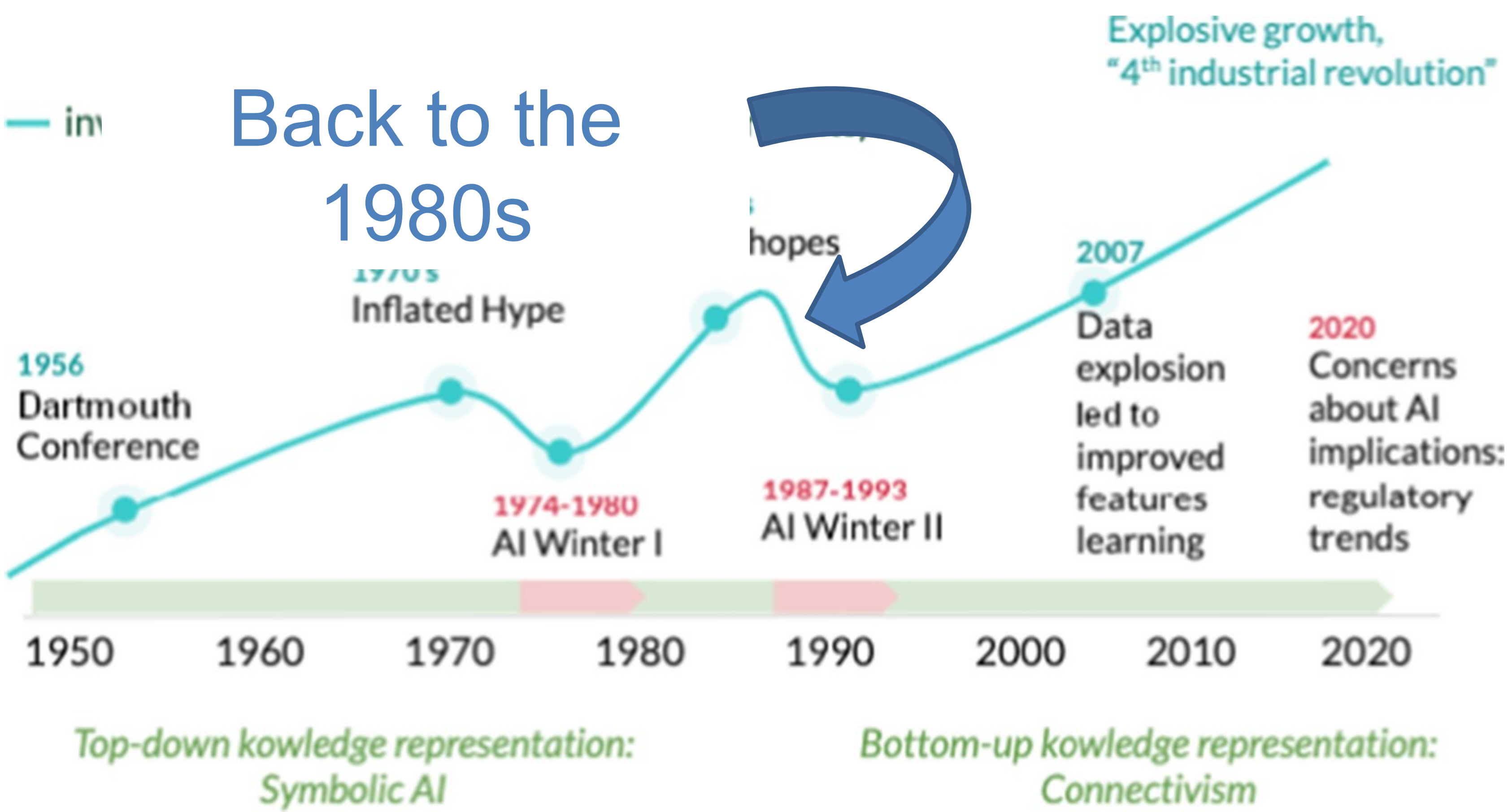
An Intro to Deep Learning for NLP

Mausam

Disclaimer: this is an outsider's understanding. Some details may be inaccurate

(several slides by Yonatan Belinkov Yoav Goldberg & Graham Neubig)

Back to the 1980s



[Figure: Francesconi, 2022]

The Localist vs. Distributed Debate

Distributed Representations

G. E. HINTON, J. L. McCLELLAND, and D. E. RUMELHART

Connectionist modelling in psychology: A localist manifesto

Local vs. Distributed Coding

Simon Thorpe

Mike Page
Medical Research Council Cognitic
Cambridge, CB2 2EF, United Kingc
mike.page@mrc-cbu.cam.ac.uk

Compositional connectionism in cognitive science II: the localist/distributed dimension

Ross W. Gayler & Simon D. Levy

Pages 85-89 | Published online: 27 May 2011

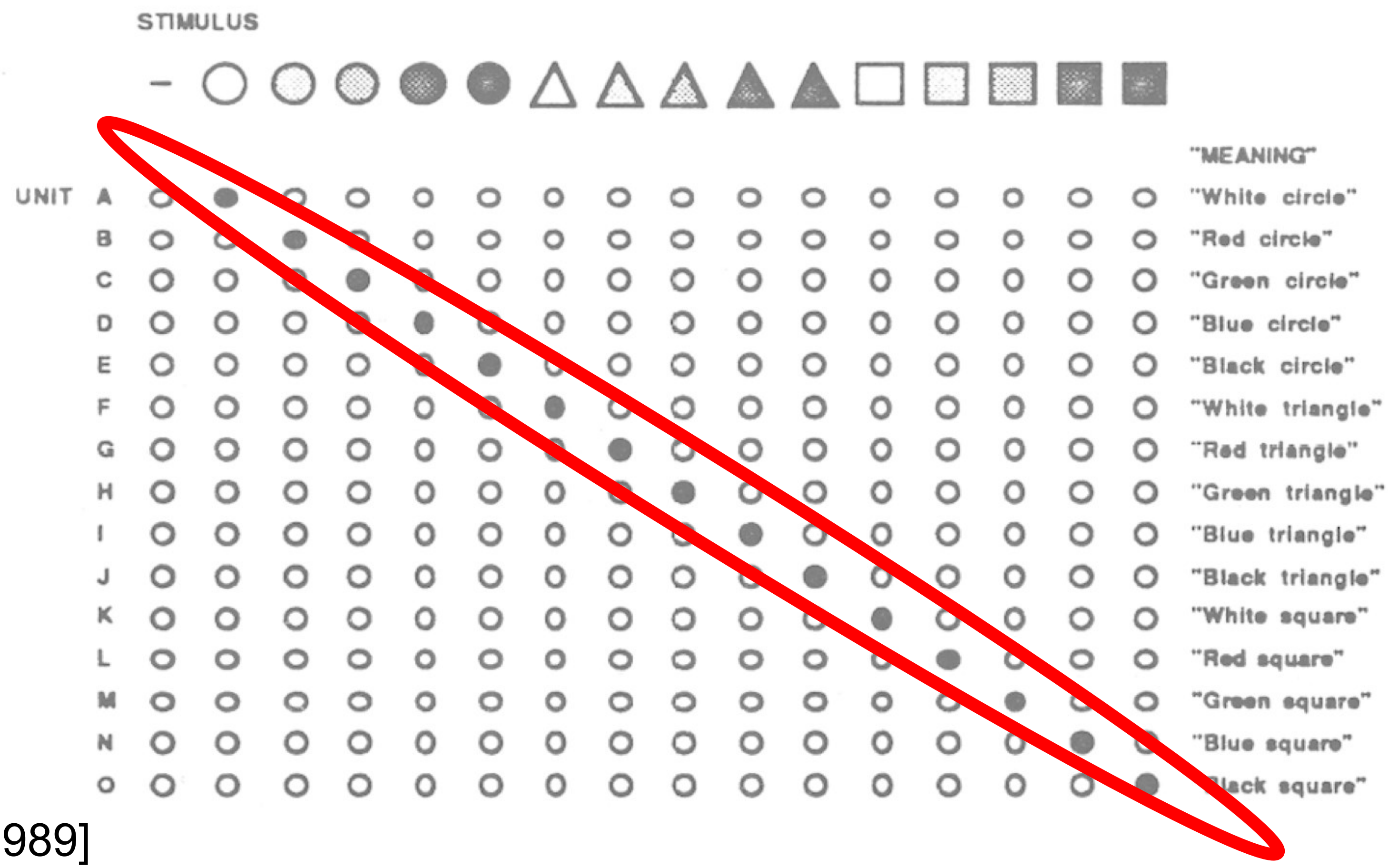
Distributed vs. Localist Representations

Localist: "..one computing element for each entity"

Distributed:

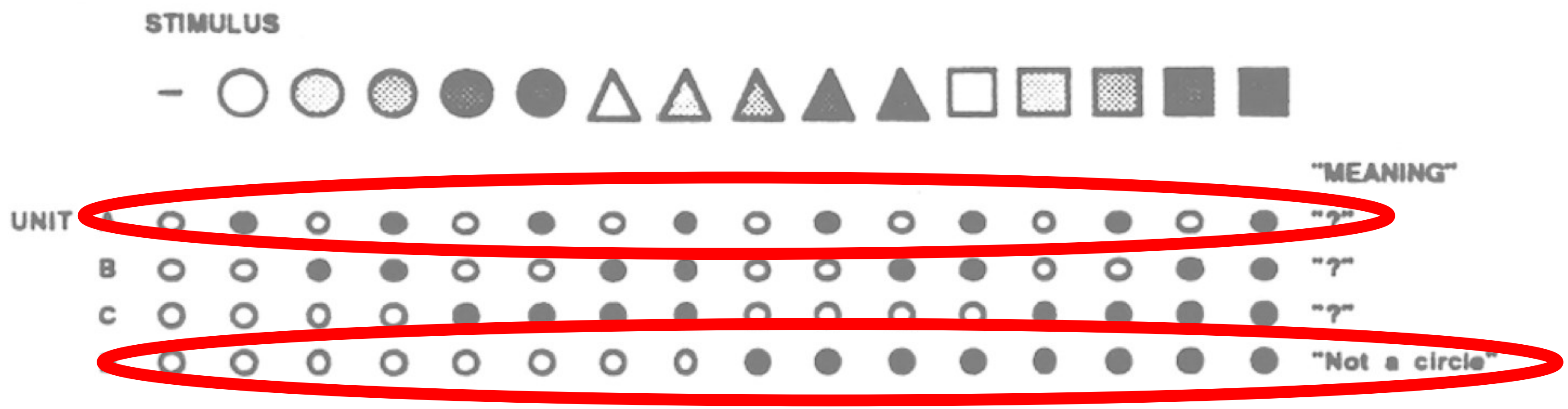
- "Each entity is represented by a pattern of activity distributed over many computing elements"
- "each computing element is involved in representing many different entitites"

Local Representation



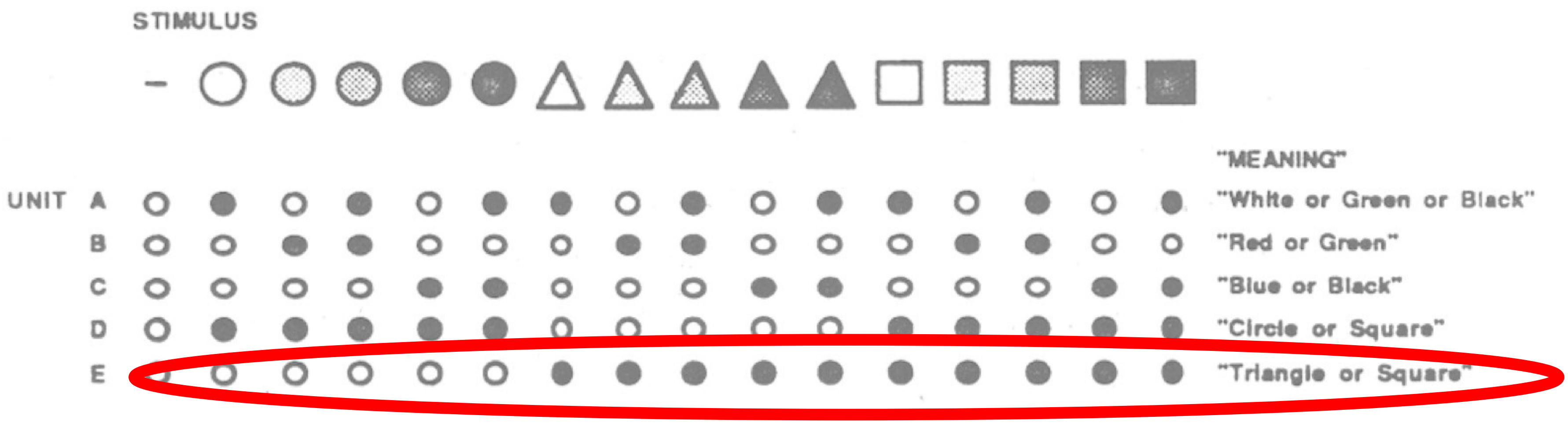
[Thorpe 1989]

Distributed Representation



[Thorpe 1989]

Semi-Distributed Representation



[Thorpe 1989]

Distributed Representations: Pros and Cons

Distributed representations:

- Efficient 😀
- Continuous 😀
- Degrade gracefully 😀
- Less interpretable 😢

Localist representations

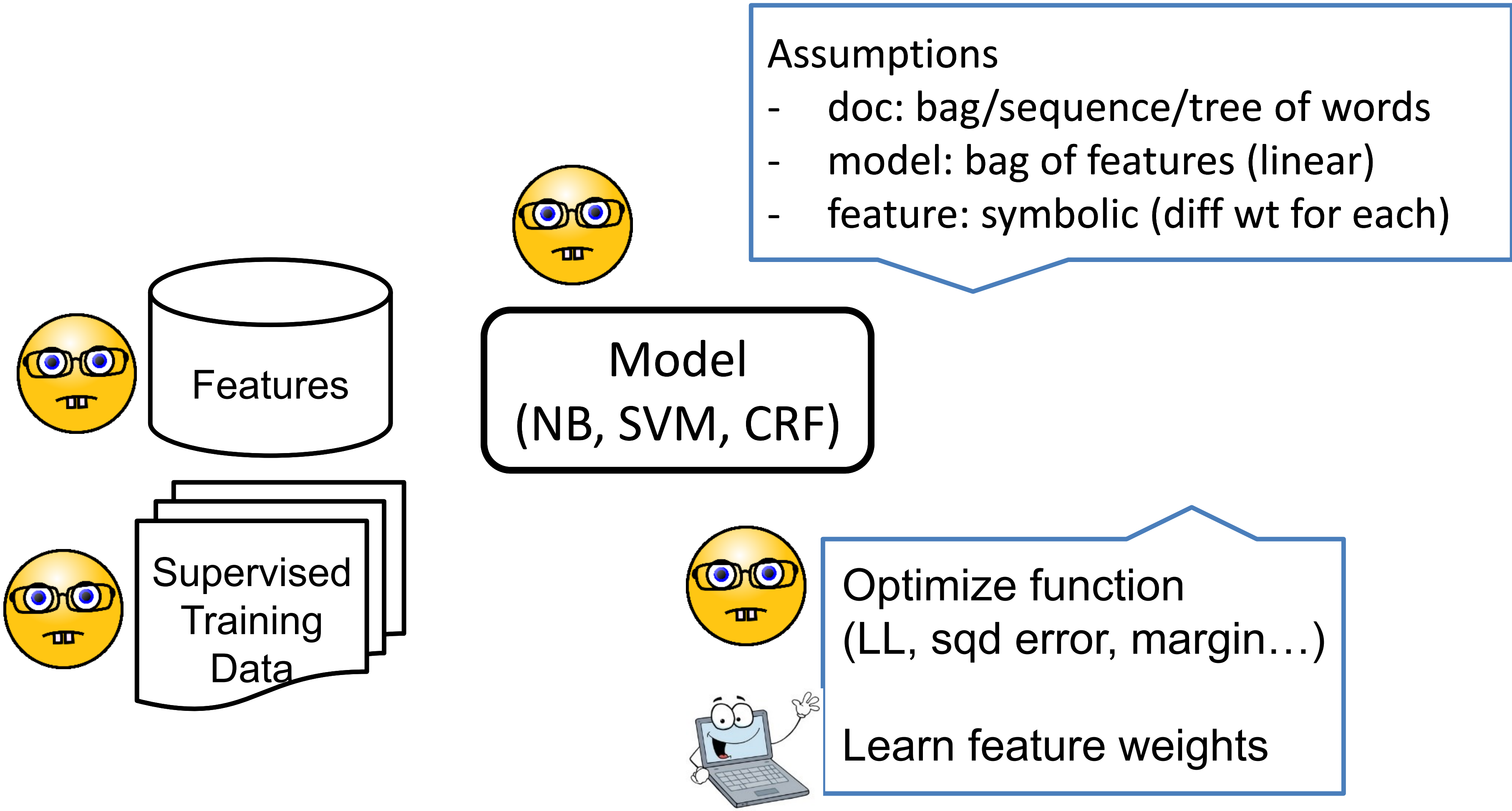
- Easier to work with(?) 😀
- More interpretable 😀

[Pate 2002]

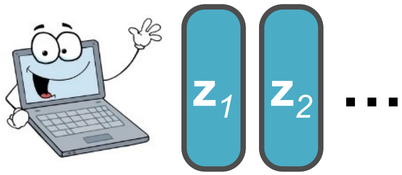
So, who won?

I think we all know…

NLP before DL #1



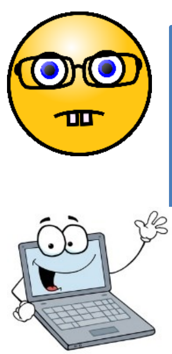
NLP before DL #2



Assumptions
- doc/query/word is a vector of numbers
- dot product can compute <u>similarity</u>
  - via distributional hypothesis

Model
(MF, LSA, IR)

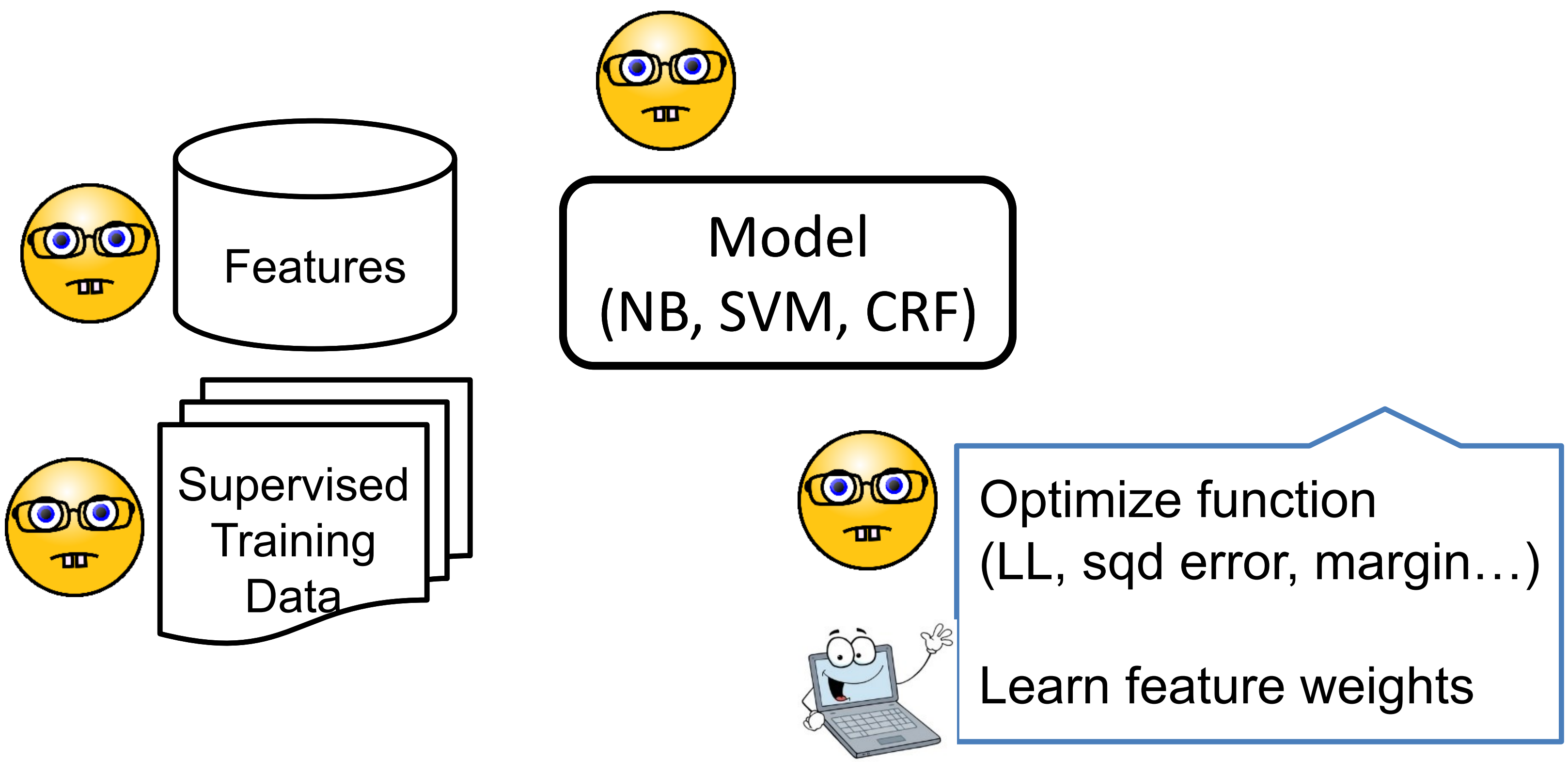
Unsupervised
Co-occurrence
Data



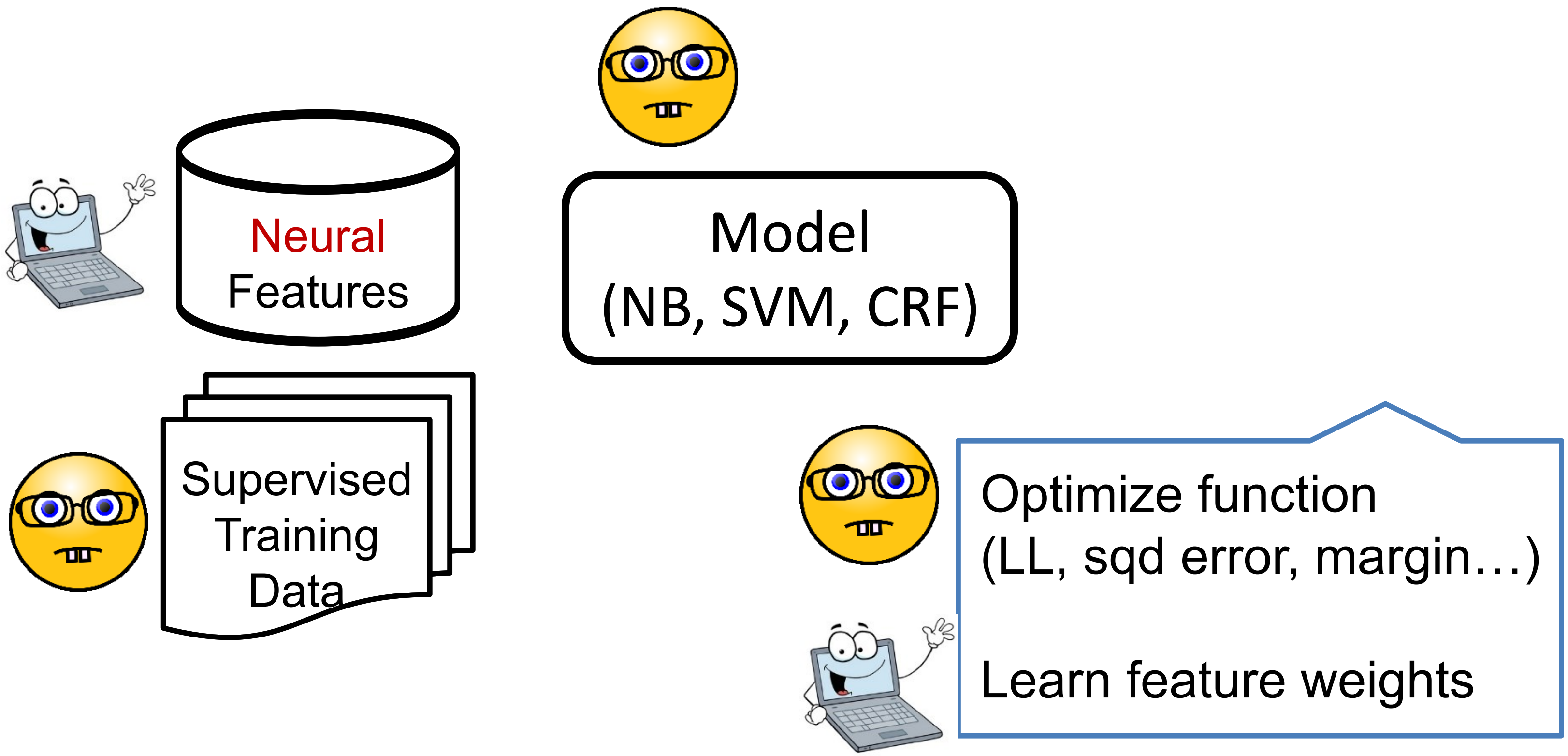
Optimize function
(LL, sqd error, margin…)

Learn vectors

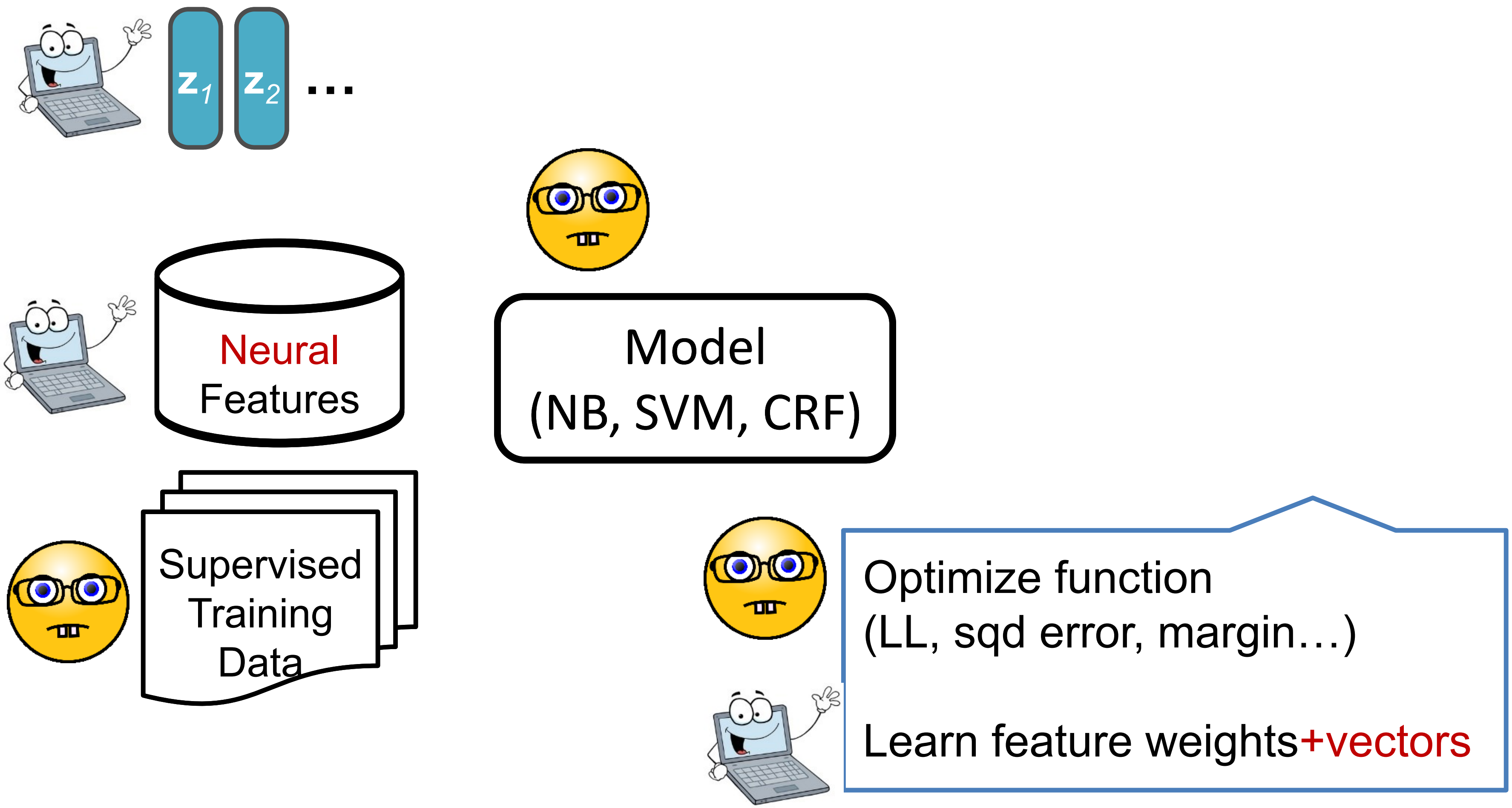
NLP with DL


Features
Supervised Training Data
Model
(NB, SVM, CRF)
Optimize function
(LL, sqd error, margin…)
Learn feature weights

NLP with DL

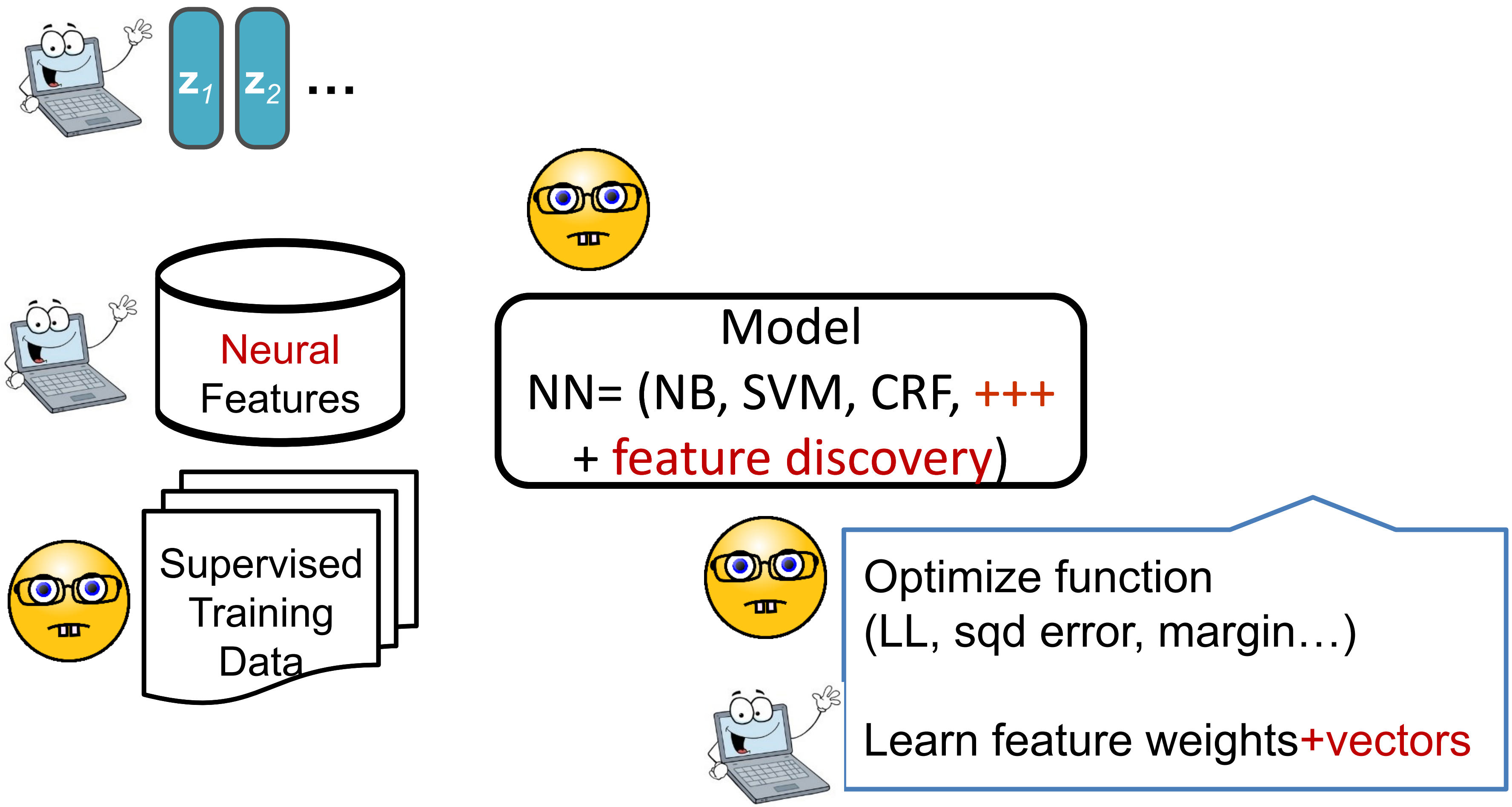

Neural
Features
Supervised
Training
Data
Model
(NB, SVM, CRF)
Optimize function
(LL, sqd error, margin…)
Learn feature weights

NLP with DL



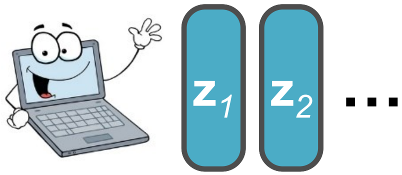
z_1 z_2 …
Neural Features
Supervised Training Data
Model (NB, SVM, CRF)
Optimize function (LL, sqd error, margin…)
Learn feature weights+vectors

NLP with DL



z_1 z_2 …
Neural Features
Supervised Training Data
Model
NN= (NB, SVM, CRF, +++
+ feature discovery)
Optimize function
(LL, sqd error, margin…)
Learn feature weights+vectors

NLP with DL



Neural Features

Supervised Training Data

Assumptions

- doc/query/word is a vector of numbers
- doc: bag/sequence/tree of words
- feature: neural (weights are shared)
- model: bag/seq of features (non-linear)

Model

NN= (NB, SVM, CRF, +++

+ feature discovery)

Optimize function

(LL, sqd error, margin…)

Learn feature weights+vectors

Meta-thoughts

Features

- Learned
- in a task specific end2end way
- not limited by human creativity

Everything is a “Point”

- Word embedding
- Phrase embedding
- Sentence embedding
- Word embedding in context of sentence
- Etc

- Also known as dense/distributed representations

Points are good → reduce sparsity by wt sharing

a single (complex) model can handle all pts

Universal Representations

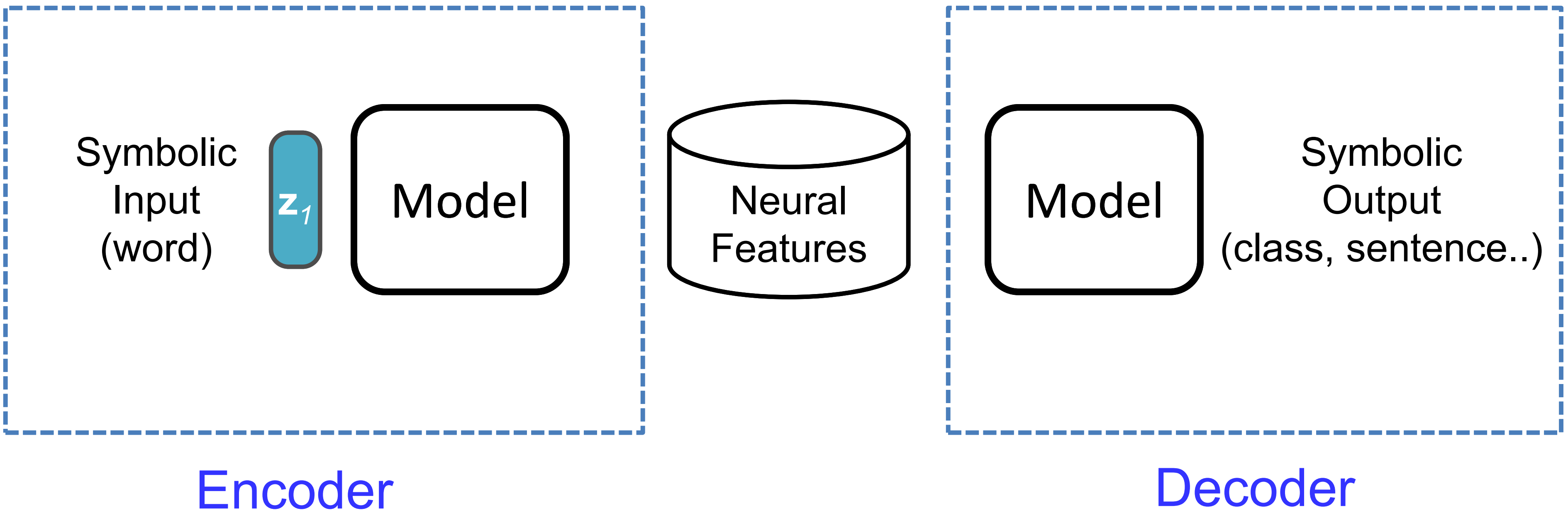
- Non-linearities
  - Allow complex functions
- Put anything computable in the loss function
  - Any additional insight about data/external knowledge

Make symbolic operations continuous

- Symbolic → continuous
  - Yes/No →
    - (number between 0 and 1)
  - Good/bad →
    - (number between -1 and 1)

  - Either remember or forget →
    - partially remember
  - Select from n things →
    - weighted avg over n things

Encoder-Decoder

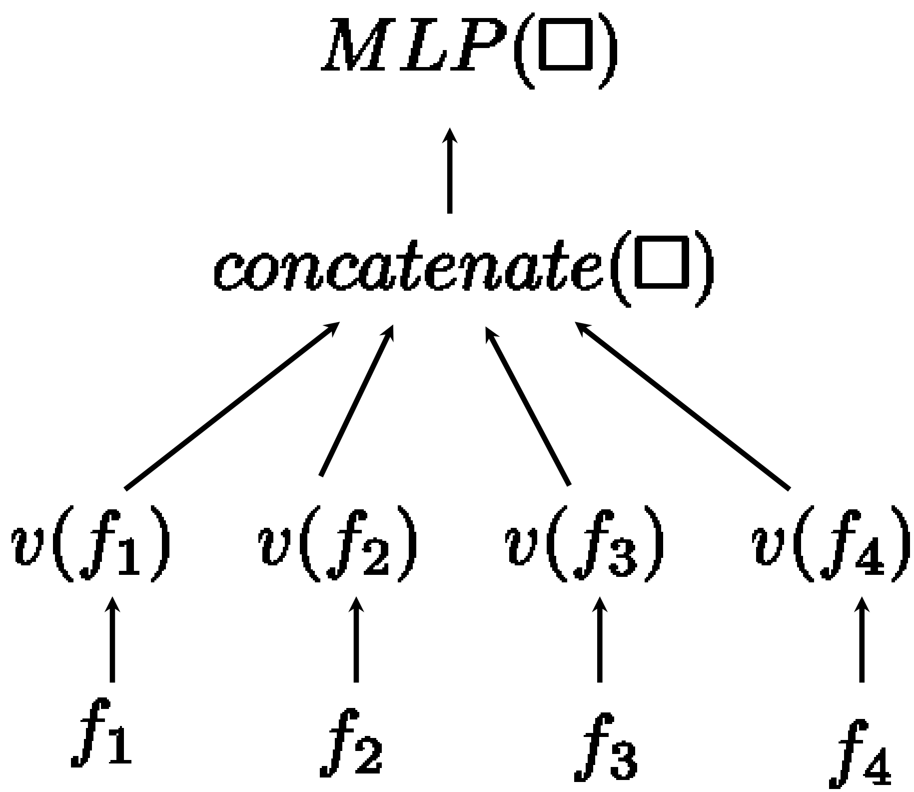


Different assumptions on data create different architectures

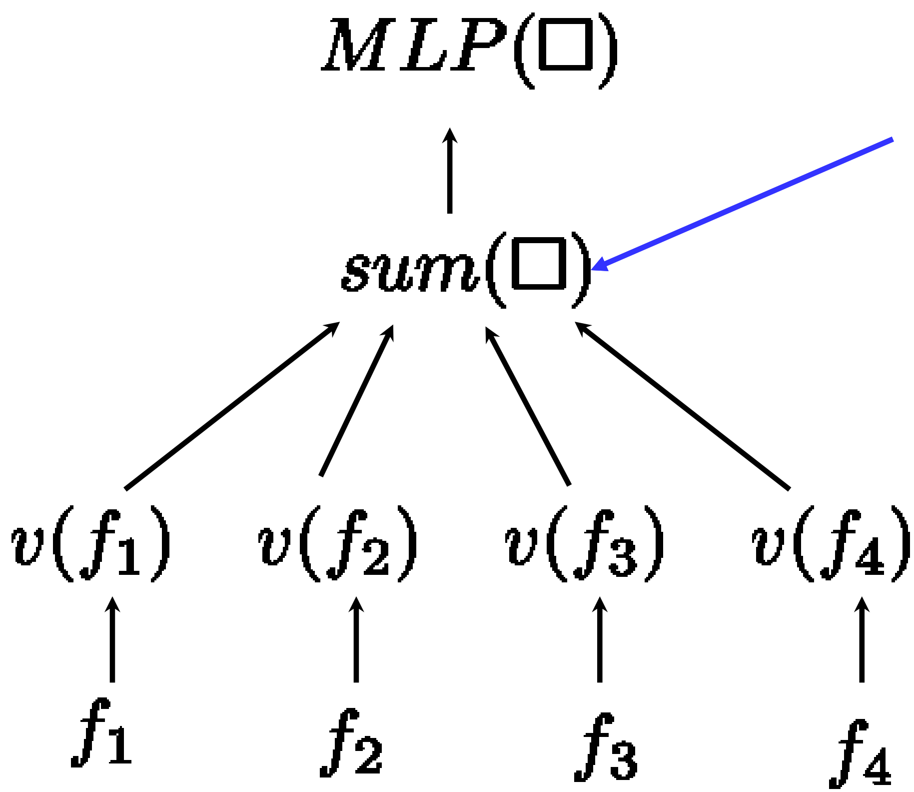
Building Blocks

+	;	.
Matrix-mult	gate	non-linearity

x;y



x+y



Can also try
Dimension-wise
Max
(later weighted sum)

Concat vs. Sum

- **Concatenating** feature vectors: the "roles" of each vector is retained.

$$concat\,(v(\text{"}the\text{"}), v(\text{"}thirsty\text{"}), v(\text{"}dog\text{"}))$$

prev word | current word | next word

- Different features can have vectors of different dim.
- Fixed number of features in each example (need to feed into a fixed dim layer).

Concat vs. Sum

- **Summing** feature vectors: "bag of features"

$$sum(v(\text{"}the\text{"}), v(\text{"}thirsty\text{"}), v(\text{"}dog\text{"}))$$

word word word

- Different feature vectors should have same dim.
- **Can encode a bag of arbitrary number of features.**

x.y

- degree of closeness
- alignment

- Uses
  - question aligns with answer //QA
  - sentence aligns with sentence //paraphrase
  - word aligns with (~important for) sentence //attention

g(Ax+b)

- 1-layer MLP
- Take x
  - project it into a different space //relevant to task
  - add some scalar bias (only increases/decreases it)
  - convert into a required output

- 2-layer MLP
  - Common way to convert input to output

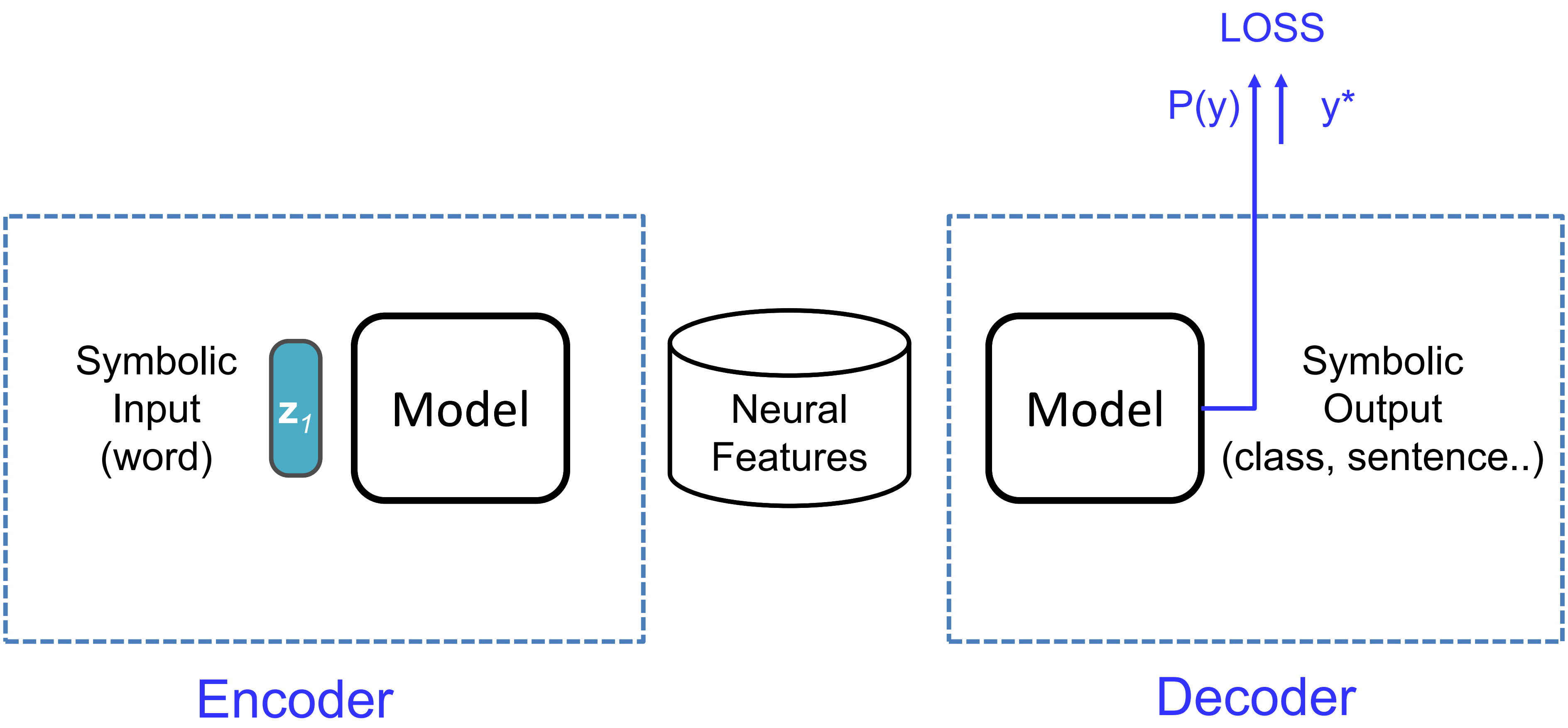
Loss Functions

Cross Entropy

Binary Cross Entropy

Max Margin

Encoder-Decoder



Common Loss Functions

- Binary Cross Entropy (2 class classification)

  Loss = -y*log p(y) – (1-y*)log(1-p(y))

- Categorical Cross Entropy (multi class class.)

$$Loss = -\sum_{k} y_k^* \log(p(y_k))$$

- Log Likelihood

$$p(y^*)$$

Common Loss Functions

- Max Margin

  Loss = max(0, 1-(score(y*)-score(y_{best})))

- Ranking loss (max margin: x ranked over x')

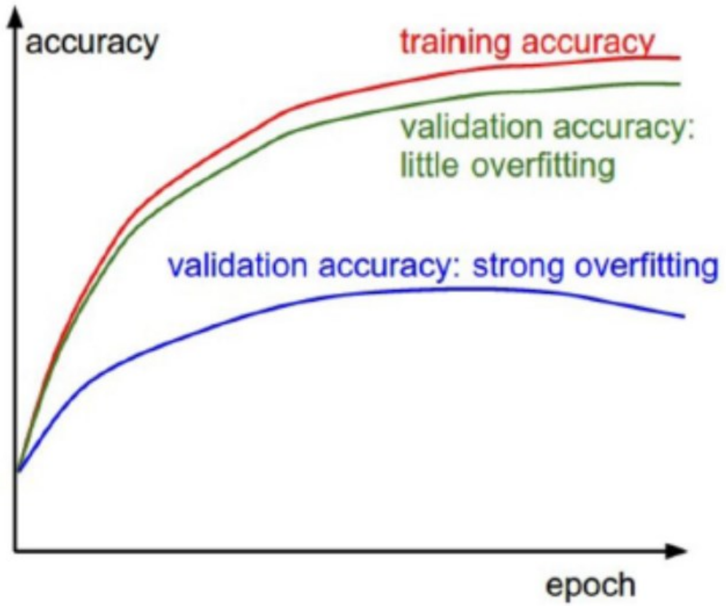
  Loss = max(0, 1-(score(x)-score(x')))

Regularization

- L1
- L2
- Elastic Net
- DropOut
- Batch Normalization
- Layer Normalization
- Problem-specific regularizations
- Early Stopping
- https://towardsdatascience.com/different-normalization-layers-in-deep-learning-1a7214ff71d6

Some Practical Advice

- Gradient check on small data
- Overfit without regularization on small data.
- Decay learning rate with time
- Regularize
- Always check learning curves



Optimization

- Stochastic Gradient Descent
- Mini-Batch Gradient Descent
- AdaGrad
- AdaDelta
- RMSProp
- Adam

Image 2: SGD without momentum

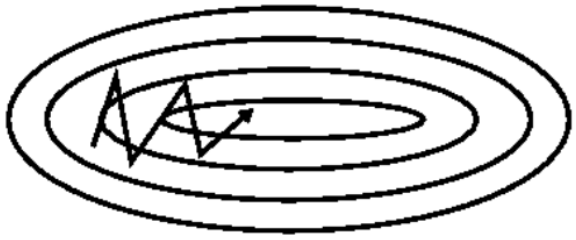


Image 3: SGD with momentum

Learning rate schedules

https://ruder.io/optimizing-gradient-descent/

Glorot/Xavier Initialization (tanh)

- Initializing W matrix of dimensionality d_{in}xd_{out}

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}, +\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}\right],$$

He's Initialization (relu)

$$W \sim G(0, \frac{2}{d_{in}+d_{out}})$$

Batching

- Padding

Vanishing and Exploding Gradients

- Clipping