t,n}^{(3)} \delta_t^{(3)}$$

Note: $\sigma'(\cdot) = \sigma(\cdot)[1 - \sigma(\cdot)]$

$$\frac{\partial L(\mathbf{w})}{\partial w_{n,m}^{(2)}} = \delta_n^{(2)} h_m$$

Backpropagation

For each training example i (omit index i for clarity):



$$\delta_m^{(1)} = \sigma'(\mathbf{w}_m^{(1)} \cdot \mathbf{x}) \sum_n w_{n,m}^{(2)} \delta_n^{(2)}$$

$$\frac{\partial L(\mathbf{w})}{\partial w_{m,d}^{(1)}} = \delta_m^{(1)} x_d$$

Is this magic?

All these are derivatives derived analytically using the **chain rule**!

Gradient descent is expressed through **backpropagation of messages** δ following the structure of the model

Training algorithm

For each training example [in a batch]

1. **Forward propagation** to compute outputs per layer
2. **Back propagate** messages δ from top to bottom layer
3. Multiply messages δ with inputs to compute **derivatives per layer**
4. **Accumulate the derivatives** from that training example

Apply the gradient descent rule

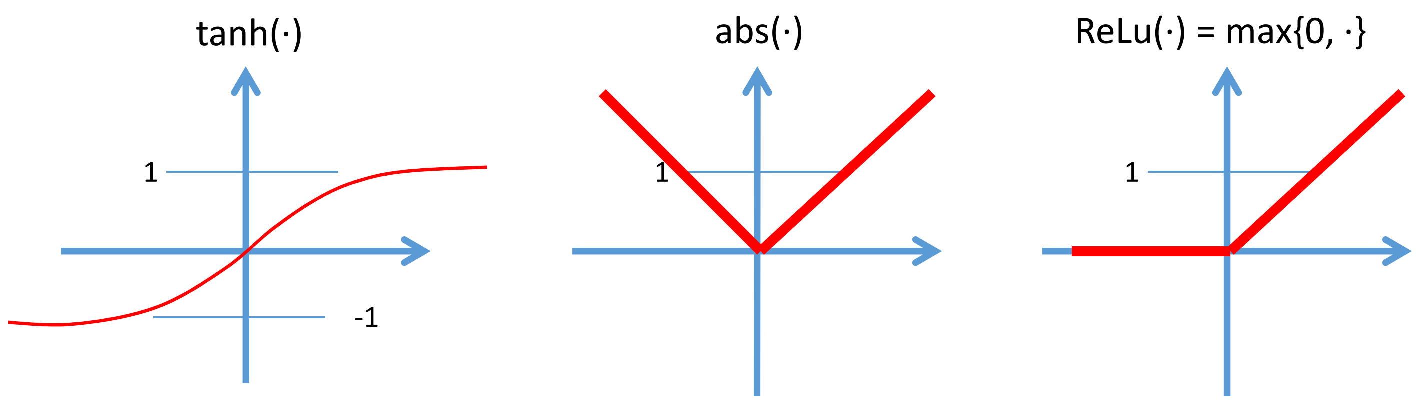
Yet, this does not work so easily...

Yet, this does not work so easily...

- **Non-convex:** Local minima; convergence criteria.
- Optimization becomes difficult with **many layers**.
- Hard to diagnose and **debug malfunctions**.
- **Many things turn out to matter**:
  - Choice of nonlinearities.
  - Initialization of parameters.
  - Optimizer parameters: step size, schedule.

Non-linearities

- **Choice of functions inside network matters.**
  - Sigmoid function yields highly non-convex loss functions
  - Some other choices often used:



$tanh'(\cdot) = 1 - tanh(\cdot)^2$

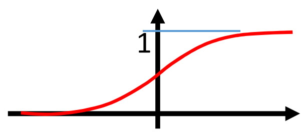
$abs'(\cdot) = sign(\cdot)$

$ReLu'(\cdot) = [\cdot > 0]$

"Rectified Linear Unit"
→ Increasingly popular.
[Nair & Hinton, 2010]

Initialization

- **Usually small random values.**
  - Try to choose so that typical input to a neuron avoids saturating



- **Initialization schemes for weights used as input to a node:**
  - tanh units: Uniform[-r, r]; sigmoid: Uniform[-4r, 4r].
  - See [Glorot et al., AISTATS 2010]

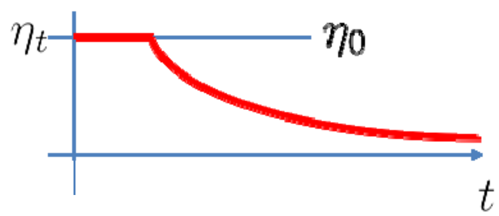
$$r = \sqrt{6/(\text{fan-in} \ + \ \text{fan-out})}$$

- **Unsupervised pre-training**

Step size

- **Fixed step-size**
  - try many, choose the best...
  - pick size with least test error on a validation set after T iterations

- **Dynamic step size**
  - decrease after T iterations



  - if simply the objective is not decreasing much, cut step by half

Momentum

Modify stochastic/batch gradient descent:

$$\text{Before}: \quad \Delta\mathbf{w} = \eta \nabla_{\mathbf{w}} L(\mathbf{w}), \quad w = w - \Delta\mathbf{w}$$

$$\text{With momentum}: \Delta\mathbf{w} = \mu \Delta\mathbf{w}_{previous} + \eta \nabla_{\mathbf{w}} L(\mathbf{w}), \quad w = w - \Delta\mathbf{w}$$

"Smooth" estimate of gradient from several steps of gradient descent:

- High-curvature directions cancel out.
- Low-curvature directions "add up" and accelerate.

Regularize!

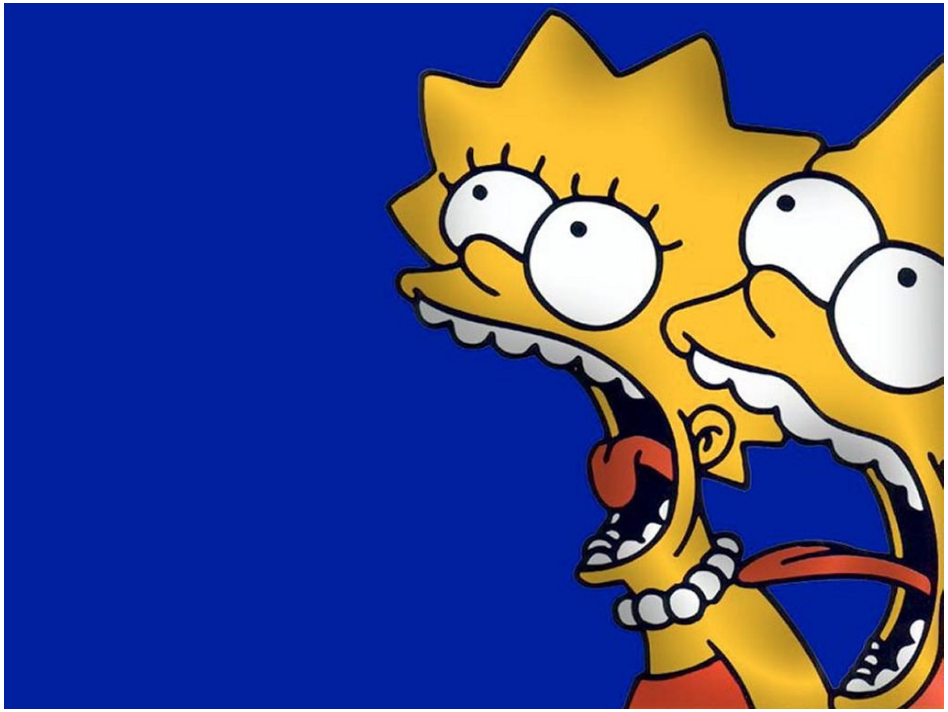
- Adding **L2 regularization** term to the loss function:

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} (L(\mathbf{w}) + \lambda \| \mathbf{w} \|_2^2)$$

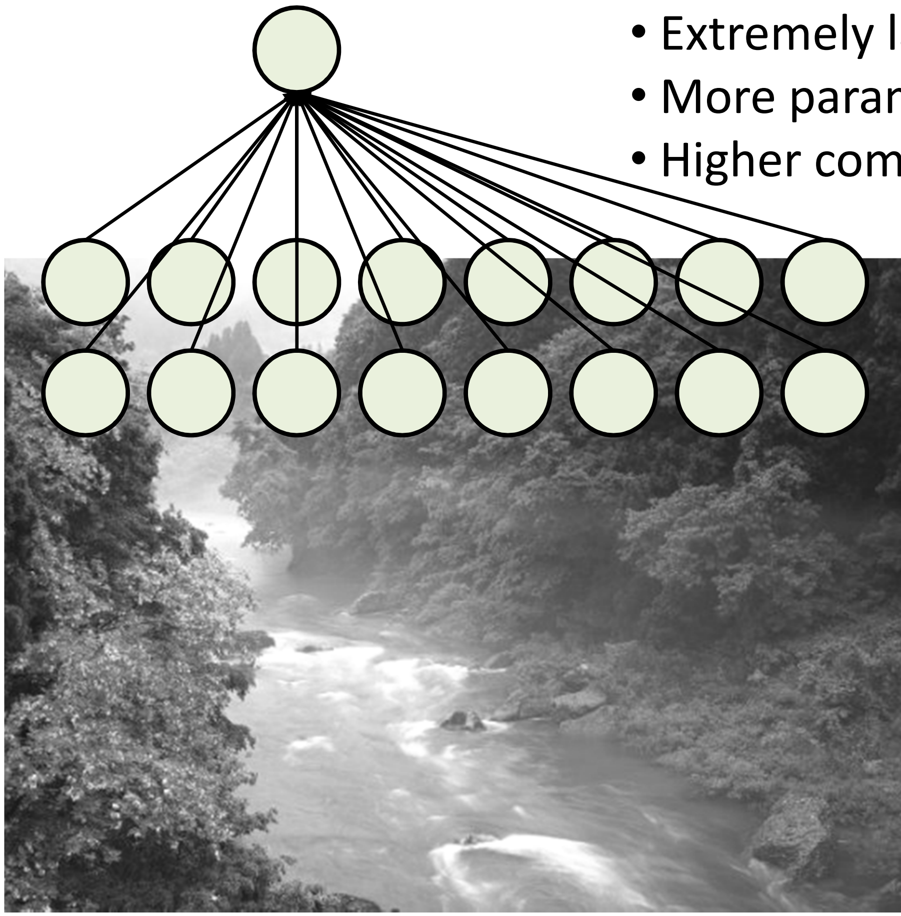
- Adding **L1 regularization** term to the loss function:

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} (L(\mathbf{w}) + \lambda \| \mathbf{w} \|_1)$$

Yet, things will not still work well!



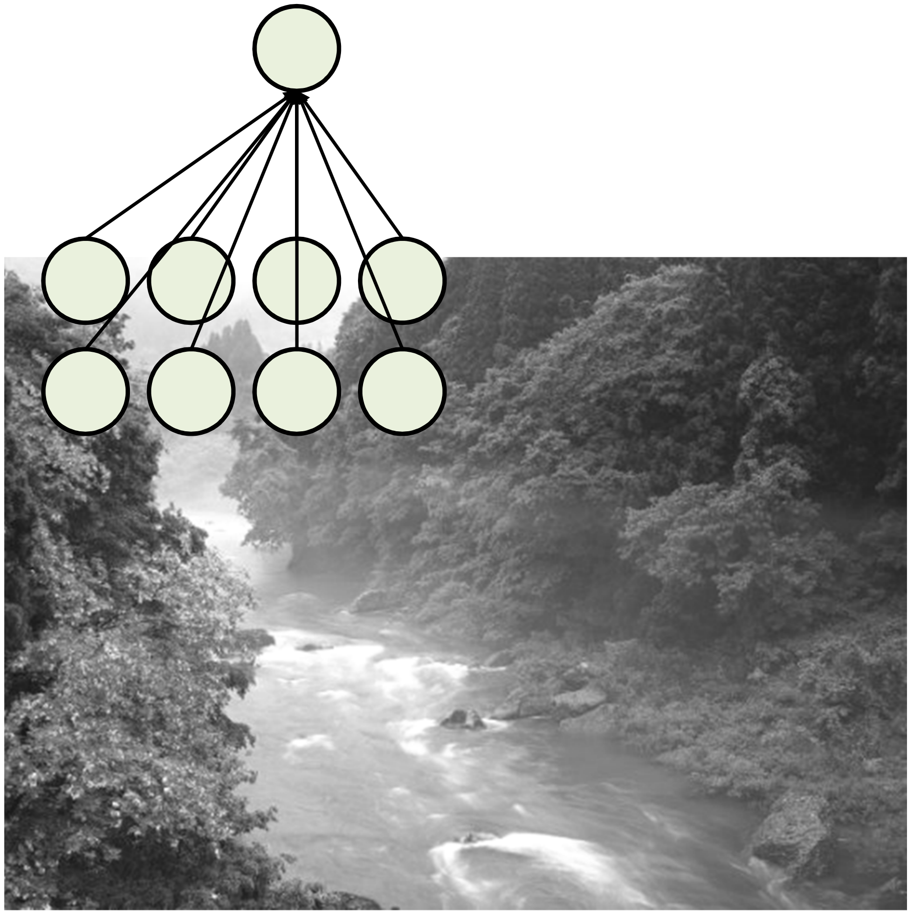
Main problem



- Extremely large number of connections.
- More parameters to train.
- Higher computational expense.

modified slides originally by Adam Coates

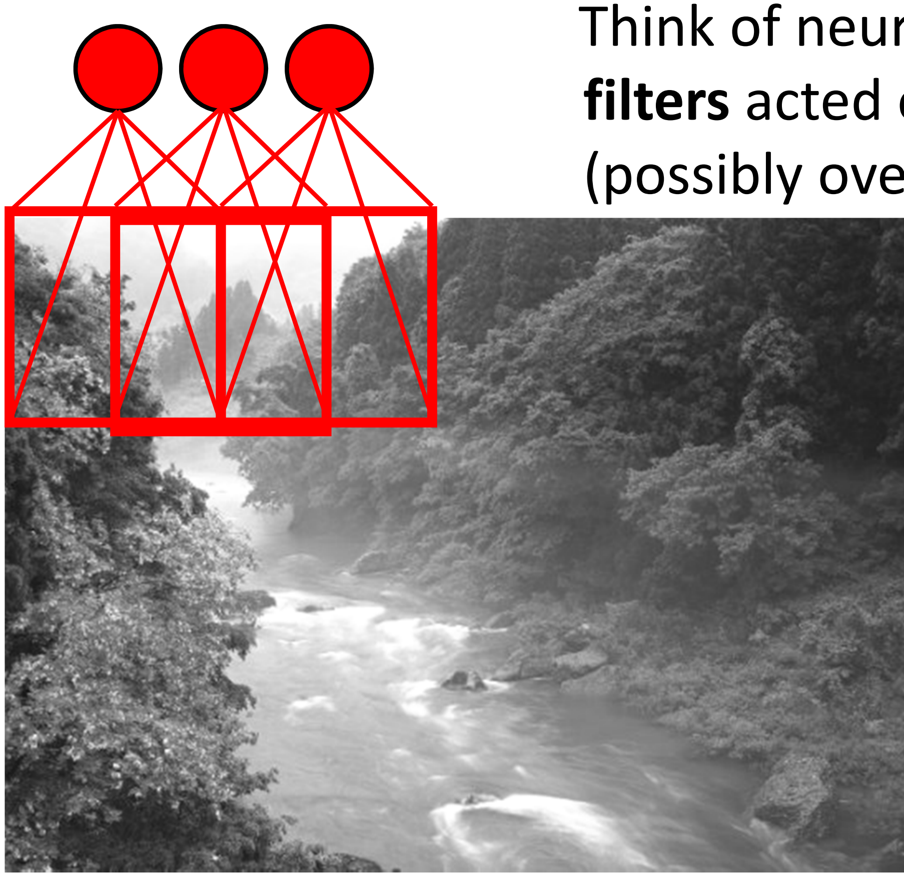
Local connectivity



Reduce parameters with
local connections!

modified slides originally by Adam Coates

Neurons as convolution filters

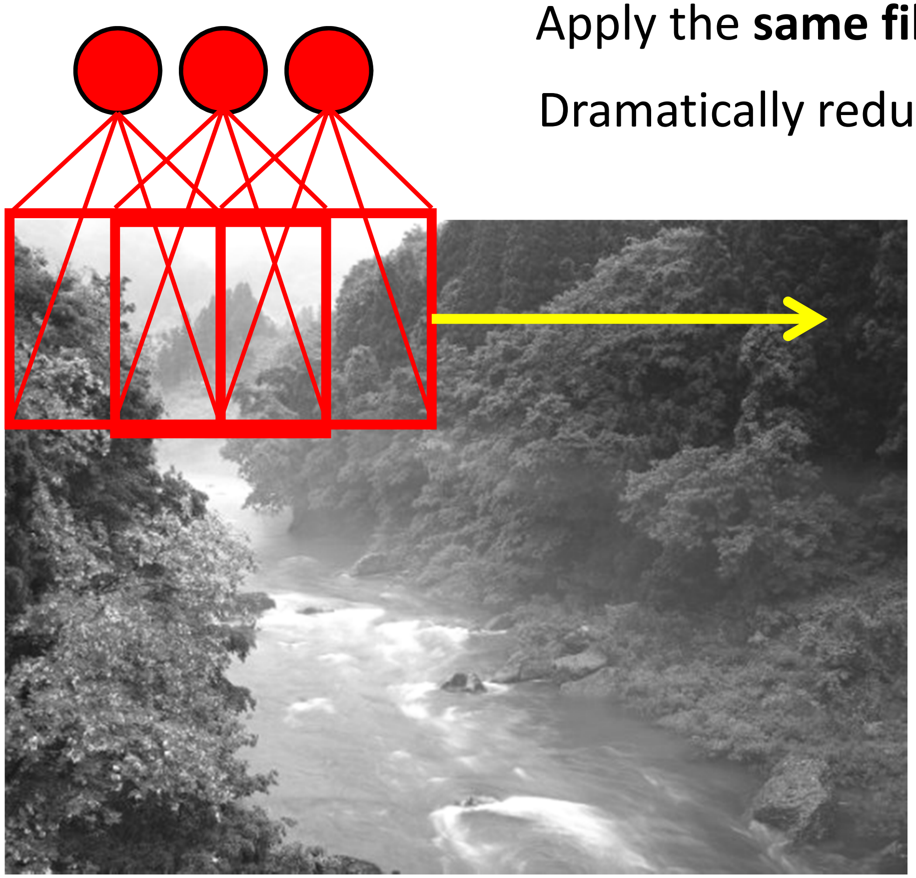


Think of neurons as convolutional **filters** acted on small adjacent (possibly overlapping) windows

Window size is called **“receptive field” size** and spacing is called **“step” or “stride”**

modified slides originally by Adam Coates

Extract repeated structure

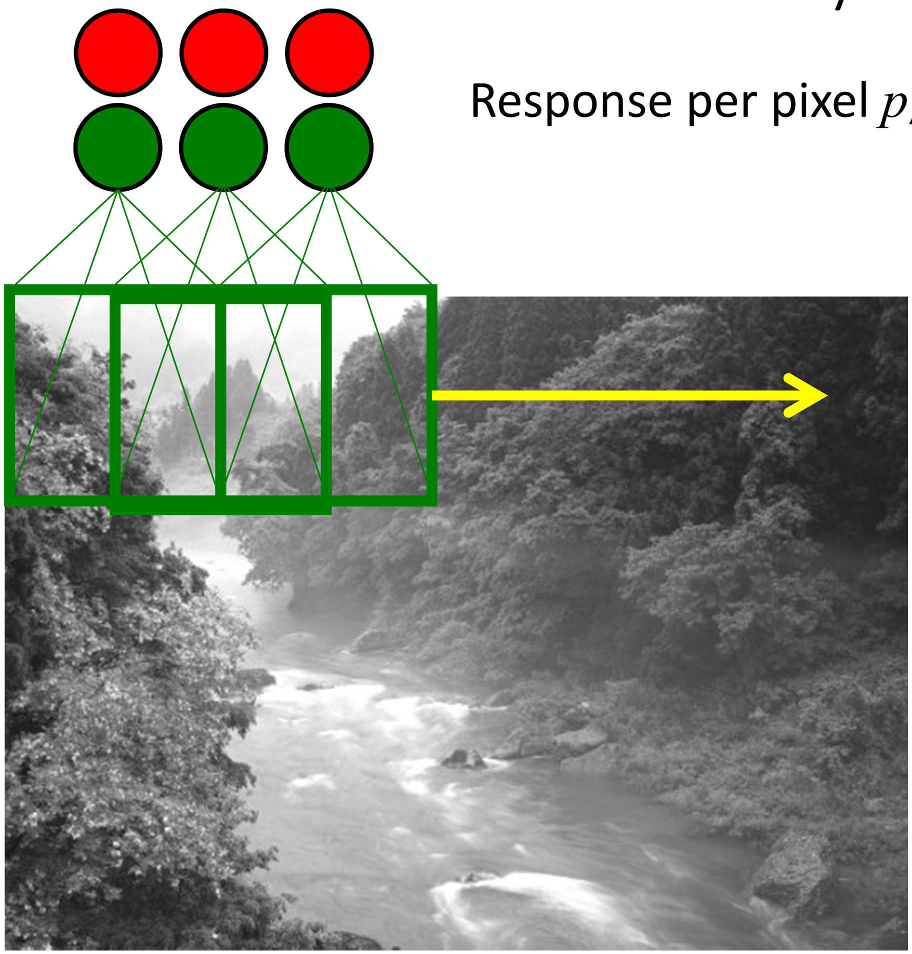


Apply the **same filter** (weights) throughout the image

Dramatically reduces the number of parameters

modified slides originally by Adam Coates

Can have many filters!



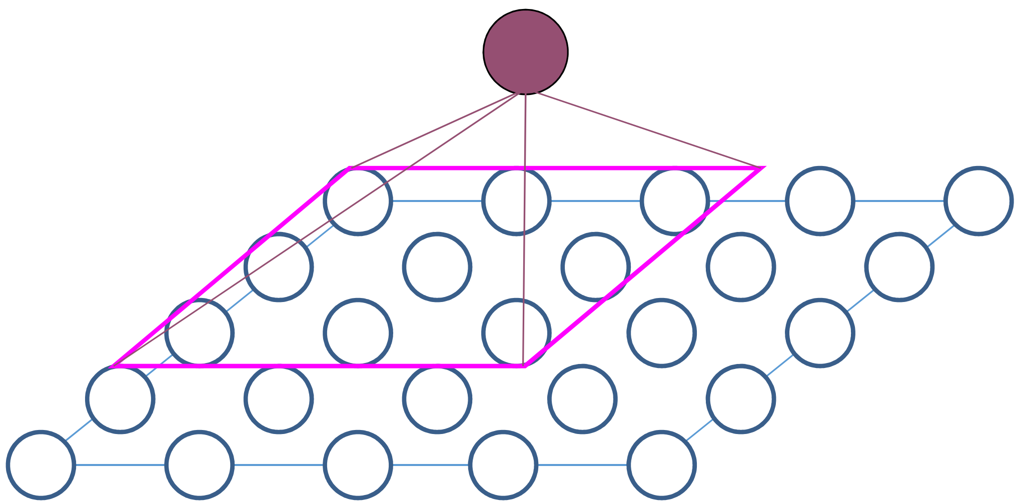
Response per pixel p, per filter f for a transfer function g:

$$h_{p,f} = g(\mathbf{w_f} \cdot \mathbf{x_p})$$

modified slides originally by Adam Coates

Pooling

Apart from hidden layers dedicated to convolution, we can have layers dedicated to extract **locally invariant** descriptors



Max pooling:

$$h_{p',f} = \max_{p}(\mathbf{x_p})$$

Mean pooling:

$$h_{p',f} = \underset{p}{avg}(\mathbf{x_p})$$

Fixed filter (e.g., Gaussian):

$$h_{p',f} = w_{gaussian} \cdot \mathbf{x_p}$$

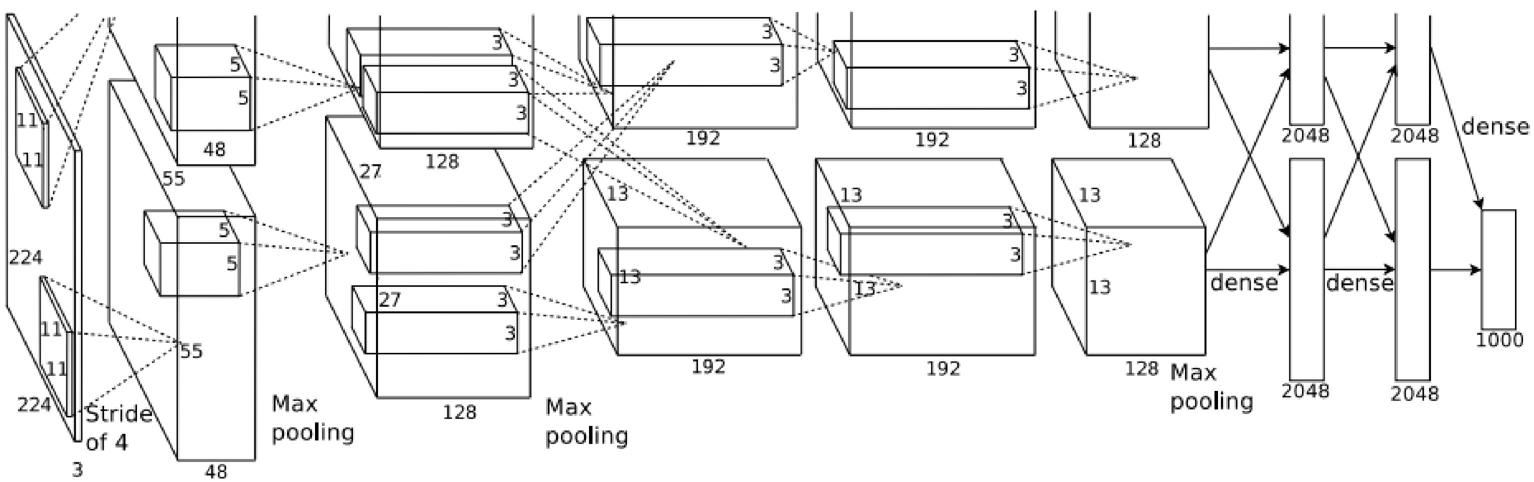
Progressively reduce the resolution of the image, so that the next convolutional filters are applied on larger scales

[Scherer et al., ICANN 2010]
[Boureau et al., ICML 2010]

Convolutional Neural Networks

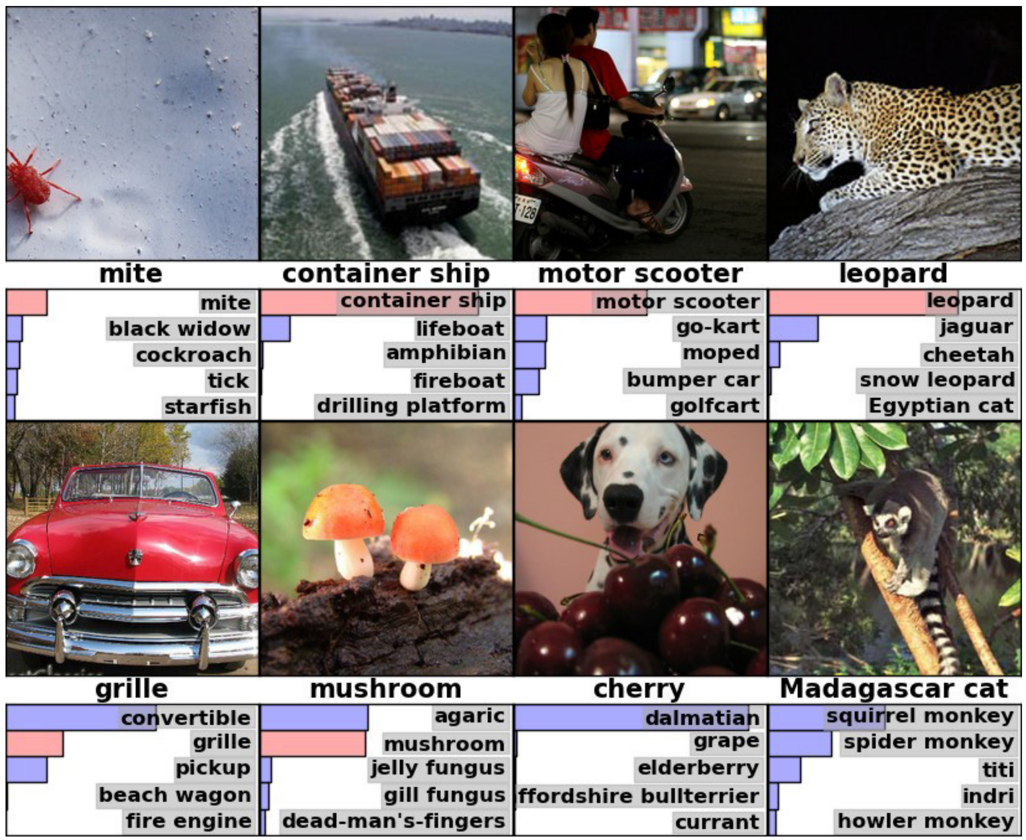
ImageNet system from Krizhevsky et al., NIPS 2012:

- Convolutional layers
- Max-pooling layers
- Rectified linear units (ReLu).
- Stochastic gradient descent, L2 regularization etc

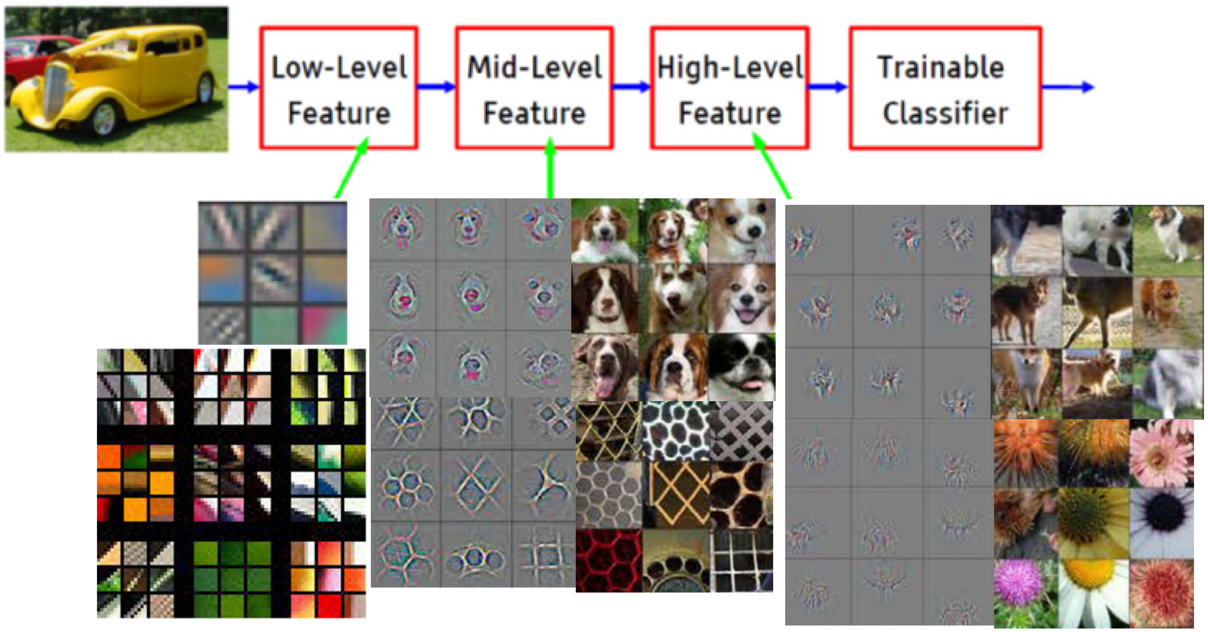


Application: Image-Net

Top result in LSVRC 2012: ~85%, Top-5 accuracy.



Learned representations



From Matthew D. Zeiler and Rob Fergus, Visualizing and Understanding Convolutional Networks, 2014

Multi-view CNNs

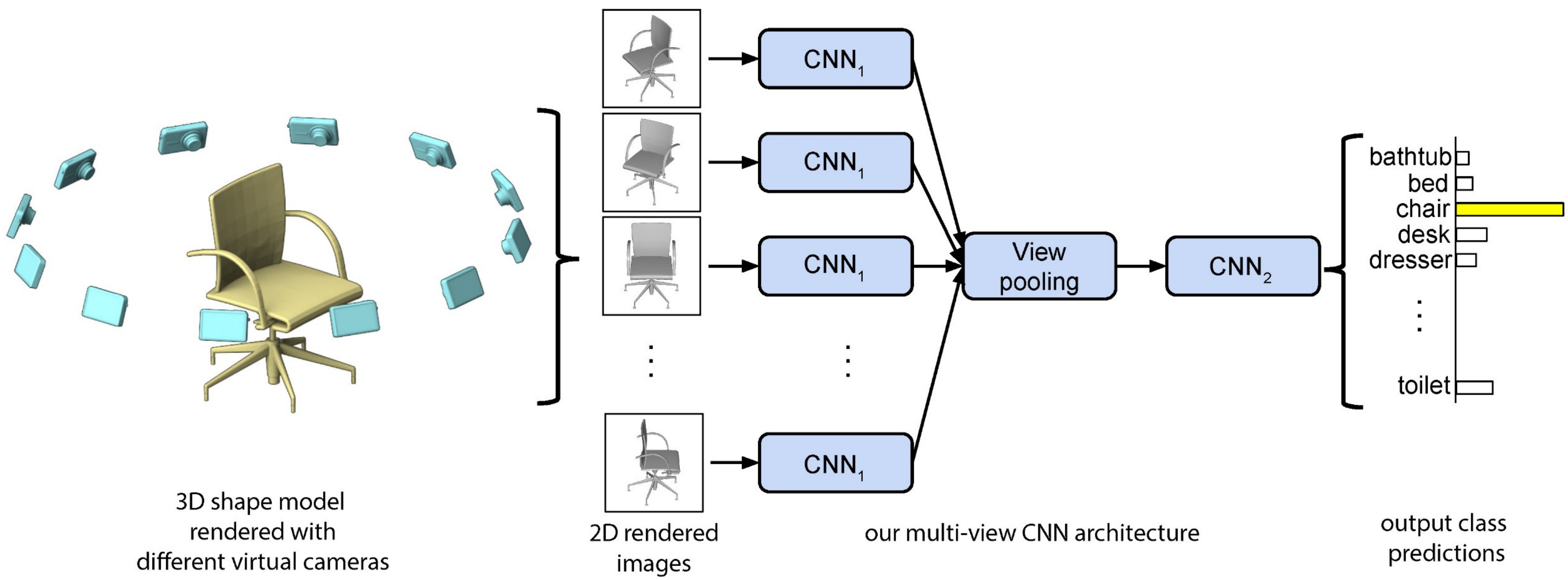


Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller, Multi-view Convolutional Neural Networks for 3D Shape Recognition, 2015

Multi-view CNNs

Use output of fully connected layer as a shape descriptor.

Shape retrieval evaluation in ModelNet40:

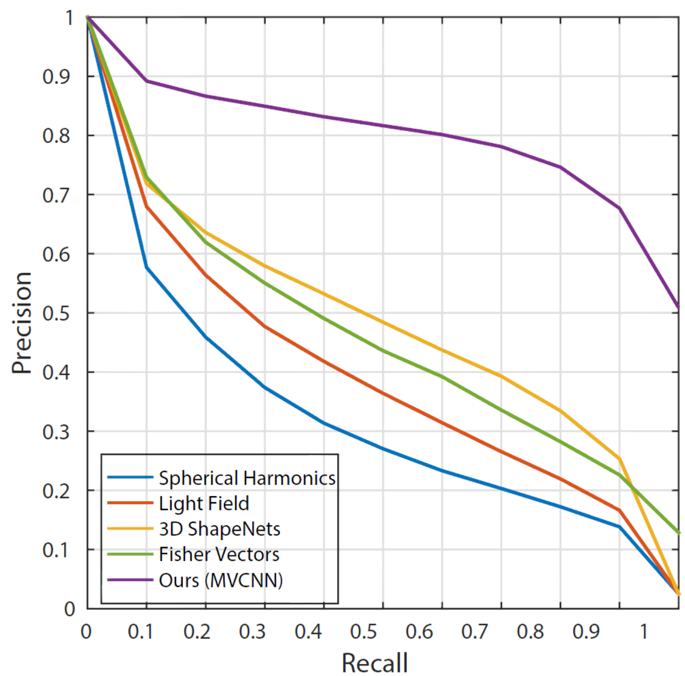


Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller, Multi-view Convolutional Neural Networks for 3D Shape Recognition, 2015

Sketch-based 3D Shape Retrieval using Convolutional Neural Networks

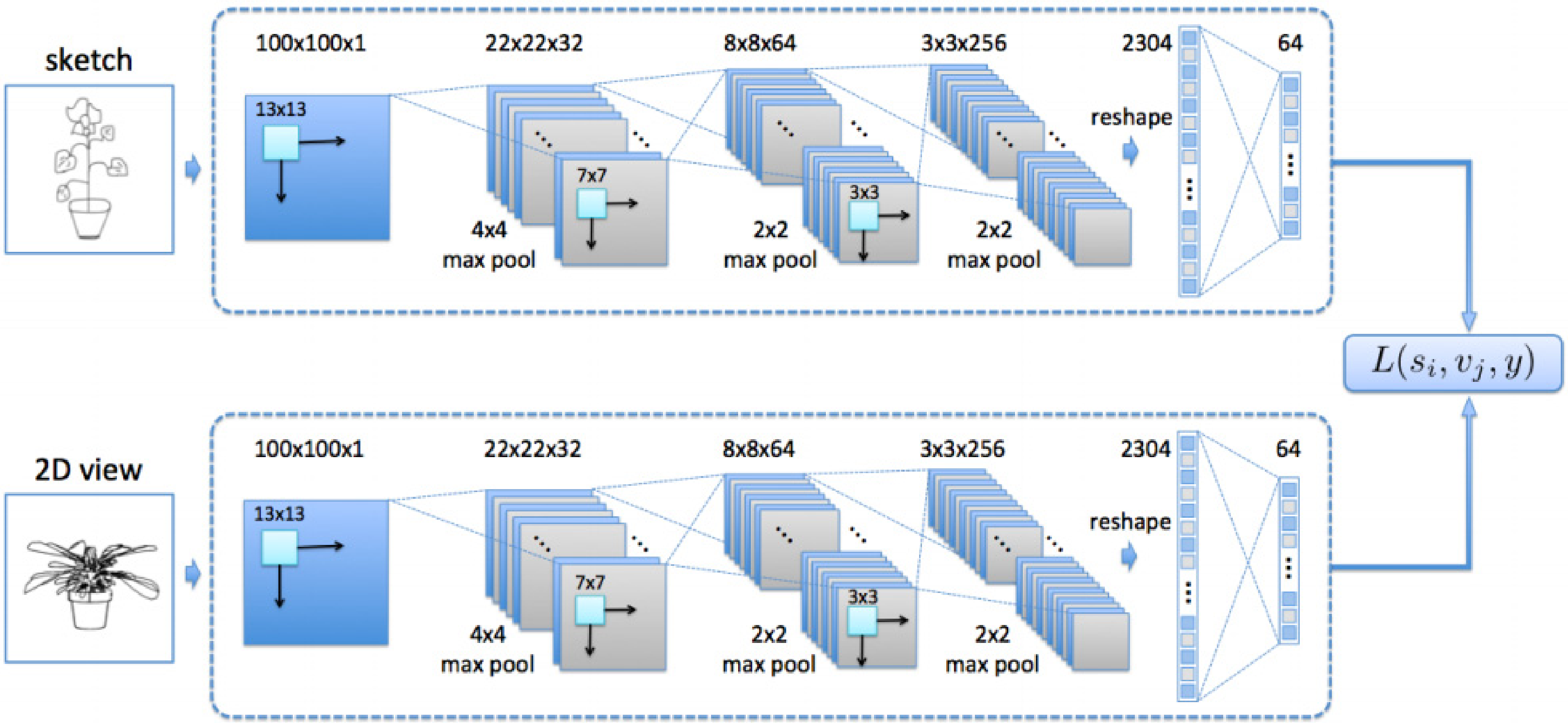


Image from Fang Wang, Le Kang, Yi Li,
Sketch-based 3D Shape Retrieval using Convolutional Neural Networks, 2015

Sketch-based 3D Shape Retrieval using Convolutional Neural Networks

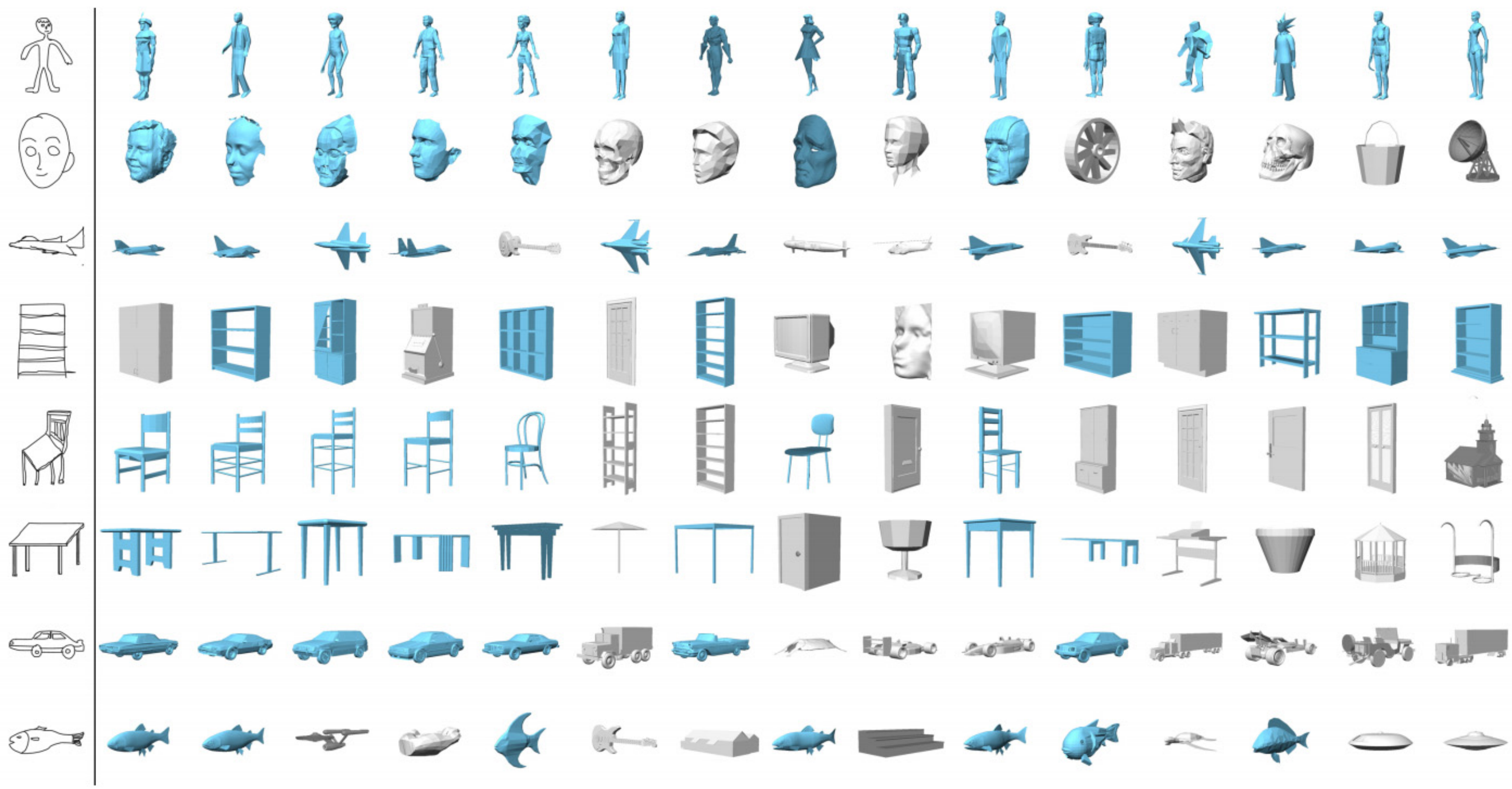


Image from Fang Wang, Le Kang, Yi Li,
Sketch-based 3D Shape Retrieval using Convolutional Neural Networks, 2015

Sketch-based 3D Shape Retrieval using Convolutional Neural Networks

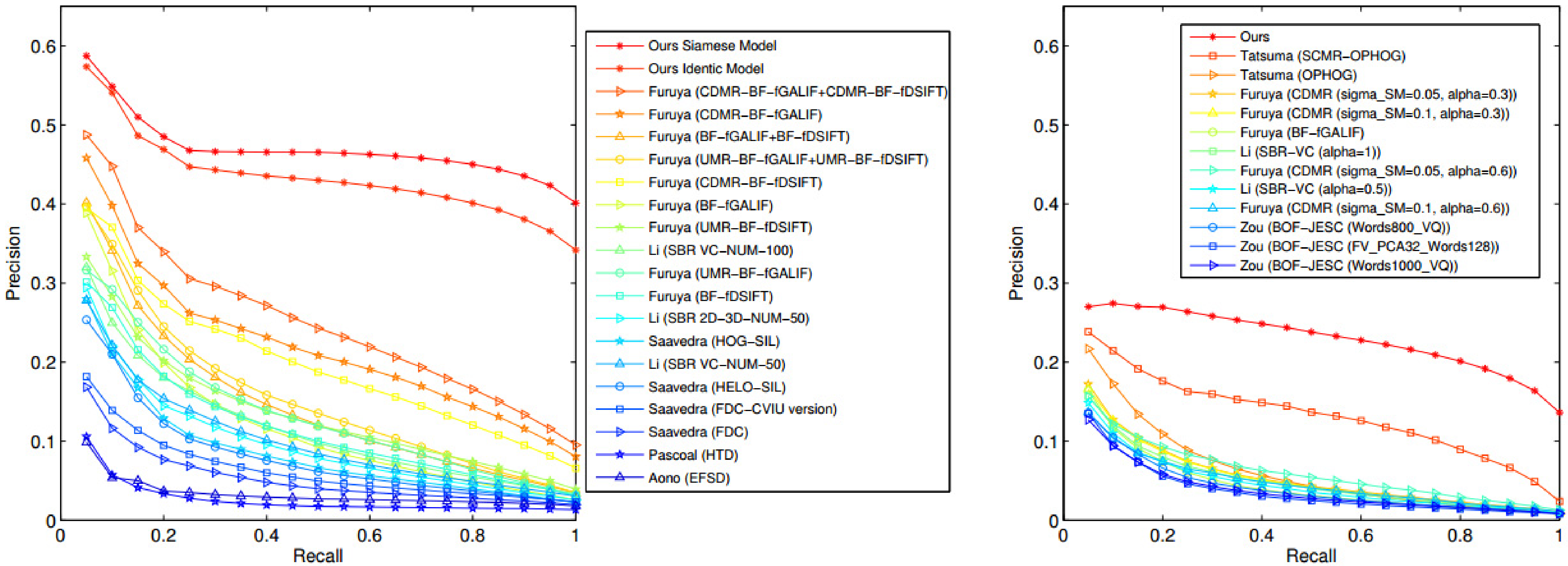


Image from Fang Wang, Le Kang, Yi Li,
Sketch-based 3D Shape Retrieval using Convolutional Neural Networks, 2015

Learning to Generate Chairs

Inverting the CNN…

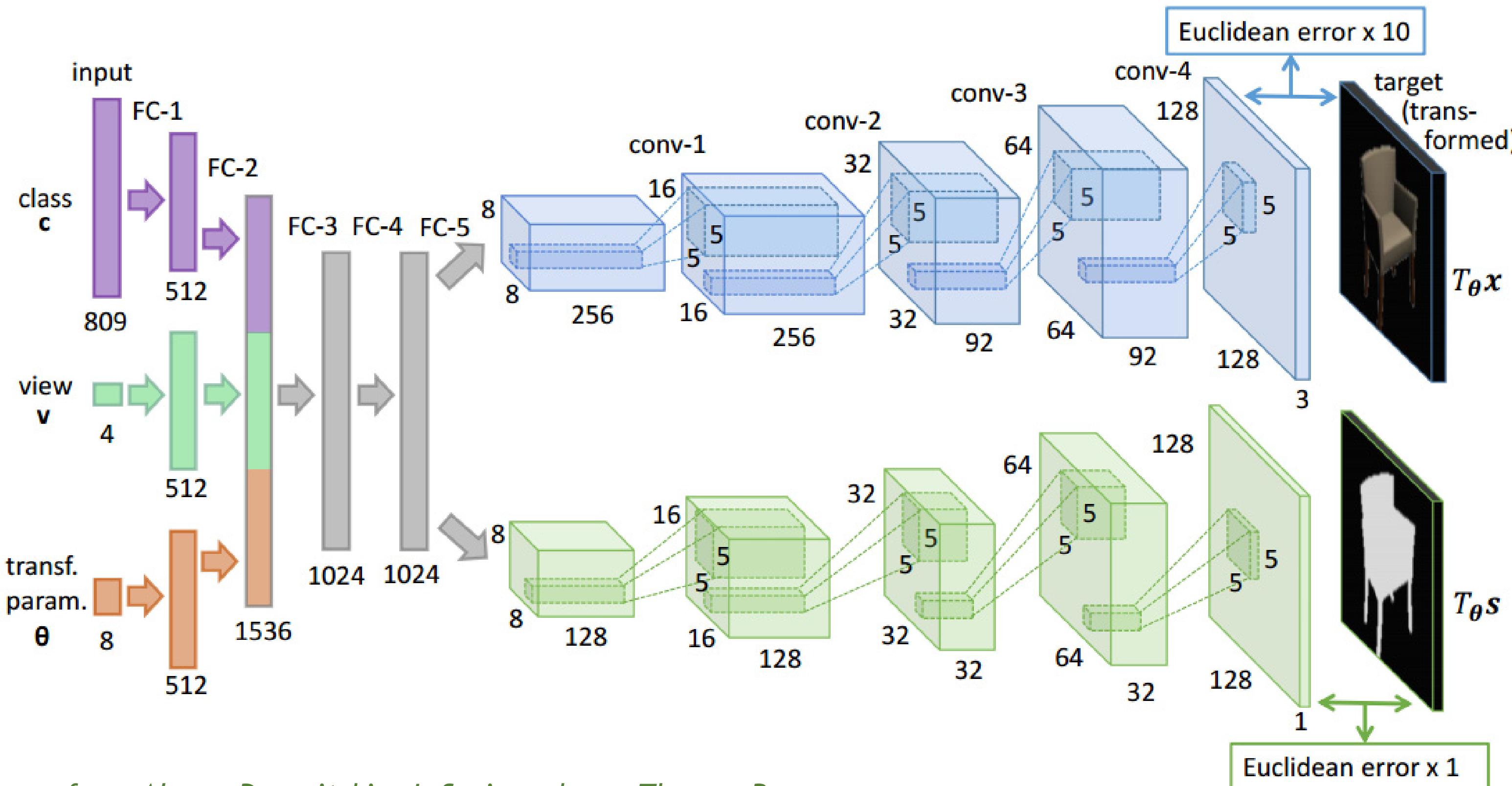


Image from Alexey Dosovitskiy, J. Springenberg, Thomas Brox
Learning to Generate Chairs with Convolutional Neural Networks 2015

to access video: http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15/

Learning to Generate Chairs

Inverting the CNN…

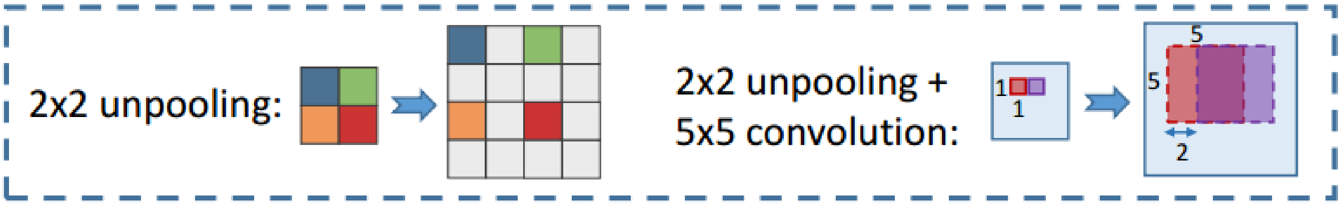


Image from Alexey Dosovitskiy, J. Springenberg, Thomas Brox
Learning to Generate Chairs with Convolutional Neural Networks 2015

Deep learning on volumetric representations

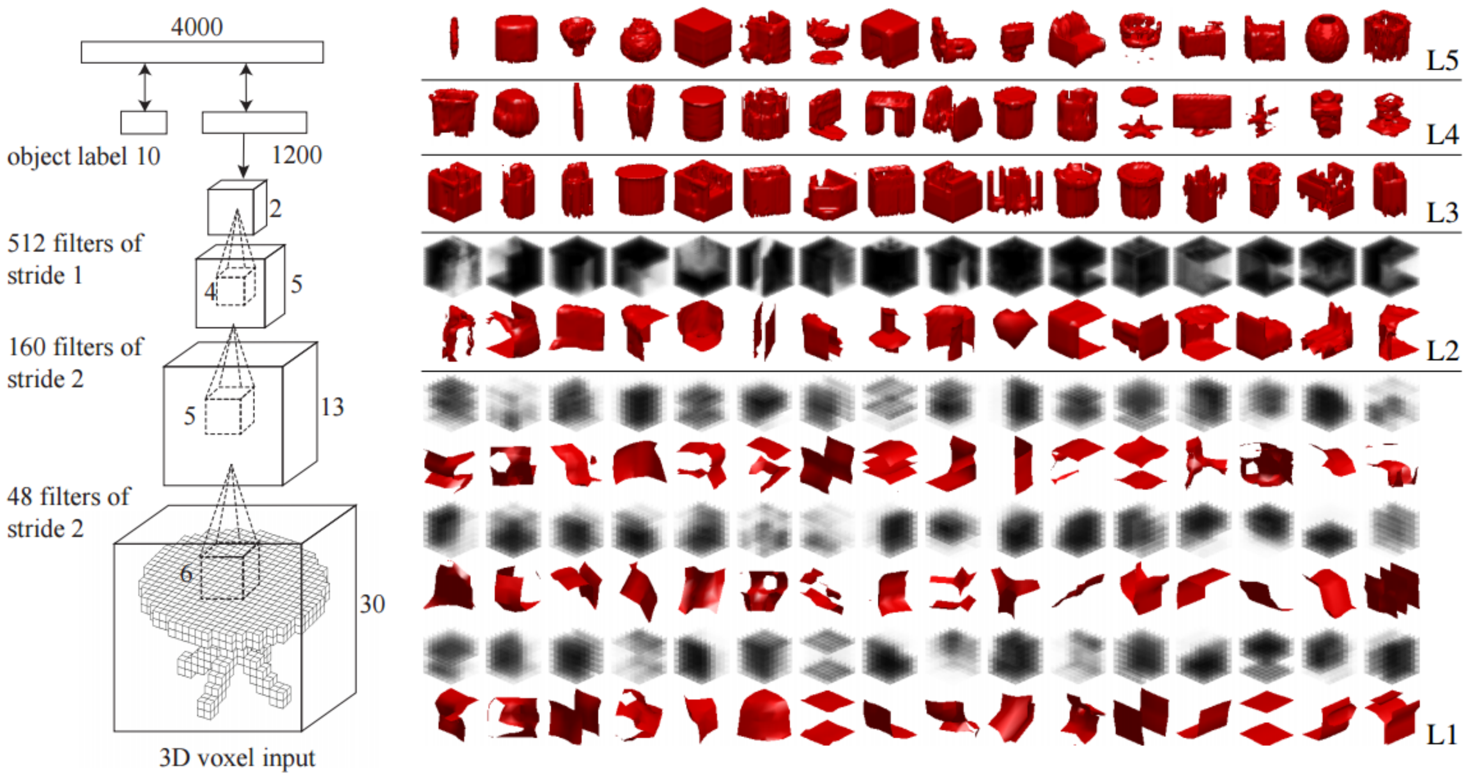


Image from Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao
3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015

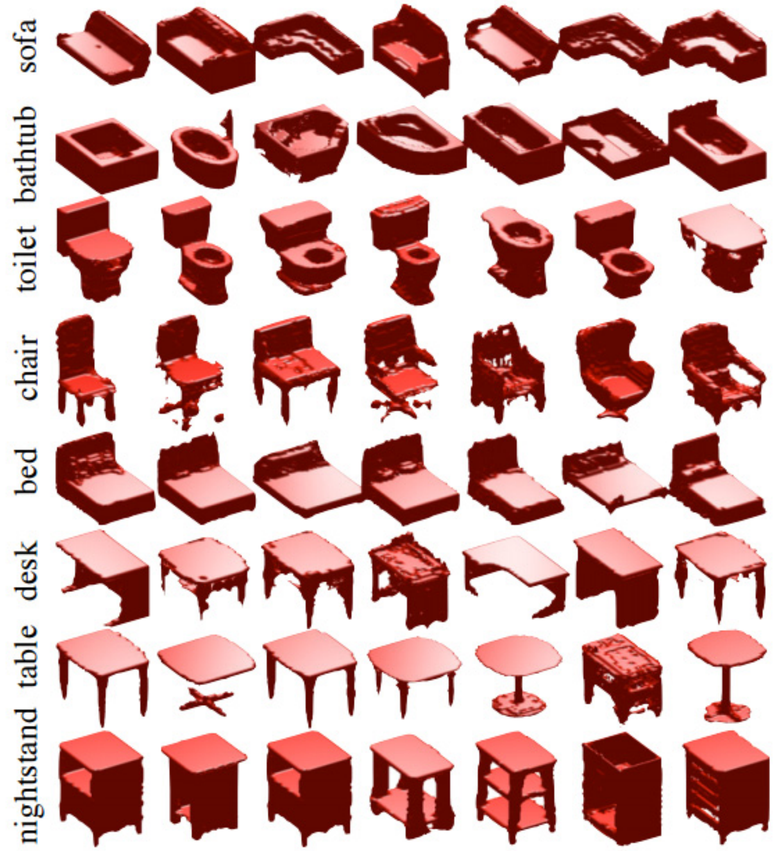


Image from Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao
3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015

Summary

Welcome to the era where machines **learn to generate 3D visual content!**

Deep learning seems one of the most promising directions

Summary

Welcome to the era where machines **learn to generate 3D visual content!**

Deep learning seems one of the most promising directions

Big challenges:

- Generate **plausible, detailed, novel** 3D geometry from **high-level specifications, approximate** directions
- What **shape representation** should deep networks operate on?
- Integrate with approaches that optimize for **function** and **human-object interaction**