WordNet

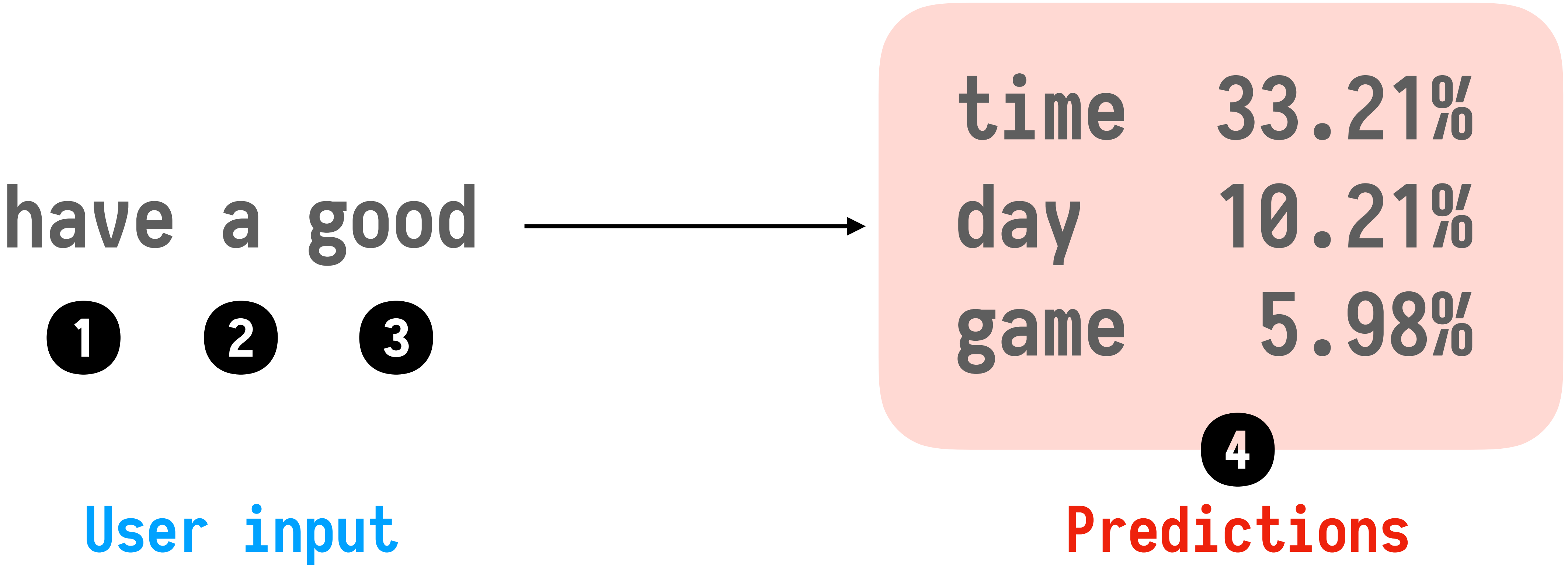
Understanding the basic math behind deep learning

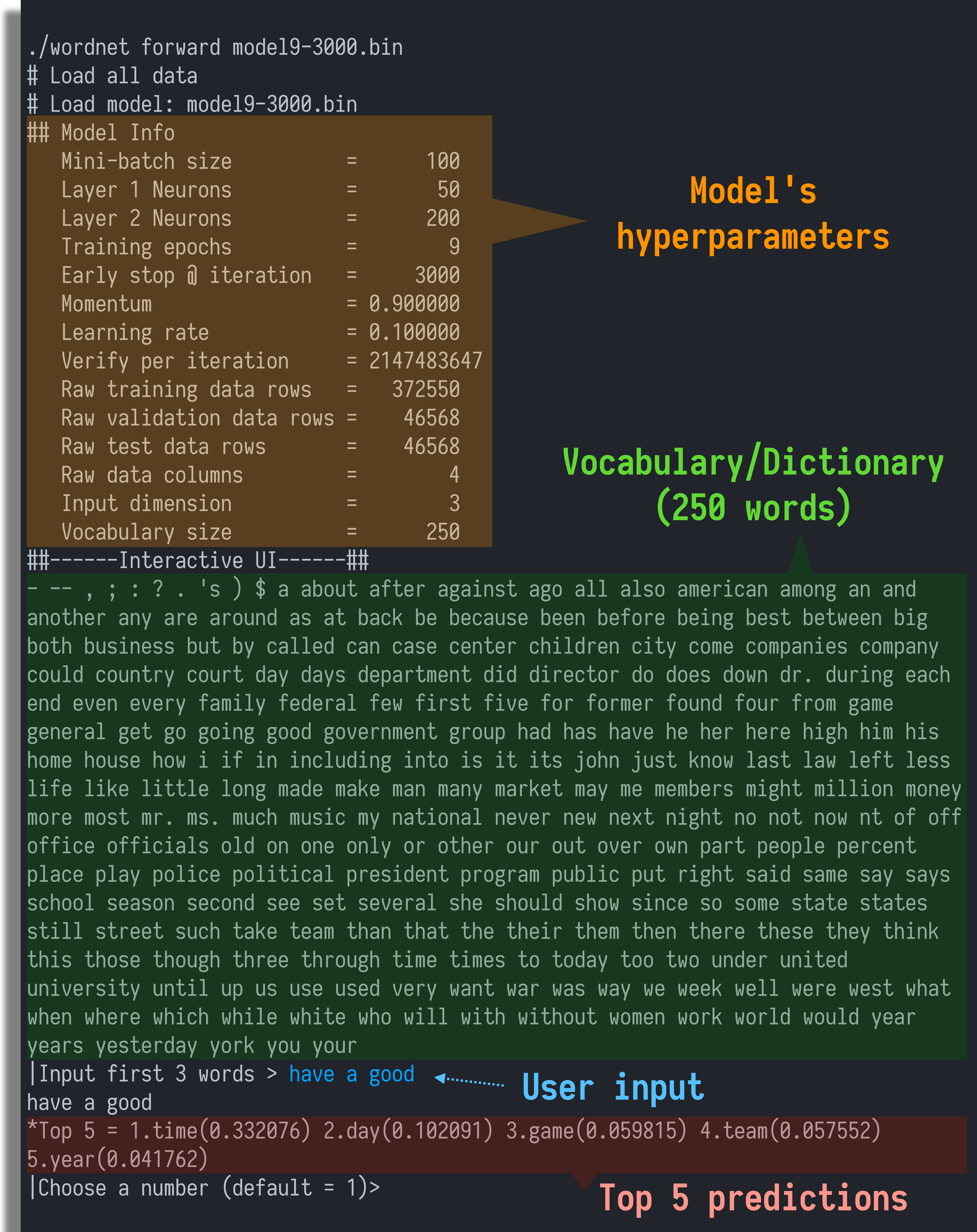
https://github.com/gingerbig/wordnet

Background

So basically, what's WordNet?

- An assignment of Hinton's Coursera Course *Neural Networks for Machine Learning*
- A very simple network that reads 3 words and predicts the next one
- Rewritten in C with standard libraries; no extra libs required





```
./wordnet forward model9-3000.bin
# Load all data
# Load model: model9-3000.bin
## Model Info
   Mini-batch size            =       100
   Layer 1 Neurons            =        50
   Layer 2 Neurons            =       200
   Training epochs            =         9
   Early stop @ iteration     =      3000
   Momentum                   =  0.900000
   Learning rate              =  0.100000
   Verify per iteration       = 2147483647
   Raw training data rows     =    372550
   Raw validation data rows   =     46568
   Raw test data rows         =     46568
   Raw data columns           =         4
   Input dimension            =         3
   Vocabulary size            =       250
##------Interactive UI------##
- -- , ; : ? . 's ) $ a about after against ago all also american among an and
another any are around as at back be because been before being best between big
both business but by called can case center children city come companies company
could country court day days department did director do does down dr. during each
end even every family federal few first five for former found four from game
general get go going good government group had has have he her here high him his
home house how i if in including into is it its john just know last law left less
life like little long made make man many market may me members might million money
more most mr. ms. much music my national never new next night no not now nt of off
office officials old on one only or other our out over own part people percent
place play police political president program public put right said same say says
school season second see set several she should show since so some state states
still street such take team than that the their them then there these they think
this those though three through time times to today too two under united
university until up us use used very want war was way we week well were west what
when where which while white who will with without women work world would year
years yesterday york you your
|Input first 3 words > have a good
have a good
*Top 5 = 1.time(0.332076) 2.day(0.102091) 3.game(0.059815) 4.team(0.057552)
5.year(0.041762)
|Choose a number (default = 1)>
```

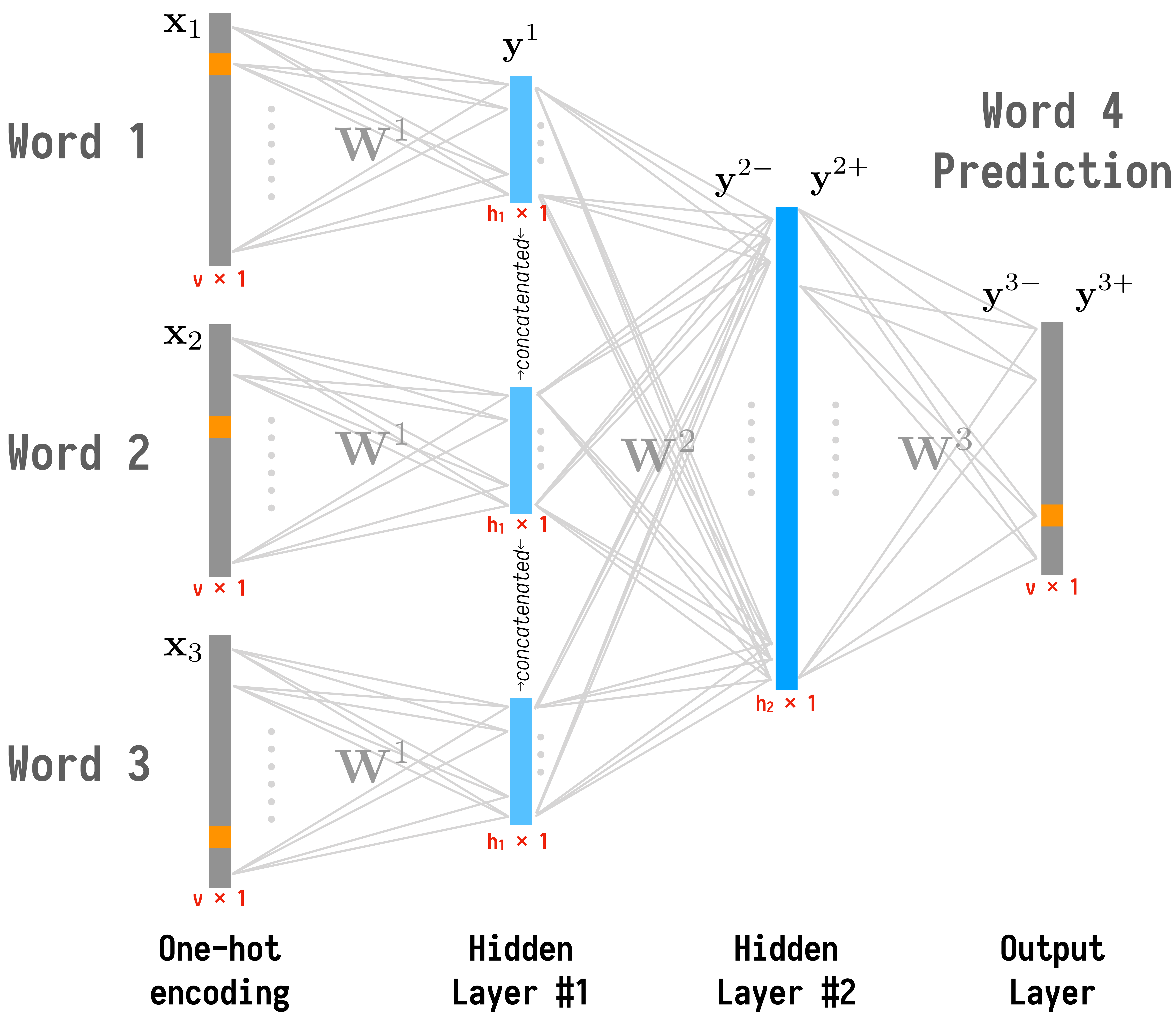
Model's hyperparameters

Vocabulary/Dictionary (250 words)

User input

Top 5 predictions

Layout of WordNet



$$\underset{(D\times h_1)\times 1}{\mathbf{y}^1} = \begin{bmatrix} \underset{h_1\times v}{\mathbf{W}^1}\underset{v\times 1}{\mathbf{x}_1} \\ \underset{h_1\times v}{\mathbf{W}^1}\underset{v\times 1}{\mathbf{x}_2} \\ \underset{h_1\times v}{\mathbf{W}^1}\underset{v\times 1}{\mathbf{x}_3} \end{bmatrix}$$

Input

$$\underset{h_2\times 1}{\mathbf{y}^{2-}} = \underset{h_2\times(D\times h_1)}{\mathbf{W}^2}\underset{(D\times h_1)\times 1}{\mathbf{y}^1} + \underset{h_2\times 1}{\mathbf{b}^2}$$

$$\underset{h_2\times 1}{\mathbf{y}^{2+}} = \boldsymbol{\sigma}(\underset{h_2\times 1}{\mathbf{y}^{2-}})$$

$$\underset{v\times 1}{\mathbf{y}^{3-}} = \underset{v\times h_2}{\mathbf{W}^3}\underset{h_2\times 1}{\mathbf{y}^{2+}} + \underset{v\times 1}{\mathbf{b}^3}$$

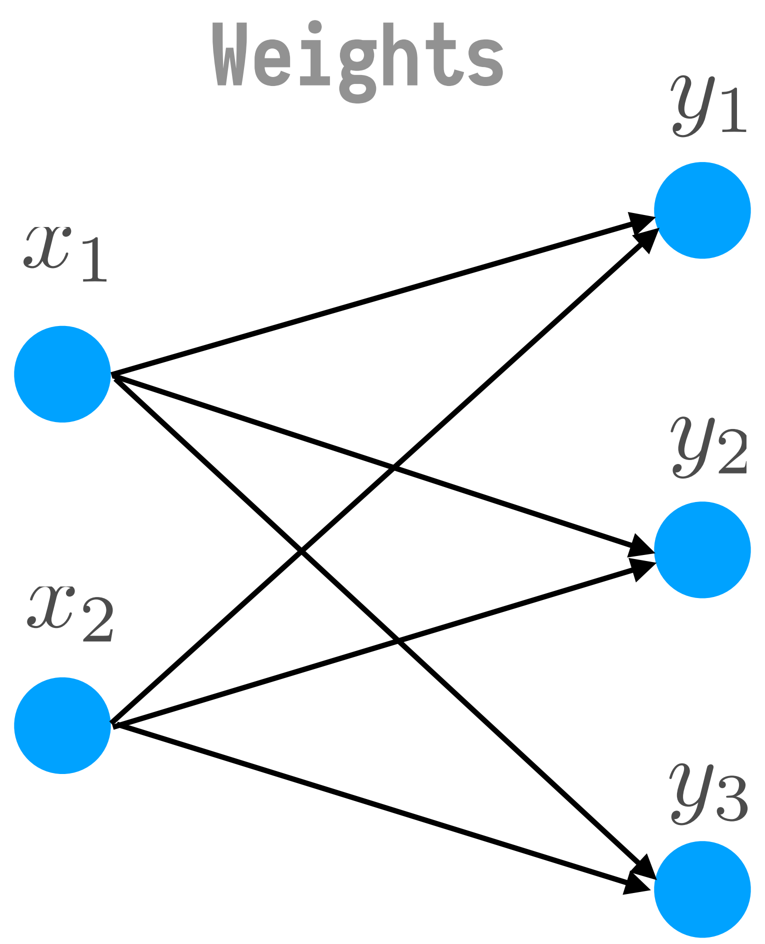
Output

$$\underset{v\times 1}{\mathbf{y}^{3+}} = \mathbf{s}(\underset{v\times 1}{\mathbf{y}^{3-}})$$

Symbols

v	Vocabulary size	250
D	Input dimension (words)	3
h₁	Layer #1's neurons	50
h₂	Layer #2's neurons	200
σ()	Sigmoid function	/
s()	Softmax function	/

Concept #1: Fully connected layer



Input
state values

Output
state values

What you see in a paper

$$y_1 = \sigma(w_{11}x_1 + w_{12}x_2 + b_1)$$
$$y_2 = \sigma(w_{21}x_1 + w_{22}x_2 + b_2)$$
$$y_3 = \sigma(w_{31}x_1 + w_{32}x_2 + b_3)$$

Activation Function

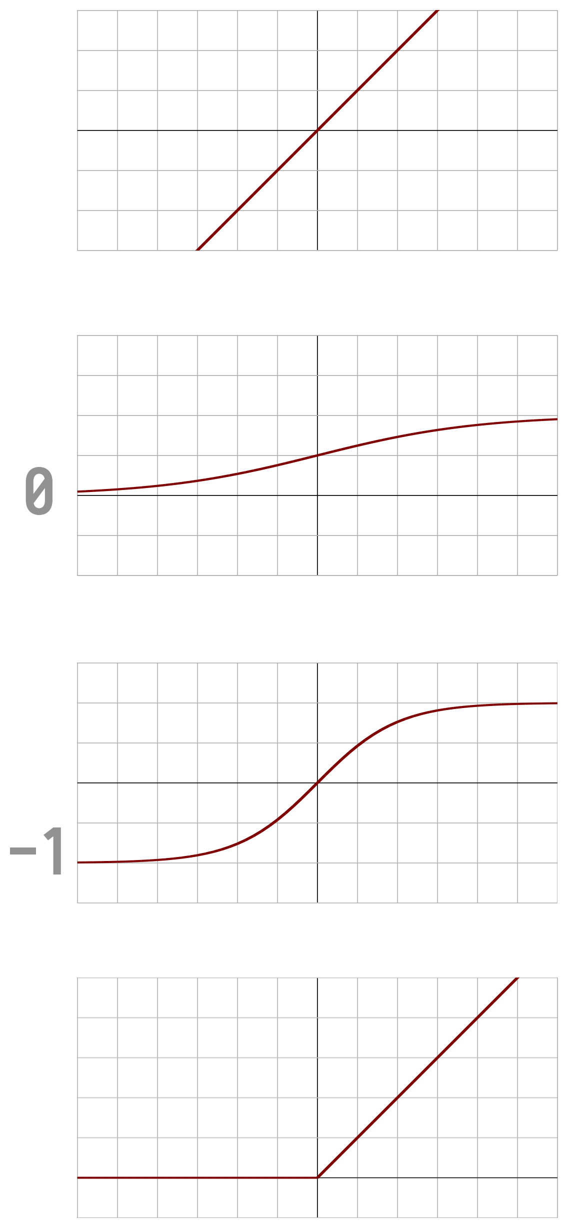
Biases

Two-step expression

Useful for deriving formulae

$$\mathbf{y}_- = \mathbf{W}\mathbf{x} + \mathbf{b}$$
$$\mathbf{y}_+ = \boldsymbol{\sigma}(\mathbf{y}_-)$$

What it actually does



Identity

WordNet uses it

Sigmoid
(Logistic)

TanH

ReLU

Activation functions?

https://en.wikipedia.org/wiki/Activation_function

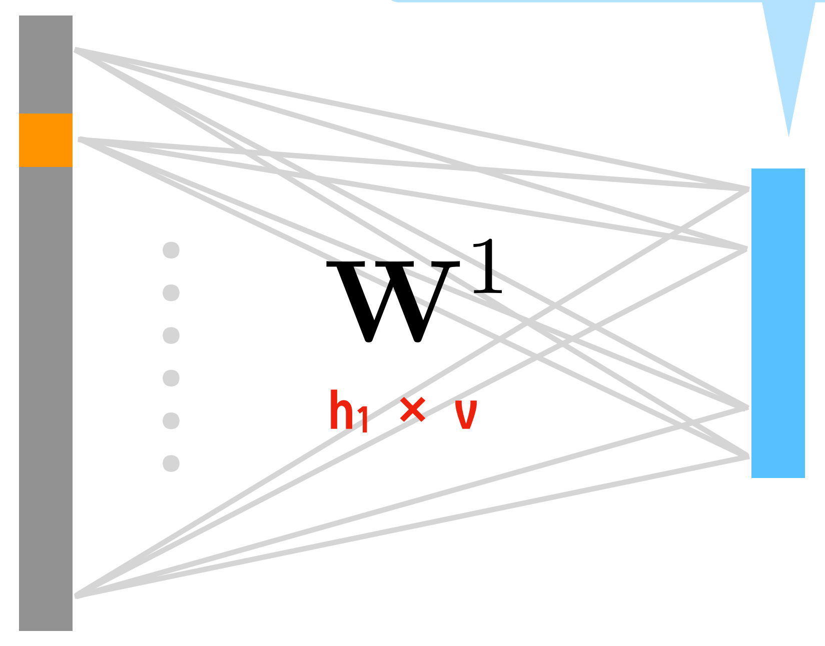
To introduce nonlinearity!

Concept #2: One-hot vector and text embedding

How can we express a word in a neural network?

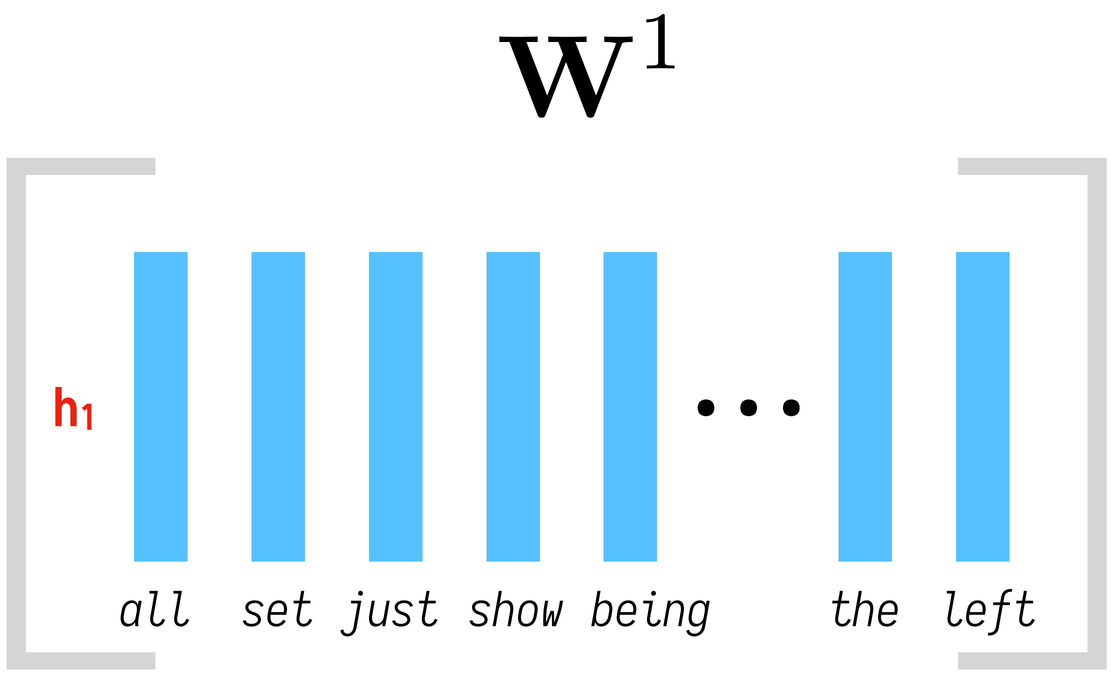
"just" is

Vocabulary	One-hot vector
all	0
set	0
just	1
show	0
being	0
money	0
over	0
both	0
years	0
four	0



Text embedding
A **dense** vector

Dimension = Vocabulary size
A **sparse** vector

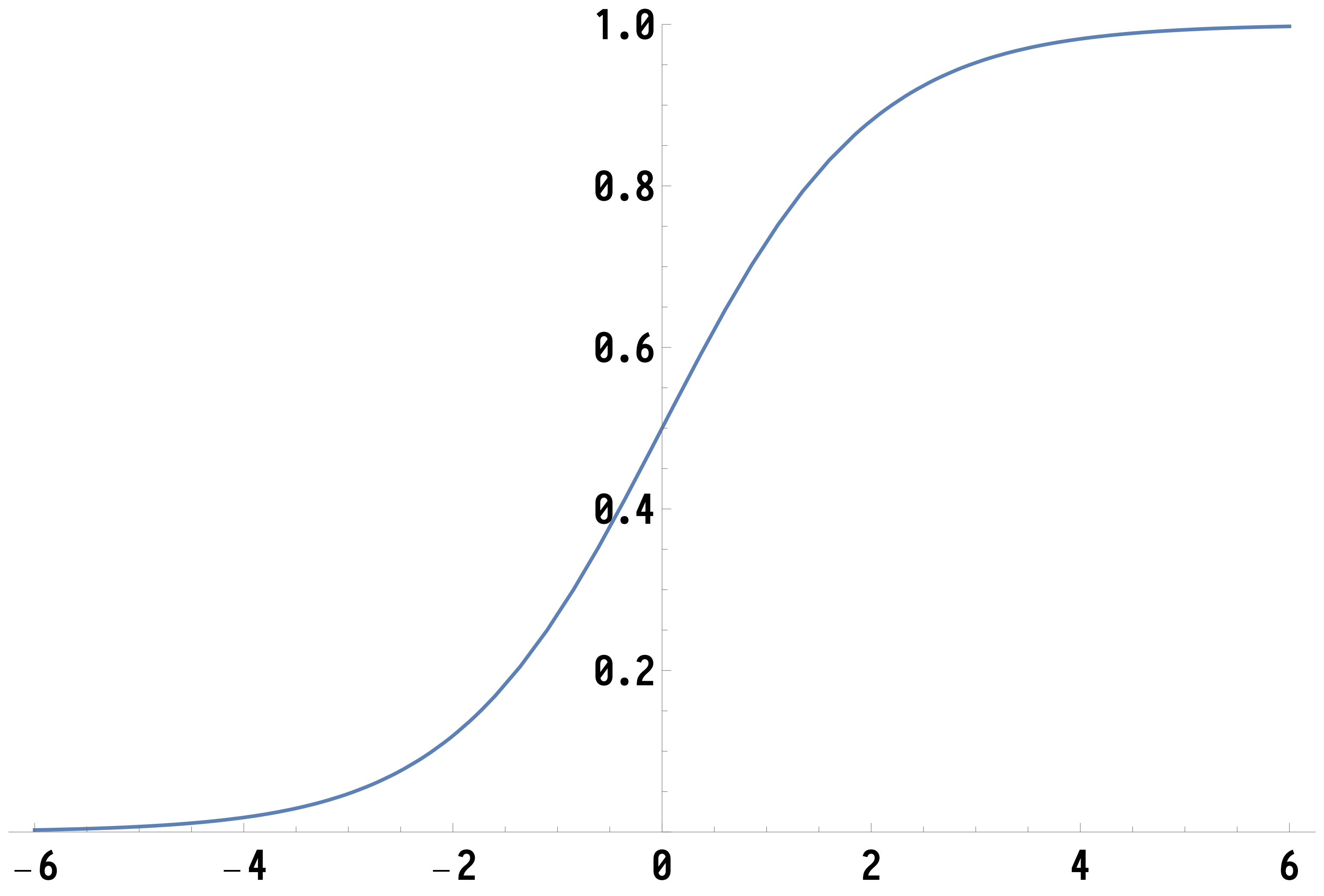


column no. = vocabulary size

- Each column is a "word embedding" for the vocabulary
- We can view $\mathbf{W}^1\mathbf{x}_1$ as the one-hot $\mathbf{x}_1$ selects a corresponding column

$$\underbrace{\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}}_{\mathbf{W}^1}\underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{\mathbf{x}_1}=\underbrace{\begin{bmatrix}2\\ 4\\ 6\end{bmatrix}}_{2^{nd}\text{ column of } W^1}$$

Concept #3: Sigmoid/logistic function



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- Nonlinear
- Squash a real value within (0, 1)

Convenient property

$$\sigma = \frac{1}{1+e^{-x}}$$

$$\frac{d\sigma}{dx} = \frac{0-(-e^{-x})}{(1+e^{-x})^2}$$

$$= \frac{1+e^{-x}-1}{(1+e^{-x})^2}$$

$$= \frac{1}{1+e^{-x}} - \left(\frac{1}{1+e^{-x}}\right)^2$$

$$= \sigma - \sigma^2$$

$$= \sigma(1-\sigma)$$

Vector form

$$\boldsymbol{\sigma}(\mathbf{x}) \downarrow \begin{bmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \vdots \\ \sigma(x_n) \end{bmatrix}$$

Concept #4: Softmax function

$$s(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

For probability estimation

1 $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ 2 $\begin{bmatrix} e^{x_1} \\ e^{x_2} \\ \vdots \\ e^{x_n} \end{bmatrix}$ 3 $\mathrm{sum} = e^{x_1} + e^{x_2} + \cdots + e^{x_n}$ 4 $\begin{bmatrix} e^{x_1}/\mathrm{sum} \\ e^{x_2}/\mathrm{sum} \\ \vdots \\ e^{x_n}/\mathrm{sum} \end{bmatrix}$

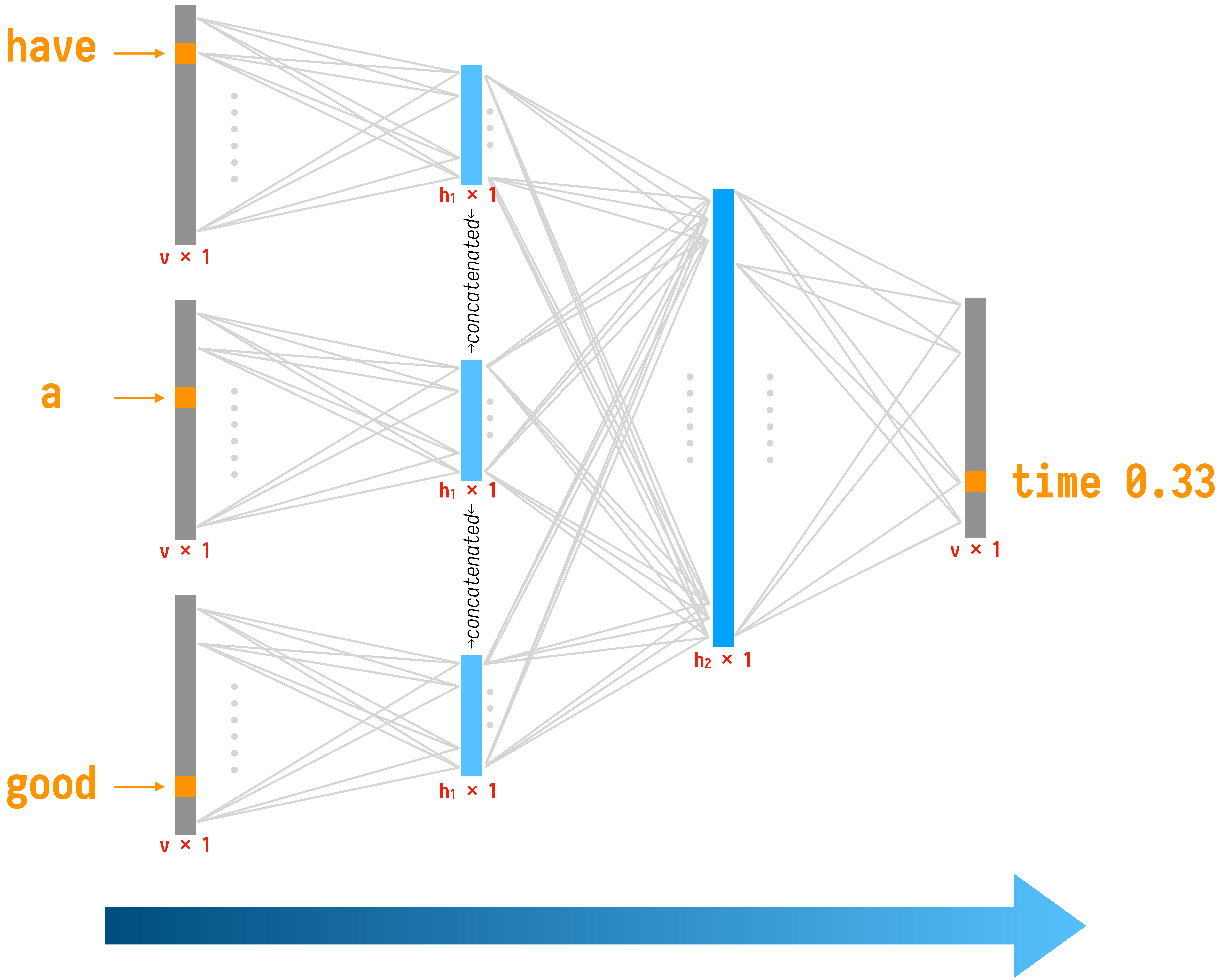
Input **Output**

... to avoid overflow, the actual calculation is

1 $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ 2 $\mathbf{z} = \begin{bmatrix} x_1 - x_{\max} \\ x_2 - x_{\max} \\ \vdots \\ x_n - x_{\max} \end{bmatrix}$ 3 $\mathbf{s}(\mathbf{x}) = \mathbf{s}(\mathbf{z})$

$$\begin{aligned} s(\mathbf{z})_i &= \frac{e^{x_i - c}}{\sum_{j=1}^{n} e^{x_j - c}} \\ &= \frac{e^{x_i} \cancel{e^{-c}}}{\sum_{j=1}^{n} e^{x_j} \cancel{e^{-c}}} \\ &= \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \end{aligned}$$

Concept #5: Inference (forward propagation)



$$\mathbf{y}^{1} = \begin{bmatrix} \mathbf{W}^{1}\mathbf{x}_1 \\ \mathbf{W}^{1}\mathbf{x}_2 \\ \mathbf{W}^{1}\mathbf{x}_3 \end{bmatrix}$$

$$\mathbf{y}^{2-} = \mathbf{W}^{2}\mathbf{y}^{1} + \mathbf{b}^{2}$$

$$\mathbf{y}^{2+} = \boldsymbol{\sigma}(\mathbf{y}^{2-})$$

$$\mathbf{y}^{3-} = \mathbf{W}^{3}\mathbf{y}^{2+} + \mathbf{b}^{3}$$

$$\mathbf{y}^{3+} = \mathbf{s}(\mathbf{y}^{3-})$$

Treated as constants 🟩

Concept #6: Training

What does training do?

- Find out suitable values for all the **parameters**

How do we know if a set of parameters are good?

- Use a proper **loss function**
- In WordNet, the loss function is *cross entropy*

$$L = \sum_{i=1}^{v} -y_i^t \log y_i^{3+}$$

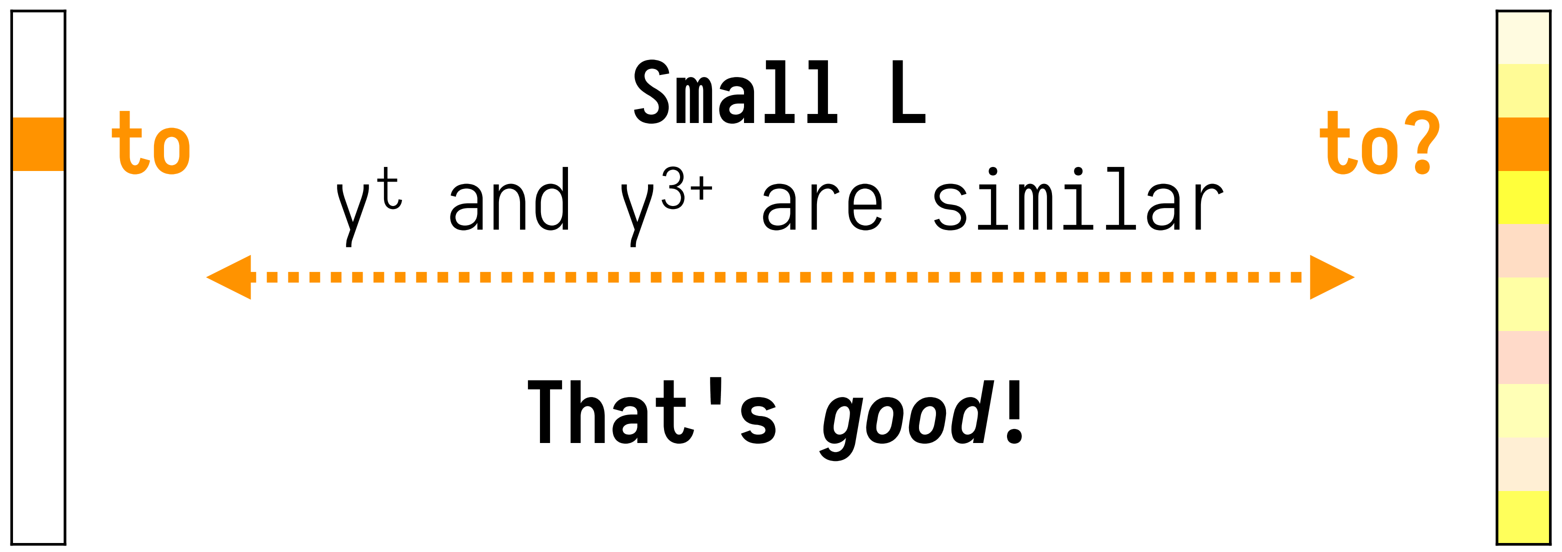
Ground truth (target)

Inference result

Text corpus

we are going to do this our way

- (we are going) → to
- (are going to) → do
- (going to do) → this
- ...



Small L

y^t and y^{3+} are similar

That's *good*!

Training data

$$\mathbf{y}^1 = \begin{bmatrix} \mathbf{W}^1 \mathbf{x}_1 \\ \mathbf{W}^1 \mathbf{x}_2 \\ \mathbf{W}^1 \mathbf{x}_3 \end{bmatrix} \begin{array}{l} \leftarrow \text{we} \\ \leftarrow \text{are} \\ \leftarrow \text{going} \end{array}$$

$$\mathbf{y}^{2-} = \mathbf{W}^2 \mathbf{y}^1 + \mathbf{b}^2$$

$$\mathbf{y}^{2+} = \boldsymbol{\sigma}(\mathbf{y}^{2-})$$

$$\mathbf{y}^{3-} = \mathbf{W}^3 \mathbf{y}^{2+} + \mathbf{b}^3$$

$$\leftarrow \underset{v \times 1}{\mathbf{y}^{3+}} = \mathbf{s}(\mathbf{y}^{3-})$$

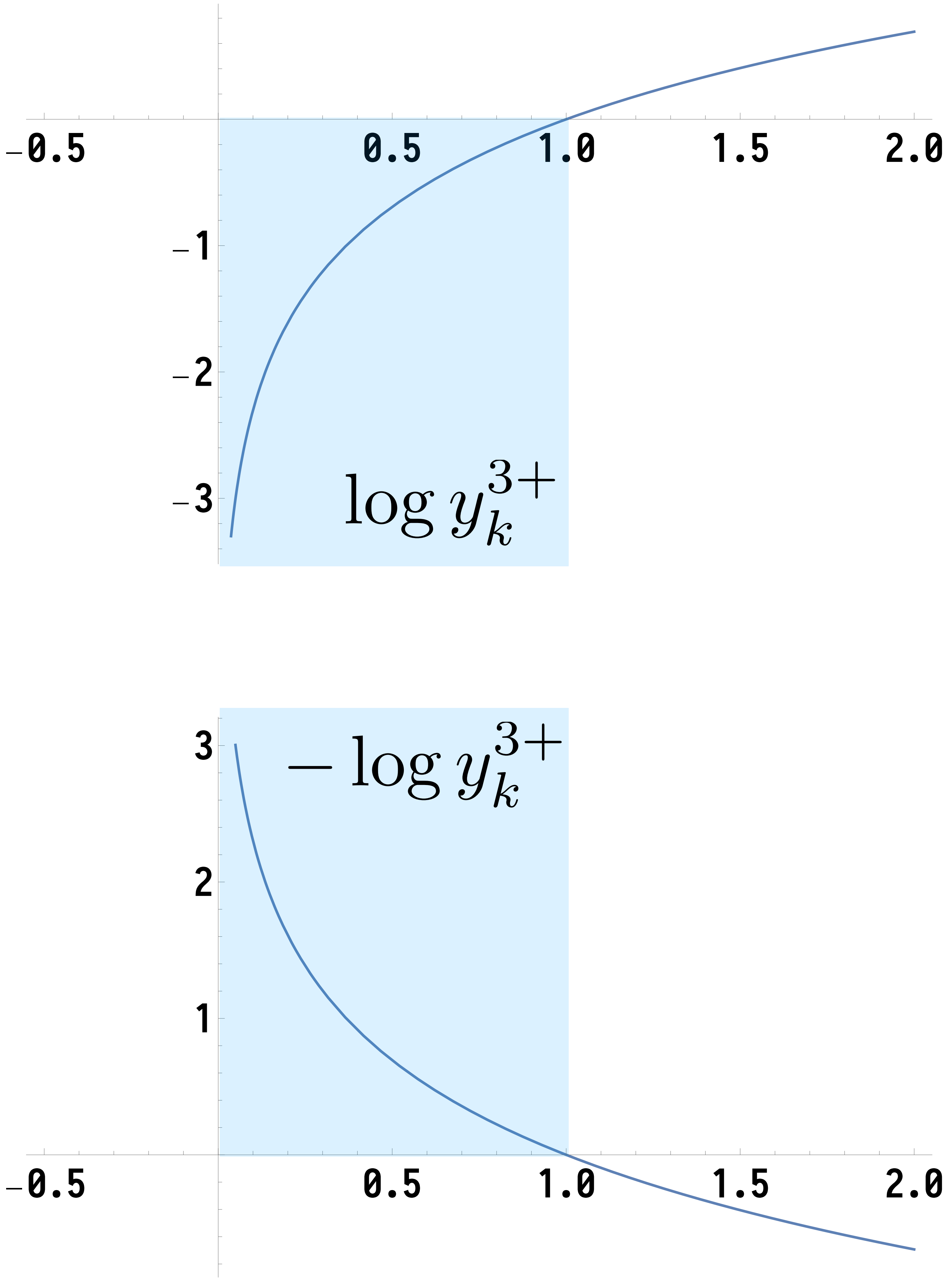
Parameters ▨

Off-topic: How *min* L pushes y^{3+} towards y

- y^{3+}: as the output of the softmax function, its entries are constrained within [0, 1], i.e. we have $0 \leq y^{3+}_i \leq 1$
- y: ground truth one-hot vector; assume $y_k = 1$, other entries are all 0
- Then, the loss function becomes

$$L = -y_k \log y_k^{3+} = -\log y_k^{3+}$$

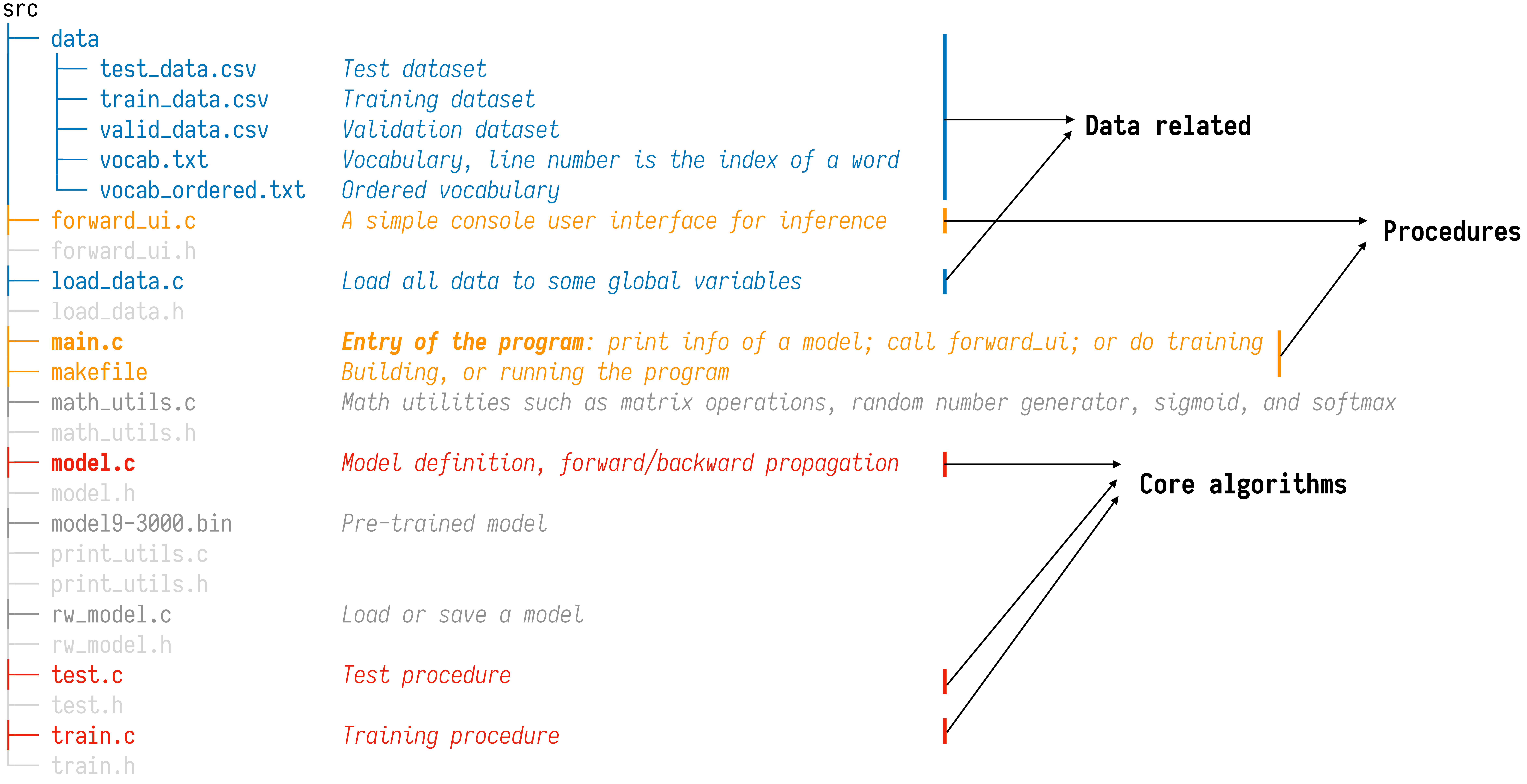
So, obviously if and only if $y^{3+}_k = 1$, the loss function arrives at its minimum. ∎



$$0 \leq y_k^{3+} \leq 1$$

... and one more thing for pedants: $\lim_{y \to 0} y \log y = 0$

Project structure



```
src
├── data
│   ├── test_data.csv        Test dataset
│   ├── train_data.csv       Training dataset
│   ├── valid_data.csv       Validation dataset
│   ├── vocab.txt            Vocabulary, line number is the index of a word
│   └── vocab_ordered.txt    Ordered vocabulary
├── forward_ui.c             A simple console user interface for inference
├── forward_ui.h
├── load_data.c              Load all data to some global variables
├── load_data.h
├── main.c                   Entry of the program: print info of a model; call forward_ui; or do training
├── makefile                 Building, or running the program
├── math_utils.c             Math utilities such as matrix operations, random number generator, sigmoid, and softmax
├── math_utils.h
├── model.c                  Model definition, forward/backward propagation
├── model.h
├── model9-3000.bin          Pre-trained model
├── print_utils.c
├── print_utils.h
├── rw_model.c               Load or save a model
├── rw_model.h
├── test.c                   Test procedure
├── test.h
├── train.c                  Training procedure
└── train.h
```

Model.h: Model definition

```
typedef struct _WordNet {
  HyperParameter hp;

  int D;                                    Alias inputDimension
  int N;                                    Alias miniBatchSize
  int vocabSize;                            Alias v
  int h1;                                   Alias layer1Neurons
  int h2;                                   Alias layer2Neurons
  double *W1;                               W¹
  double *dW1;                              Change of W¹
  double *inputWordVectorBatch;             ➥
  double *xit;                              Buffer x₁ᵀ, x₂ᵀ, or x₃ᵀ
  double *bufferW1xInputWordVectorBatch;    ➥
  double *layer1StateBatch;                 Batch of y¹
  double *y1t;                              Batch of y¹ᵀ
  double *W2;                               W²
  double *dW2;                              Change of W²
  double *W2t;                              W²ᵀ
  double *layer2StateBatch;                 Batch of y²
  double *y2t;                              Batch of y²ᵀ
  double *bias2;                            b²
  double *db2;                              Change of b²
  double *W3;                               W³
  double *dW3;                              Change of W³
  double *W3t;                              W³ᵀ
  double *layer3StateBatch;                 y³
  double *bias3;                            b³
  double *db3;                              Change of b³
  const double *outputStateBatch;           Alias y³
  double *targetVectorBatch;                Alias yᵗ
  const double *yt;                         ➥

  double *dLdy3_;                           ∂L/∂y³⁻
  double *dLdy3_t;                          (∂L/∂y³⁻)ᵀ
  double *dLdW3;                            ∂L/∂W³
  double *dLdb3;                            ∂L/∂b³
  double *dLdy2;                            ∂L/∂y²⁺
  double *dLdy2_;                           ∂L/∂y²⁻
  double *dLdb2;                            ∂L/∂b²
  double *dLdW2;                            ∂L/∂W²
  double *dLdy1;                            ∂L/∂y¹
  double *dLdy1i;                           ➥ See derivation
  double *dLdW1;                            ∂L/∂W¹
} WordNet;
```

```
typedef struct _HyperParameter {
  // Tunable
  int miniBatchSize;              N
  int layer1Neurons;              h₁
  int layer2Neurons;              h₂
  int epoch;                      Epoch
  int earlyStopIteration;         Early stop
  double momentum;                Momentum
  double learningRate;            Learning Rate
  int verifyPerIterBatch;

  // Fixed
  int rawDataRow4Training;        372550
  int rawDataRow4Validation;      46568
  int rawDataRow4Test;            46568
  int rawDataColumn;              4
  int inputDimension;             = D = 3
  int vocabSize;                  250
} HyperParameter;
```

$$\underset{(D\times h_1)\times 1}{\mathbf{y}^1} = \overset{h_1\times v}{\left[\begin{array}{c}\mathbf{W}^1\mathbf{x}_1\\ \mathbf{W}^1\mathbf{x}_2\\ \mathbf{W}^1\mathbf{x}_3\end{array}\right]}\begin{array}{c}v\times 1\\ v\times 1\\ v\times 1\end{array}$$

$$\underset{h_2\times 1}{\mathbf{y}^{2-}} = \underset{h_2\times(D\times h_1)}{\mathbf{W}^2}\underset{(D\times h_1)\times 1}{\mathbf{y}^1} + \underset{h_2\times 1}{\mathbf{b}^2}$$

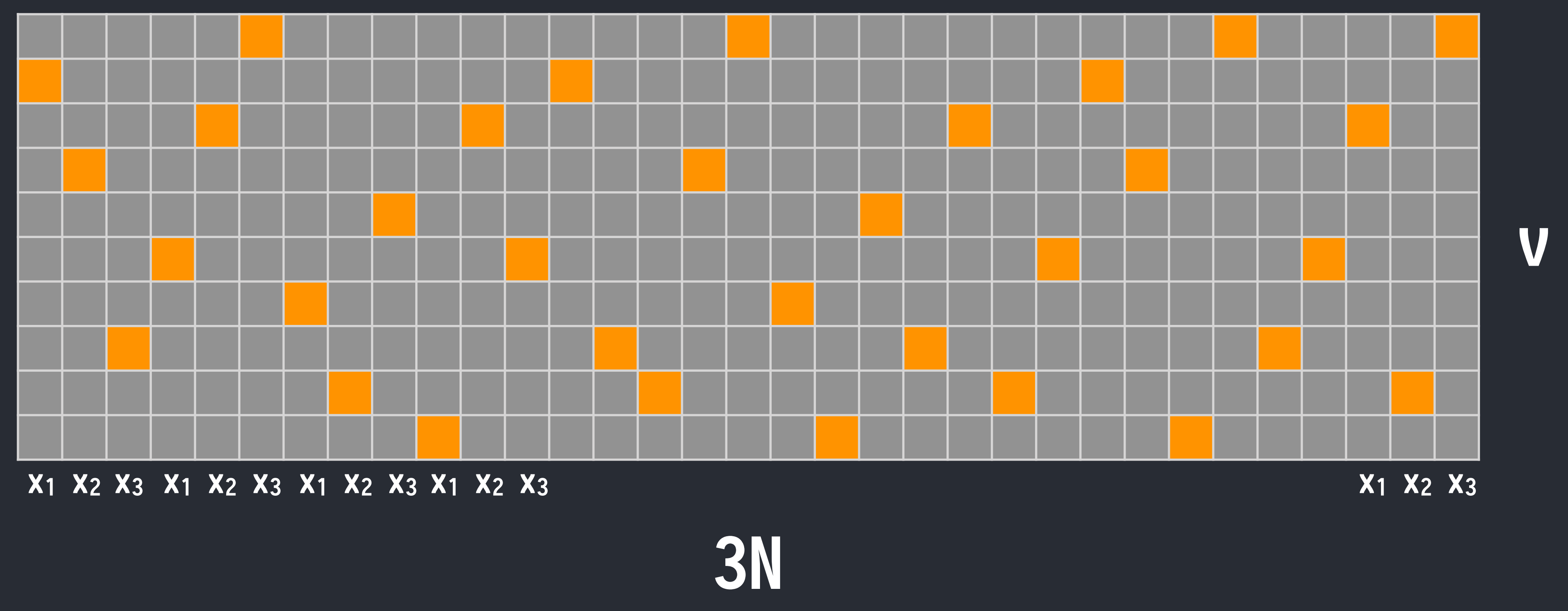
$$\underset{h_2\times 1}{\mathbf{y}^{2+}} = \sigma(\underset{h_2\times 1}{\mathbf{y}^{2-}})$$

$$\underset{v\times 1}{\mathbf{y}^{3-}} = \underset{v\times h_2}{\mathbf{W}^3}\underset{h_2\times 1}{\mathbf{y}^{2+}} + \underset{v\times 1}{\mathbf{b}^3}$$

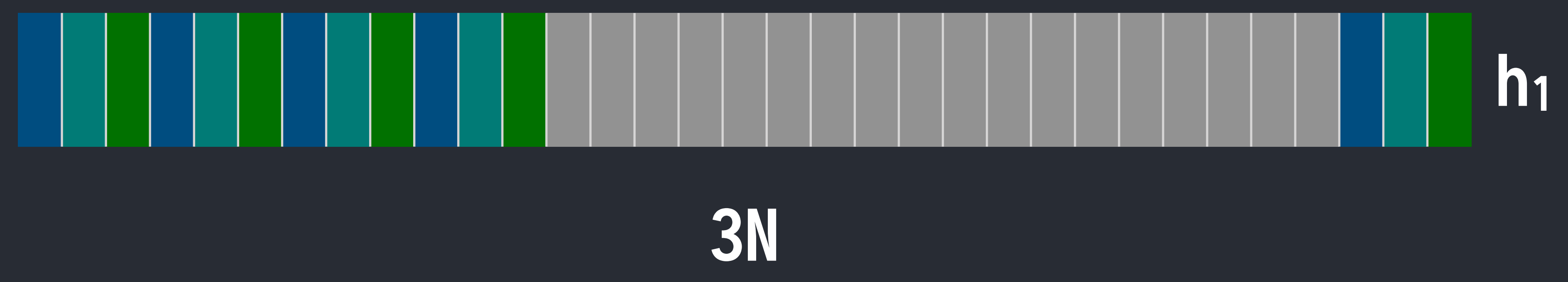
$$\underset{v\times 1}{\mathbf{y}^{3+}} = \mathbf{s}(\underset{v\times 1}{\mathbf{y}^{3-}})$$

$$\min L = \sum_{i=1}^{v} -y_i^t \log y_i^{3+}$$

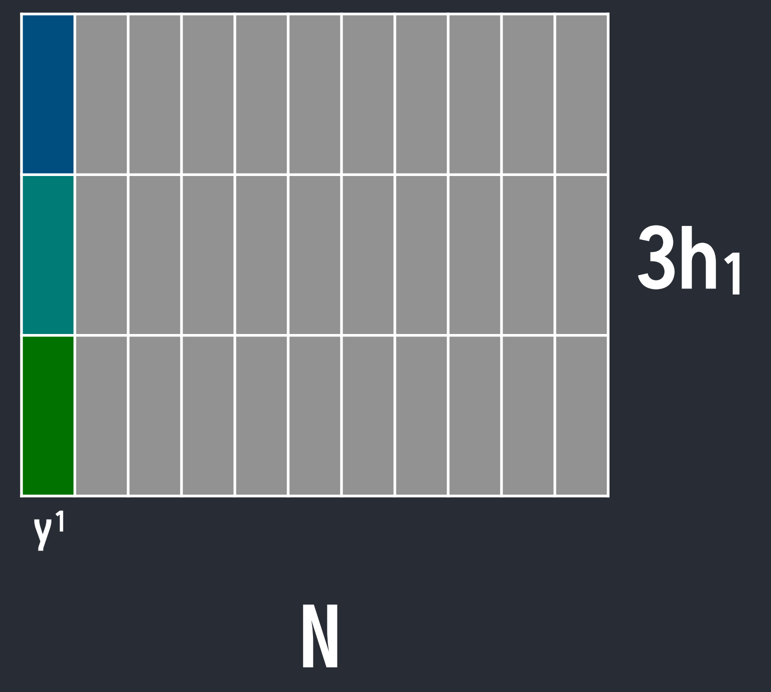
inputWordVectorBatch



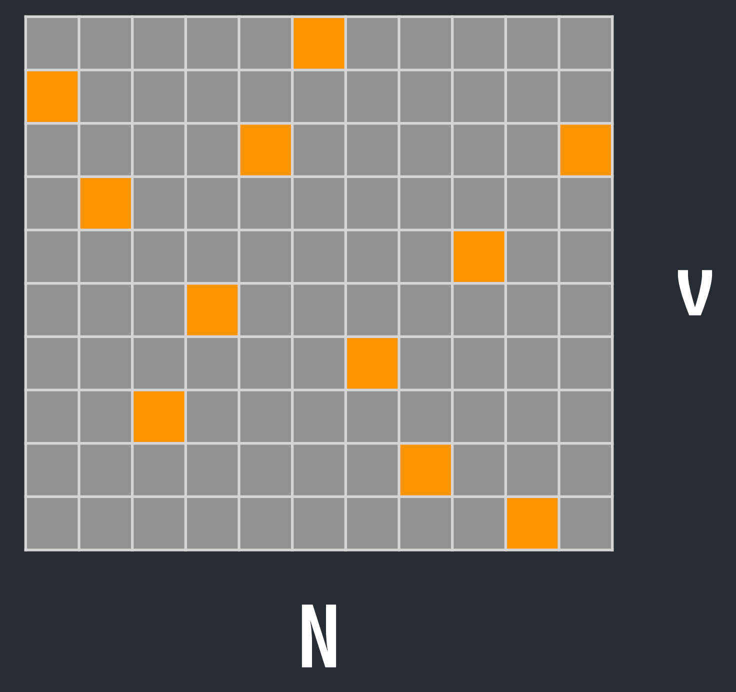
bufferW1xInputWordVectorBatch = W¹ * inputWordVectorBatch



layer1StateBatch, i.e. y¹ batch

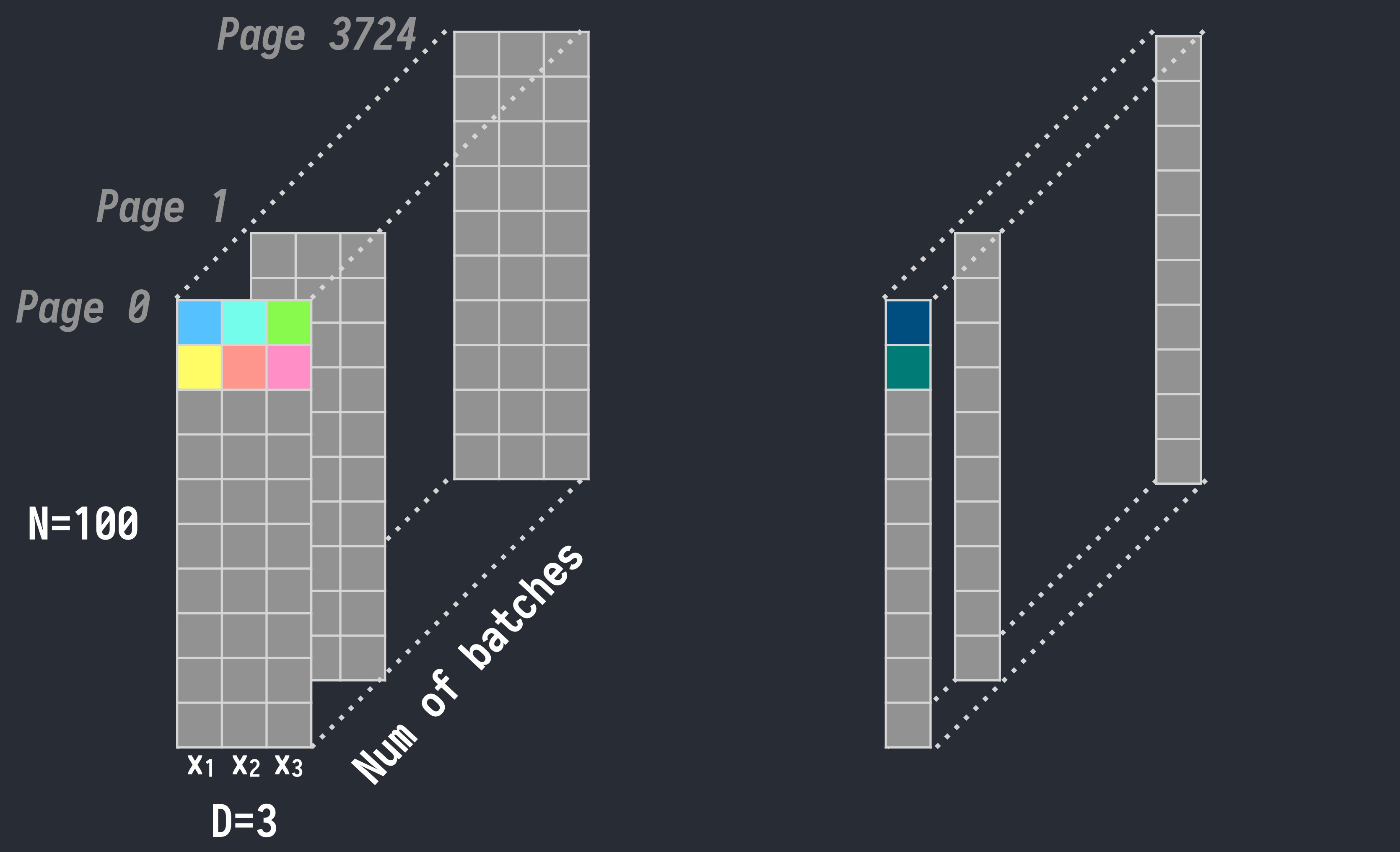


yt, target batch



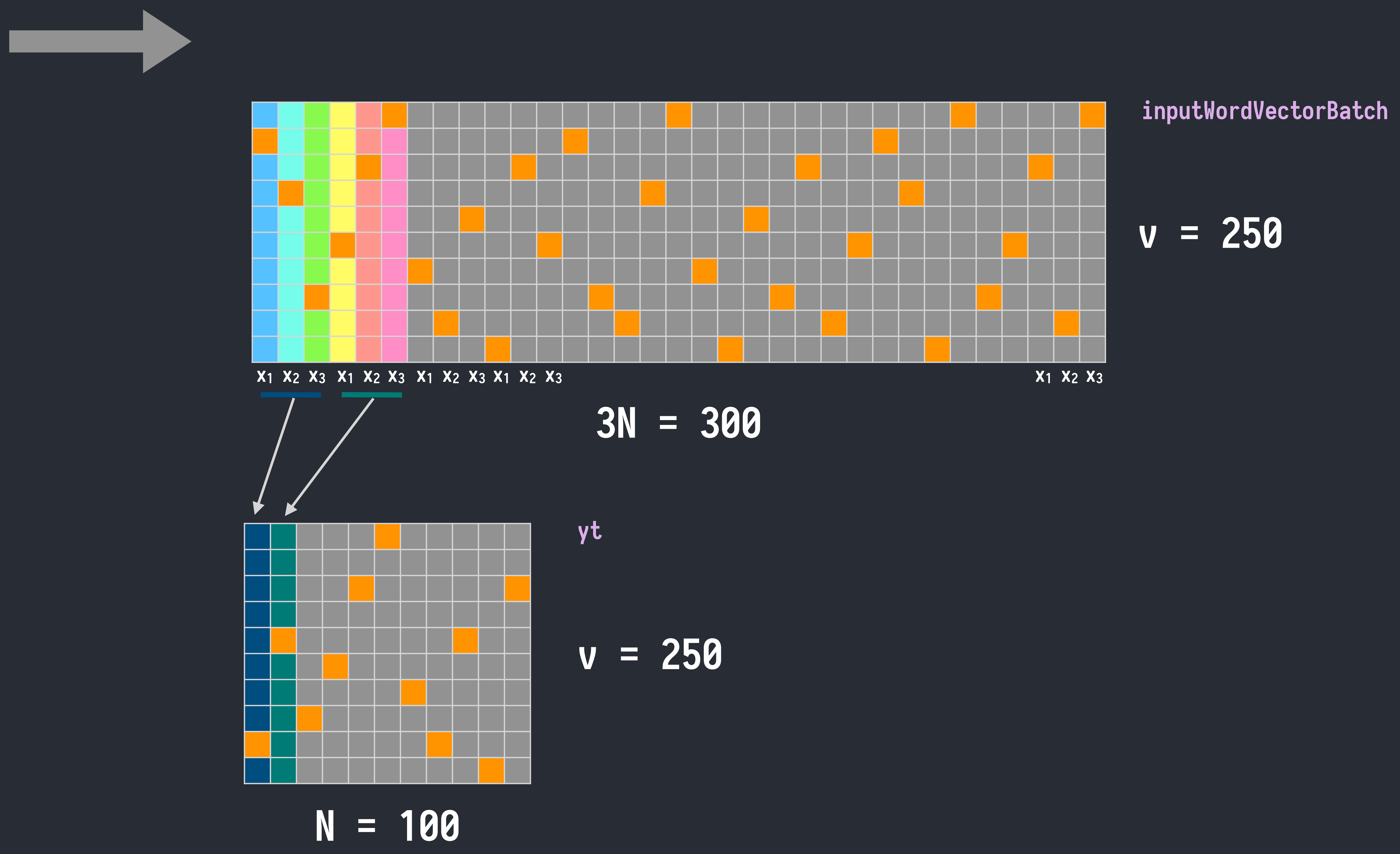
Mini-batches of matrices in C

Training data (372550 × 4)

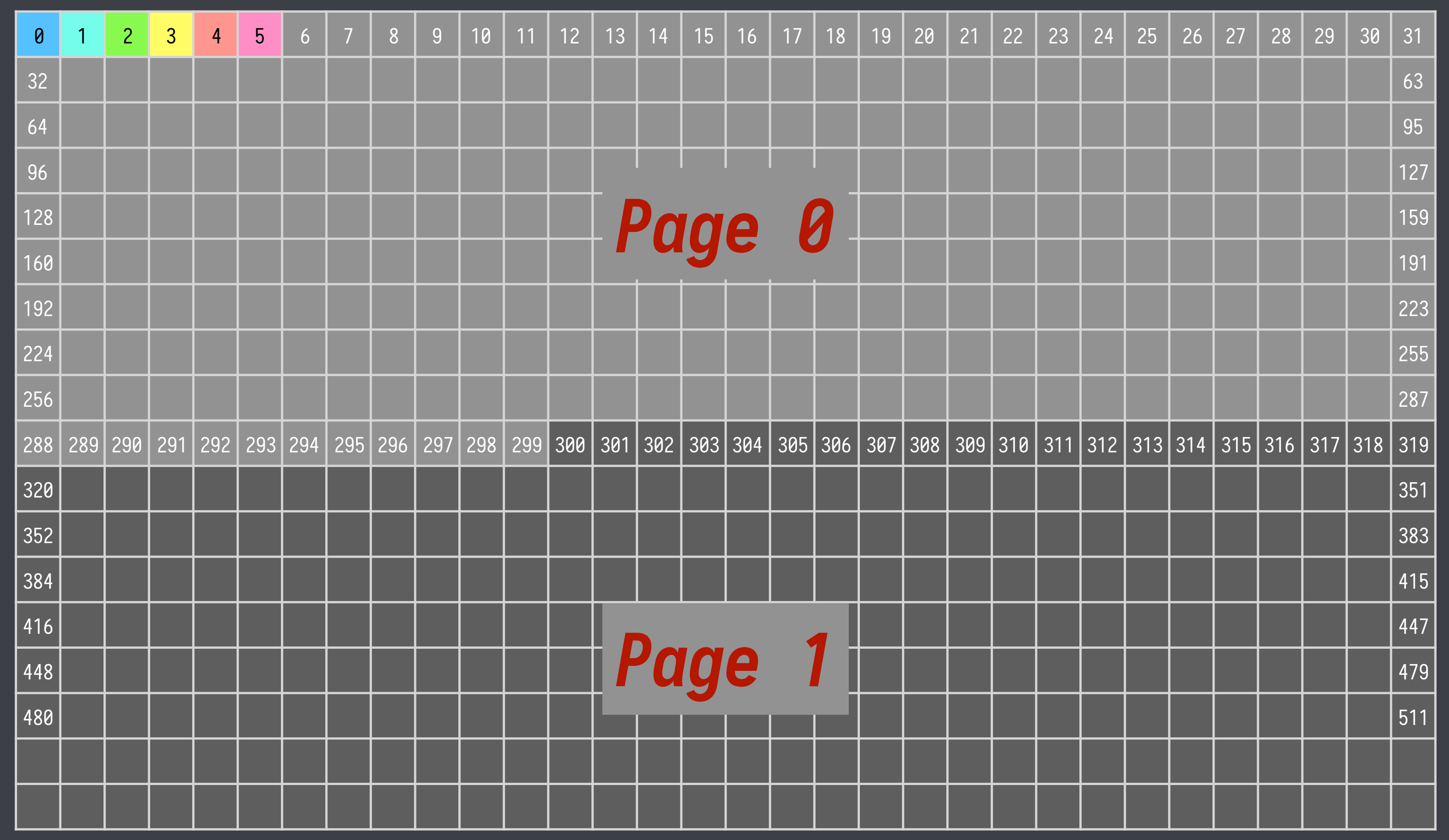


double *batchInput4Training   double *batchTarget4Training

Read a page and fill



Continuous storage in memory



So, how can I reference batchInput4Training's entry on page *k*, at row *i* and column *j*?

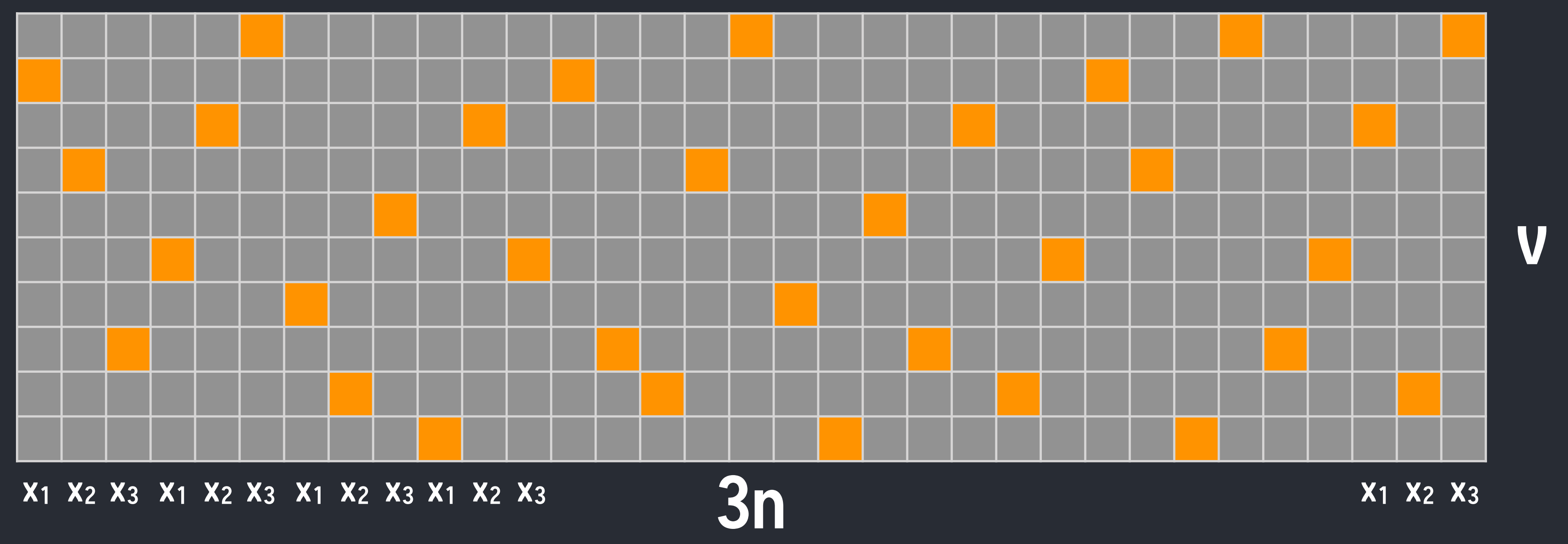
- batchInput4Training[*k* * N * D + *i* * D + *j*]

Model.c: Inference

```
void forwardPropagate(WordNet *model, const int *miniBatchInput, int n) {
  int i, j, k, l, index;
  int N = n;
  int D = model->D;

  memset(model->inputWordVectorBatch, 0,
         model->vocabSize * D * N * sizeof(double));
  k = 0;
  for (i = 0; i < N; ++i) {
    for (j = 0; j < D; ++j) {
      index = miniBatchInput[i * D + j] - 1; // MATLAB index starts from 1
      model->inputWordVectorBatch[index * (D * N) + k] = 1.0;
      ++k;
    }
  }
  multiplyMatrix(model->W1, model->inputWordVectorBatch, model->h1,
                 model->vocabSize, D * N, model->bufferW1xInputWordVectorBatch);
  k = 0, l = 0;
  for (j = 0; j < D * N; ++j) {
    for (i = 0; i < model->h1; ++i) {
      model->layer1StateBatch[k++ * N + l] =
        model->bufferW1xInputWordVectorBatch[i * (D * N) + j];
      if (k == model->h1 * D) {
        k = 0;
        ++l;
      }
    }
  }
  multiplyMatrix(model->W2, model->layer1StateBatch, model->h2, model->h1 * D,
                 N, model->layer2StateBatch);
  for (j = 0; j < N; ++j) {
    for (i = 0; i < model->h2; ++i) {
      model->layer2StateBatch[i * N + j] += model->bias2[i];
    }
  }
  sigmoid(model->layer2StateBatch, model->h2, N);
  multiplyMatrix(model->W3, model->layer2StateBatch, model->vocabSize,
                 model->h2, N, model->layer3StateBatch);
  for (j = 0; j < N; ++j) {
    for (i = 0; i < model->vocabSize; ++i) {
      model->layer3StateBatch[i * N + j] += model->bias3[i];
    }
  }
  softmax(model->layer3StateBatch, model->vocabSize, N);
}
```

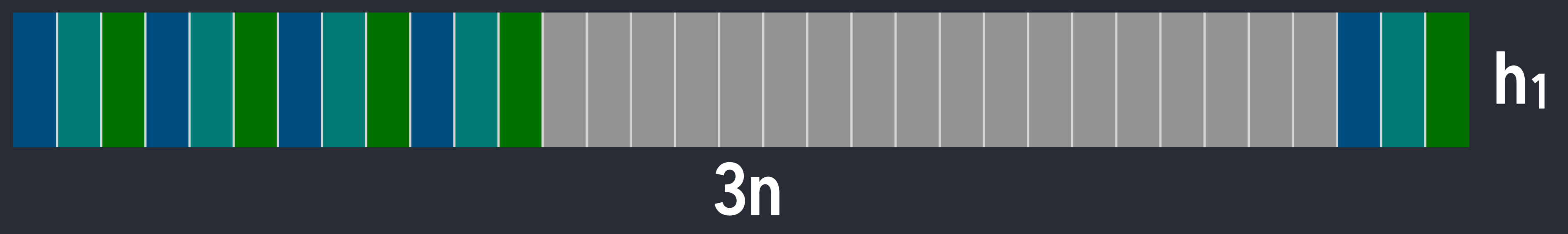
Vary from 1 to miniBatchSize N

1 Fill inputWordVectorBatch



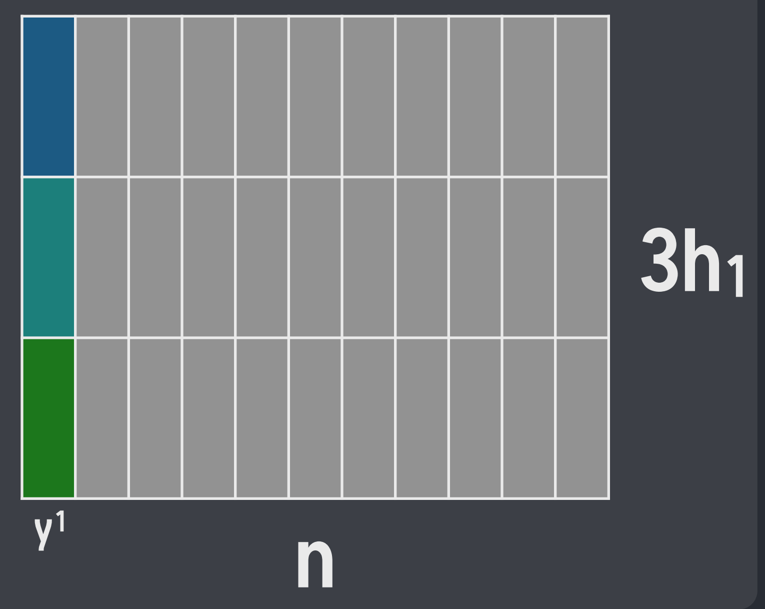
v
x_1 x_2 x_3 x_1 x_2 x_3 x_1 x_2 x_3 x_1 x_2 x_3
3n
x_1 x_2 x_3

2 Calculate bufferW1xInputWordVectorBatch = W^1 * inputWordVectorBatch



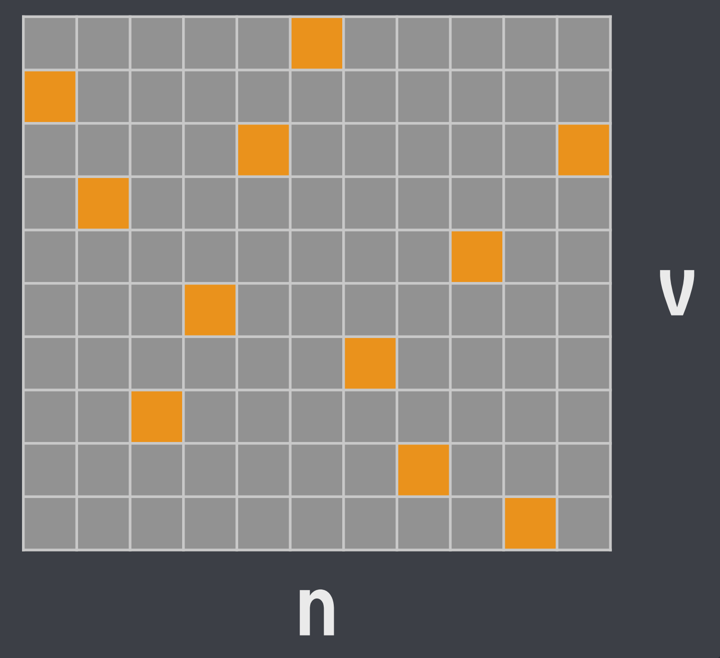
3 Manipulate layer1StateBatch, i.e. y^1 batch

4 Calculate batch $y^{2-} = W^2y^1 + b^2$



5 Calculate batch $y^{2+} = \sigma(y^{2-})$

6 Calculate batch $y^{3-} = W^3y^{2+} + b^3$



7 Calculate batch $y^{3+} = \sigma(y^{3-})$

$$\mathbf{y}^1 = \begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}$$

$$\mathbf{y}^{2-} = \mathbf{W}^2\mathbf{y}^1 + \mathbf{b}^2$$

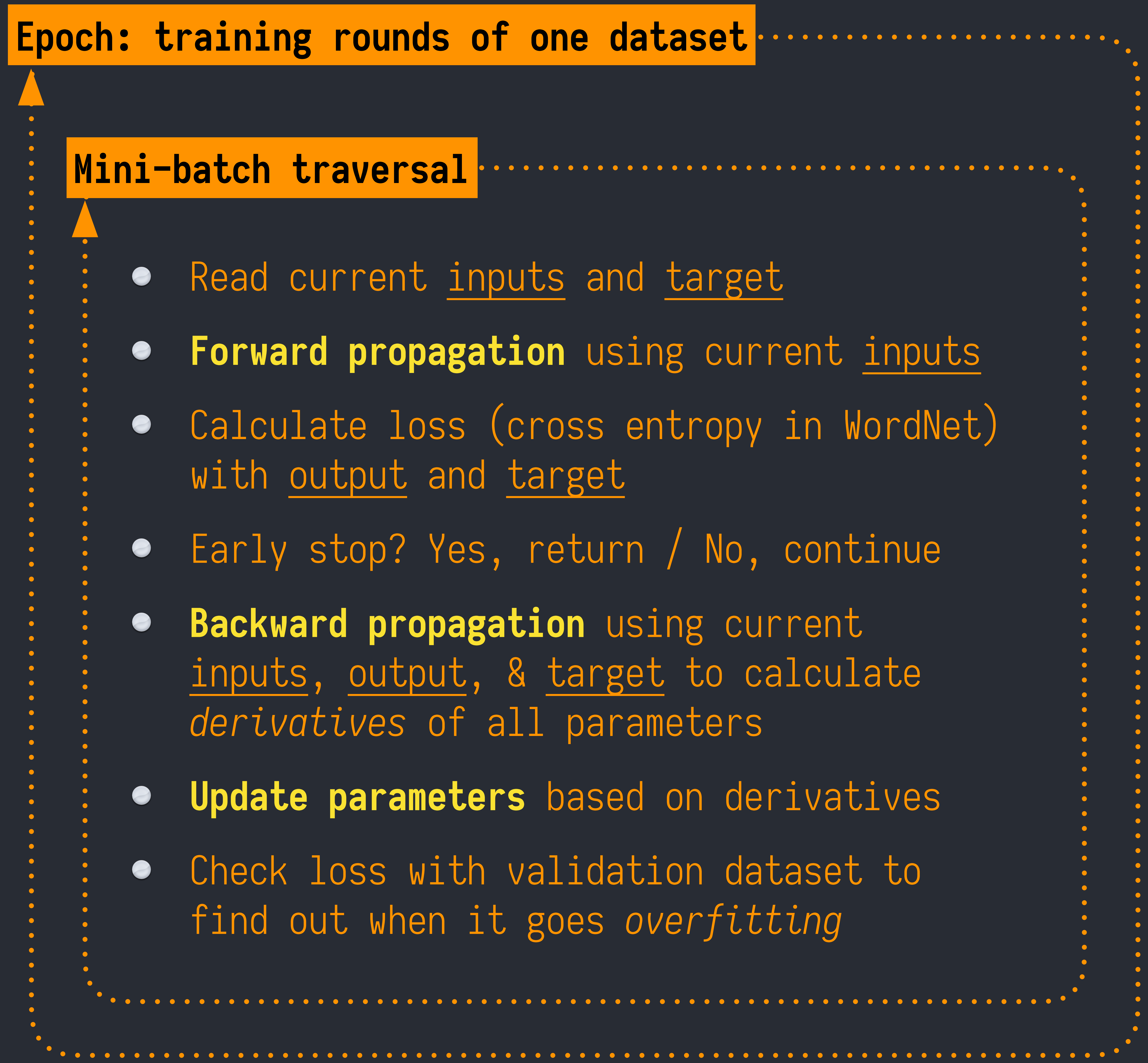
$$\mathbf{y}^{2+} = \sigma(\mathbf{y}^{2-})$$

$$\mathbf{y}^{3-} = \mathbf{W}^3\mathbf{y}^{2+} + \mathbf{b}^3$$

$$\mathbf{y}^{3+} = \mathrm{s}(\mathbf{y}^{3-})$$

Train.c: Training procedure

```c
void train(WordNet *model, const int *batchInput4Training,
           const int *batchTarget4Training, int batchNum4Training,
           const int *batchInput4Validation, const int *batchTarget4Validation,
           int batchNum4Validation) {
  ......
  ceCounter = 0;
  for (iterEpoch = 0; iterEpoch < epoch; ++iterEpoch) {
    printf("# Begin iteration Epoch = %d\n", iterEpoch);

    crossEntropy4Training = 0.0;
    for (iterBatch = 0; iterBatch < batchNum4Training; ++iterBatch, ++ceCounter) {
      currentInputBatch = &batchInput4Training[iterBatch * miniBatchSize * inputDimension];

      currentTargetBatch = &batchTarget4Training[iterBatch * miniBatchSize *
                                                 (rawDataColumn - inputDimension)];

      forwardPropagate(model, currentInputBatch, model->N);
      loadTarget2TargetVectorBatch(model, currentTargetBatch); // yt
      temp = averageCrossEntropy(model->targetVectorBatch, model->outputStateBatch,
                                 vocabSize, miniBatchSize);
      crossEntropy4Training += (temp - crossEntropy4Training) / (iterBatch + 1);
      printf("  Training CE (@%d, minibatch %d of %d, epoch %d) = %.8lf\n",
             ceCounter + 1, iterBatch + 1, batchNum4Training, iterEpoch + 1,
             crossEntropy4Training);

      if (iterEpoch == epoch - 1 && iterBatch == earlyStopIteration) {
        return;
      }

      backPropagate(model);

      updateNetworkParameters(model, momentum, learningRate);

      if (iterBatch && iterBatch % verifyPerIterBatch == 0) {
        printf("##Start validation ...\n");
        ......
        printf("##Validation CE (@%d) = %.8lf\n", ceCounter + 1,
               crossEntropy4Validation);
      }
    }
  }
}
```

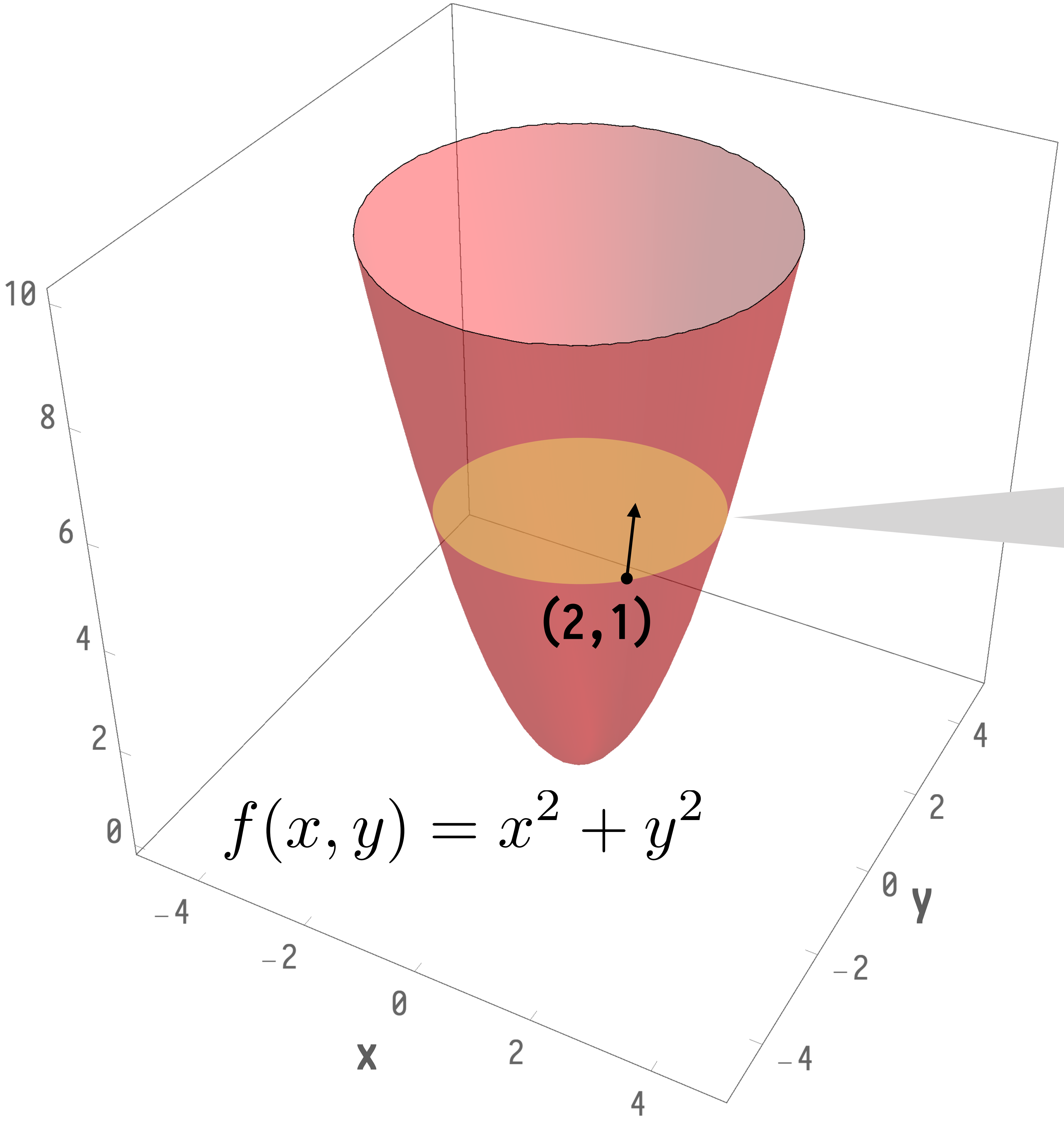


Epoch: training rounds of one dataset

Mini-batch traversal

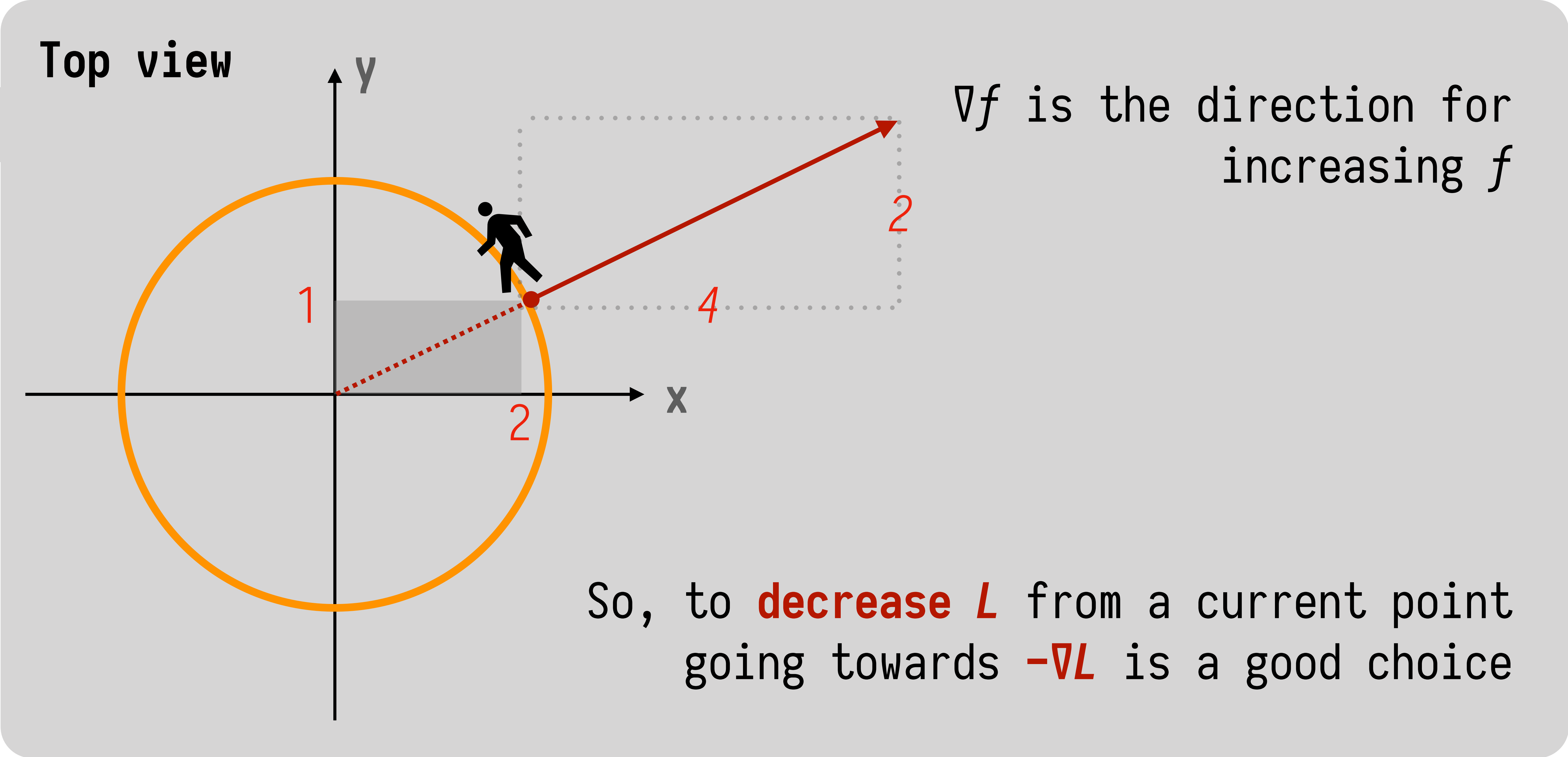
- Read current inputs and target
- **Forward propagation** using current inputs
- Calculate loss (cross entropy in WordNet) with output and target
- Early stop? Yes, return / No, continue
- **Backward propagation** using current inputs, output, & target to calculate *derivatives* of all parameters
- **Update parameters** based on derivatives
- Check loss with validation dataset to find out when it goes *overfitting*

Backprop: Derivative based optimization



$$f(x,y) = x^2 + y^2$$

- Gradient: $\nabla f = \left[\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\right]^{\mathsf{T}} = [2x\ 2y]^{\mathsf{T}}$
- Gradient at point (2,1): $\nabla f|_{(2,1)} = [4\ 2]^{\mathsf{T}}$



∇*f* is the direction for increasing *f*

So, to **decrease** ***L*** from a current point going towards **-∇*L*** is a good choice

Problem to solve: $\dfrac{\partial L}{\partial \mathbf{W}^3}, \dfrac{\partial L}{\partial \mathbf{b}^3}, \dfrac{\partial L}{\partial \mathbf{W}^2}, \dfrac{\partial L}{\partial \mathbf{b}^2}, \dfrac{\partial L}{\partial \mathbf{W}^1}$

Backprop: The chain rule

Think about a simple case

- $f(u,v) = u^2 + v$
- $u(t) = 2t,\ v(t) = t^2$
- $\left.\dfrac{\partial f}{\partial t}\right|_{t=3}$

$$f(t) = 4t^2 + t^2 = 5t^2$$

$$\frac{\partial f}{\partial t} = 10t$$

$$\left.\frac{\partial f}{\partial t}\right|_{t=3} = 30$$

Total change $\Delta f \approx \dfrac{\partial f}{\partial u}\Delta u + \dfrac{\partial f}{\partial v}\Delta v$

Change rate
in direction u

Change rate
in direction v

$$\lim_{\Delta t \to 0} \frac{\Delta f}{\Delta t} = \lim_{\Delta t \to 0} \left(\frac{\partial f}{\partial u}\frac{\Delta u}{\Delta t} + \frac{\partial f}{\partial v}\frac{\Delta v}{\Delta t} \right)$$

$$\boxed{\frac{\partial f}{\partial t} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial t}}$$

$$u(3) = 6,\ v(3) = 9$$

$$\left.\frac{\partial f}{\partial t}\right|_{t=3} = 2u(3) \cdot 2 + 1 \cdot 2t = 24 + 6 = 30$$

Backprop: Procedure

- Calculate the gradient on the output layer $\mathbf{g} \leftarrow \frac{\partial L}{\partial \mathbf{y}^{l+}}$

k = *l*, *l*-1, ..., 1

- Convert gradient to the one before activation

$$\mathbf{g} \leftarrow \frac{\partial L}{\partial \mathbf{y}^{k-}} = \mathbf{g} \odot \frac{\partial \sigma^k}{\partial \mathbf{y}^{k-}}$$

Activation function on Layer k

- Compute gradients on weights and biases

$$\frac{\partial L}{\partial \mathbf{b}^k} = \mathbf{g}$$

$$\frac{\partial L}{\partial \mathbf{W}^k} = \mathbf{g}[\mathbf{y}^{(k-1)+}]^\mathsf{T}$$

- Propagate gradient to the next lower layer

$$\mathbf{g} \leftarrow \frac{\partial L}{\partial \mathbf{y}^{(k-1)+}} = [\mathbf{W}^k]^\mathsf{T}\mathbf{g}$$

$$\mathbf{y}^1 = \begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}$$

$$\mathbf{y}^{2-} = \mathbf{W}^2\mathbf{y}^1 + \mathbf{b}^2$$

$$\mathbf{y}^{2+} = \boldsymbol{\sigma}(\mathbf{y}^{2-})$$

$$\mathbf{y}^{3-} = \mathbf{W}^3\mathbf{y}^{2+} + \mathbf{b}^3$$

$$\mathbf{y}^{3+} = \mathrm{s}(\mathbf{y}^{3-})$$

$$\min L = \sum_{i=1}^{v} -y_i^t \log y_i^{3+}$$

Backprop: Details of $\underset{v\times 1}{\frac{\partial L}{\partial \mathbf{y}^{3-}}} = \left[\frac{\partial L}{\partial y_1^{3-}} \; \frac{\partial L}{\partial y_2^{3-}} \; \cdots \; \frac{\partial L}{\partial y_v^{3-}}\right]^{\mathsf{T}}$

$$L = -y_1^t \log \overbrace{\frac{e^{y_1^{3-}}}{\sum_{j=1}^v e^{y_j^{3-}}}}^{\text{Softmax}} - y_2^t \log \frac{e^{y_2^{3-}}}{\sum_{j=1}^v e^{y_j^{3-}}} - \cdots - y_v^t \log \frac{e^{y_v^{3-}}}{\sum_{j=1}^v e^{y_j^{3-}}}$$

$$= -y_1^t(y_1^{3-} - \underbrace{\log \sum_{j=1}^{v} e^{y_j^{3-}}}_{/\partial y^{3-}_1}) - y_2^t(y_2^{3-} - \log \sum_{j=1}^{v} e^{y_j^{3-}}) - \cdots - y_v^t(y_v^{3-} - \log \sum_{j=1}^{v} e^{y_j^{3-}})$$

$$\frac{\partial L}{\partial y_1^{3-}} = -y_1^t + y_1^t \frac{e^{y_1^{3-}}}{\sum_{j=1}^v e^{y_j^{3-}}} + y_2^t \frac{e^{y_1^{3-}}}{\sum_{j=1}^v e^{y_j^{3-}}} + \cdots + y_v^t \frac{e^{y_1^{3-}}}{\sum_{j=1}^v e^{y_j^{3-}}}$$

$$= -y_1^t + y_1^t y_1^{3+} + y_2^t y_1^{3+} + \cdots + y_v^t y_1^{3+}$$

$$= -y_1^t + \underbrace{(y_1^t + y_2^t + \cdots y_v^t)}_{\text{One-hot}} y_1^{3+}$$

$$= y_1^{3+} - y_1^t$$

$$\boxed{\underset{v\times 1}{\frac{\partial L}{\partial \mathbf{y}^{3-}}} = \underset{v\times 1}{\mathbf{y}^{3+}} - \underset{v\times 1}{\mathbf{y}^{t}}}$$

$$\mathbf{y}^1 = \begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}$$

$$\mathbf{y}^{2-} = \mathbf{W}^2\mathbf{y}^1 + \mathbf{b}^2$$

$$\mathbf{y}^{2+} = \boldsymbol{\sigma}(\mathbf{y}^{2-})$$

$$\underset{v\times 1}{\mathbf{y}^{3-}} = \underset{v\times h_2}{\mathbf{W}^3}\underset{h_2\times 1}{\mathbf{y}^{2+}} + \underset{v\times 1}{\mathbf{b}^3}$$

$$\underset{v\times 1}{\mathbf{y}^{3+}} = \mathbf{s}(\underset{v\times 1}{\mathbf{y}^{3-}})$$

$$\min L = \sum_{i=1}^{v} -y_i^t \log y_i^{3+}$$

Backprop: Details of $\underset{v \times 1}{\frac{\partial L}{\partial \mathbf{b}^3}} = \left[\frac{\partial L}{\partial b_1^3} \; \frac{\partial L}{\partial b_2^3} \; \cdots \; \frac{\partial L}{\partial b_v^3} \right]^{\mathsf{T}}$

$$\frac{\partial L}{\partial b_1^3} = \frac{\partial L}{\partial y_1^{3-}} \underset{1}{\frac{\partial y_1^{3-}}{\partial b_1^3}} + \frac{\partial L}{\partial y_2^{3-}} \underset{0}{\frac{\partial y_2^{3-}}{\partial b_1^3}} + \cdots + \frac{\partial L}{\partial y_v^{3-}} \underset{0}{\frac{\partial y_v^{3-}}{\partial b_1^3}}$$

$$y_1^{3-} = (\mathbf{W}^3_{1,\cdot})\mathbf{y}^{2+} + b_1^3$$

$$\underset{v \times 1}{\frac{\partial L}{\partial \mathbf{b}^3}} = \begin{bmatrix} \underset{1}{\frac{\partial y_1^{3-}}{\partial b_1^3}} & \underset{0}{\frac{\partial y_2^{3-}}{\partial b_1^3}} & \cdots & \underset{0}{\frac{\partial y_v^{3-}}{\partial b_1^3}} \\ \underset{0}{\frac{\partial y_1^{3-}}{\partial b_2^3}} & \underset{1}{\frac{\partial y_2^{3-}}{\partial b_2^3}} & \cdots & \underset{0}{\frac{\partial y_v^{3-}}{\partial b_2^3}} \\ \vdots & \vdots & \ddots & \vdots \\ \underset{0}{\frac{\partial y_1^{3-}}{\partial b_v^3}} & \underset{0}{\frac{\partial y_2^{3-}}{\partial b_v^3}} & \cdots & \underset{1}{\frac{\partial y_v^{3-}}{\partial b_v^3}} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1^{3-}} \\ \frac{\partial L}{\partial y_2^{3-}} \\ \vdots \\ \frac{\partial L}{\partial y_v^{3-}} \end{bmatrix} = \underset{v \times v}{\overset{I}{\frac{\partial \mathbf{y}^{3-}}{\partial \mathbf{b}^3}}} \underset{v \times 1}{\frac{\partial L}{\partial \mathbf{y}^{3-}}}$$

$$\boxed{\underset{v \times 1}{\frac{\partial L}{\partial \mathbf{b}^3}} = \underset{v \times 1}{\frac{\partial L}{\partial \mathbf{y}^{3-}}}}$$

$$\mathbf{y}^1 = \begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}$$

$$\mathbf{y}^{2-} = \mathbf{W}^2\mathbf{y}^1 + \mathbf{b}^2$$

$$\mathbf{y}^{2+} = \boldsymbol{\sigma}(\mathbf{y}^{2-})$$

$$\underset{v \times 1}{\mathbf{y}^{3-}} = \underset{v \times h_2}{\mathbf{W}^3}\underset{h_2 \times 1}{\mathbf{y}^{2+}} + \underset{v \times 1}{\mathbf{b}^3}$$

$$\underset{v \times 1}{\mathbf{y}^{3+}} = \mathbf{s}(\underset{v \times 1}{\mathbf{y}^{3-}})$$

$$\min L = \sum_{i=1}^{v} -y_i^t \log y_i^{3+}$$

Backprop: Details of $\underset{\text{v × h}_2}{\dfrac{\partial L}{\partial \mathbf{W}^3}}$

$$\frac{\partial L}{\partial W^3_{11}} = \frac{\partial L}{\partial y_1^{3-}}\frac{\partial y_1^{3-}}{\partial W^3_{11}} + \frac{\partial L}{\partial y_2^{3-}}{\frac{\partial y_2^{3-}}{\partial W^3_{11}}}_{0} + \cdots + \frac{\partial L}{\partial y_v^{3-}}{\frac{\partial y_v^{3-}}{\partial W^3_{11}}}_{0}$$

$$y_2^{3-} = W^3_{21}y_1^{2+} + W^3_{22}y_2^{2+} + \cdots + W^3_{2,h_2}y_{h_2}^{2+} + b_2^3$$

$$y_1^{3-} = W^3_{11}y_1^{2+} + W^3_{12}y_2^{2+} + \cdots + W^3_{1,h_2}y_{h_2}^{2+} + b_1^3$$

$$\frac{\partial L}{\partial \mathbf{W}^3} = \begin{bmatrix} \frac{\partial L}{\partial y_1^{3-}}y_1^{2+} & \frac{\partial L}{\partial y_1^{3-}}y_2^{2+} & \cdots & \frac{\partial L}{\partial y_1^{3-}}y_{h_2}^{2+} \\ \frac{\partial L}{\partial y_2^{3-}}y_1^{2+} & \frac{\partial L}{\partial y_2^{3-}}y_2^{2+} & \cdots & \frac{\partial L}{\partial y_2^{3-}}y_{h_2}^{2+} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial y_v^{3-}}y_1^{2+} & \frac{\partial L}{\partial y_v^{3-}}y_2^{2+} & \cdots & \frac{\partial L}{\partial y_v^{3-}}y_{h_2}^{2+} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial y_1^{3-}} \\ \frac{\partial L}{\partial y_2^{3-}} \\ \vdots \\ \frac{\partial L}{\partial y_v^{3-}} \end{bmatrix} \begin{bmatrix} y_1^{2+} & y_2^{2+} & \cdots & y_{h_2}^{2+} \end{bmatrix}$$

$$\rightarrow \boxed{\underset{\text{v × h}_2}{\frac{\partial L}{\partial \mathbf{W}^3}} = \underset{\text{v × 1}}{\frac{\partial L}{\partial \mathbf{y}^{3-}}}\underset{\text{1 × h}_2}{(\mathbf{y}^{2+})^{\mathsf{T}}}}$$

$$\mathbf{y}^1 = \begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}$$

$$\mathbf{y}^{2-} = \mathbf{W}^2\mathbf{y}^1 + \mathbf{b}^2$$

$$\mathbf{y}^{2+} = \boldsymbol{\sigma}(\mathbf{y}^{2-})$$

$$\underset{\text{v × 1}}{\mathbf{y}^{3-}} = \underset{\text{v × h}_2}{\mathbf{W}^3}\underset{\text{h}_2\text{ × 1}}{\mathbf{y}^{2+}} + \underset{\text{v × 1}}{\mathbf{b}^3}$$

$$\underset{\text{v × 1}}{\mathbf{y}^{3+}} = \mathbf{s}(\underset{\text{v × 1}}{\mathbf{y}^{3-}})$$

$$\min L = \sum_{i=1}^{v} -y_i^t \log y_i^{3+}$$

Backprop: Details of $\underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2+}}}$

$$
\begin{aligned}
y_1^{3-} &= W_{11}^3 y_1^{2+} + W_{12}^3 y_2^{2+} + \cdots + W_{1,h_2}^3 y_{h_2}^{2+} + b_1^3 \\
y_2^{3-} &= W_{21}^3 y_1^{2+} + W_{22}^3 y_2^{2+} + \cdots + W_{2,h_2}^3 y_{h_2}^{2+} + b_2^3 \\
y_v^{3-} &= W_{v1}^3 y_1^{2+} + W_{v2}^3 y_2^{2+} + \cdots + W_{v,h_2}^3 y_{h_2}^{2+} + b_v^3
\end{aligned}
$$

$$
\frac{\partial L}{\partial y_1^{2+}} = \frac{\partial L}{\partial y_1^{3-}} \frac{\partial y_1^{3-}}{\partial y_1^{2+}} + \frac{\partial L}{\partial y_2^{3-}} \frac{\partial y_2^{3-}}{\partial y_1^{2+}} + \cdots + \frac{\partial L}{\partial y_v^{3-}} \frac{\partial y_v^{3-}}{\partial y_1^{2+}} = W_{11}^3 \frac{\partial L}{\partial y_1^{3-}} + W_{21}^3 \frac{\partial L}{\partial y_2^{3-}} + \cdots + W_{v1}^3 \frac{\partial L}{\partial y_v^{3-}}
$$

$$
\frac{\partial L}{\partial \mathbf{y}^{2+}} = \begin{bmatrix} W_{11}^3 & W_{21}^3 & \cdots & W_{v1}^3 \\ W_{12}^3 & W_{22}^3 & \cdots & W_{v2}^3 \\ \vdots & \vdots & \ddots & \vdots \\ W_{1,h_2}^3 & W_{2,h^2}^3 & \cdots & W_{v,h_2}^3 \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1^{3-}} \\ \frac{\partial L}{\partial y_2^{3-}} \\ \vdots \\ \frac{\partial L}{\partial y_v^{3-}} \end{bmatrix} = (\mathbf{W}^3)^\top \frac{\partial L}{\partial \mathbf{y}^{3-}}
$$

➡

$$
\boxed{\underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2+}}} = \underset{h_2 \times v}{(\mathbf{W}^3)^\top} \underset{v \times 1}{\frac{\partial L}{\partial \mathbf{y}^{3-}}}}
$$

$$
\begin{aligned}
\mathbf{y}^1 &= \begin{bmatrix} \mathbf{W}^1 \mathbf{x}_1 \\ \mathbf{W}^1 \mathbf{x}_2 \\ \mathbf{W}^1 \mathbf{x}_3 \end{bmatrix} \\
\mathbf{y}^{2-} &= \mathbf{W}^2 \mathbf{y}^1 + \mathbf{b}^2 \\
\mathbf{y}^{2+} &= \boldsymbol{\sigma}(\mathbf{y}^{2-}) \\
\underset{v \times 1}{\mathbf{y}^{3-}} &= \underset{v \times h_2}{\mathbf{W}^3} \underset{h_2 \times 1}{\mathbf{y}^{2+}} + \underset{v \times 1}{\mathbf{b}^3} \\
\underset{v \times 1}{\mathbf{y}^{3+}} &= \mathbf{s}(\underset{v \times 1}{\mathbf{y}^{3-}}) \\
\min L &= \sum_{i=1}^{v} -y_i^t \log y_i^{3+}
\end{aligned}
$$

Backprop: Details of $\underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2-}}}$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d\sigma}{dx} = \sigma(1-\sigma)$$

$$\Downarrow$$

$$\frac{\partial y_i^{2+}}{\partial y_i^{2-}} = y_i^{2+}(1-y_i^{2+})$$

$$\underset{(D \times h_1) \times 1}{\mathbf{y}^1} = \overset{h_1 \times v}{\begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}} \begin{matrix} v \times 1 \\ v \times 1 \\ v \times 1 \end{matrix}$$

$$\underset{h_2 \times 1}{\mathbf{y}^{2-}} = \underset{h_2 \times (D \times h_1)}{\mathbf{W}^2}\underset{(D \times h_1) \times 1}{\mathbf{y}^1} + \underset{h_2 \times 1}{\mathbf{b}^2}$$

$$\underset{h_2 \times 1}{\mathbf{y}^{2+}} = \boldsymbol{\sigma}(\underset{h_2 \times 1}{\mathbf{y}^{2-}})$$

$$\frac{\partial L}{\partial y_1^{2-}} = \frac{\partial L}{\partial y_1^{2+}} \underbrace{\frac{\partial y_1^{2+}}{\partial y_1^{2-}}}_{y_1^{2+}(1-y_1^{2+})} + \frac{\partial L}{\partial y_2^{2+}} \underbrace{\frac{\partial y_2^{2+}}{\partial y_1^{2-}}}_{0} + \cdots + \frac{\partial L}{\partial y_{h_2}^{2+}} \underbrace{\frac{\partial y_{h_2}^{2+}}{\partial y_1^{2-}}}_{0}$$

$$\frac{\partial L}{\partial \mathbf{y}^{2-}} = \frac{\partial \mathbf{y}^{2+}}{\partial \mathbf{y}^{2-}} \frac{\partial L}{\partial \mathbf{y}^{2+}}$$

$$= \underset{h_2 \times h_2}{\begin{bmatrix} \frac{\partial y_1^{2+}}{\partial y_1^{2-}} & & & \\ & \frac{\partial y_2^{2+}}{\partial y_2^{2-}} & & \\ & & \ddots & \\ & & & \frac{\partial y_{h_2}^{2+}}{\partial y_{h_2}^{2-}} \end{bmatrix}} \begin{bmatrix} \frac{\partial L}{\partial y_1^{2+}} \\ \frac{\partial L}{\partial y_2^{2+}} \\ \vdots \\ \frac{\partial L}{\partial y_{h_2}^{2+}} \end{bmatrix} = \begin{bmatrix} y_1^{2+}(1-y_1^{2+}) & & & \\ & y_2^{2+}(1-y_2^{2+}) & & \\ & & \ddots & \\ & & & y_{h_2}^{2+}(1-y_{h_2}^{2+}) \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1^{2+}} \\ \frac{\partial L}{\partial y_2^{2+}} \\ \vdots \\ \frac{\partial L}{\partial y_{h_2}^{2+}} \end{bmatrix}$$

$$\Rightarrow \boxed{\underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2-}}} = \underset{h_2 \times 1}{\mathbf{y}^{2+}} \circ \underset{h_2 \times 1}{(\mathbf{1} - \mathbf{y}^{2+})} \circ \underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2+}}}}$$

$$\boxed{\underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{b}^2}} = \underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2-}}}, \quad \underset{h_2 \times (D \times h_1)}{\frac{\partial L}{\partial \mathbf{W}^2}} = \underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2-}}} \underset{1 \times (D \times h_1)}{(\mathbf{y}^1)^\mathsf{T}}, \quad \underset{(D \times h_1) \times 1}{\frac{\partial L}{\partial \mathbf{y}^1}} = \underset{(D \times h_1) \times h_2}{(\mathbf{W}^2)^\mathsf{T}} \underset{h_2 \times 1}{\frac{\partial L}{\partial \mathbf{y}^{2-}}}}$$

Corollary

Backprop: Details of $\underset{h_1 \times v}{\frac{\partial L}{\partial \mathbf{W}^1}}$

$$\underset{(D \times h_1) \times 1}{\mathbf{y}^1} = \begin{array}{cc} \underset{h_1 \times v}{\begin{bmatrix} \mathbf{W}^1\mathbf{x}_1 \\ \mathbf{W}^1\mathbf{x}_2 \\ \mathbf{W}^1\mathbf{x}_3 \end{bmatrix}} & \begin{array}{c} v \times 1 \\ v \times 1 \\ v \times 1 \end{array} \end{array}$$

$$\mathbf{y}^1 = \begin{bmatrix} y_1^1 \\ y_2^1 \\ \vdots \\ y_{h_1}^1 \\ \hline y_{h_1+1}^1 \\ \vdots \\ y_{2h_1}^1 \\ \hline y_{2h_1+1}^1 \\ \vdots \\ y_{3h_1}^1 \end{bmatrix} = \begin{bmatrix} W_{11}^1 x_1^1 + W_{12}^1 x_2^1 + \cdots + W_{1v}^1 x_v^1 \\ W_{21}^1 x_1^1 + W_{22}^1 x_2^1 + \cdots + W_{2v}^1 x_v^1 \\ \vdots \\ W_{h_1 1}^1 x_1^1 + W_{h_1 2}^1 x_2^1 + \cdots + W_{h_1 v}^1 x_v^1 \\ \hline W_{11}^1 x_1^2 + W_{12}^1 x_2^2 + \cdots + W_{1v}^1 x_v^2 \\ \vdots \\ W_{h_1 1}^1 x_1^2 + W_{h_1 2}^1 x_2^2 + \cdots + W_{h_1 v}^1 x_v^2 \\ \hline W_{11}^1 x_1^3 + W_{12}^1 x_2^3 + \cdots + W_{1v}^1 x_v^3 \\ \vdots \\ W_{h_1 1}^1 x_1^3 + W_{h_1 2}^1 x_2^3 + \cdots + W_{h_1 v}^1 x_v^3 \end{bmatrix}$$

$$\begin{aligned} \frac{\partial L}{\partial W_{11}^1} = {} & \frac{\partial L}{\partial y_1^1}\frac{\partial y_1^1}{\partial W_{11}^1} + \frac{\partial L}{\partial y_2^1}\underbrace{\frac{\partial y_2^1}{\partial W_{11}^1}}_{0} + \cdots + \frac{\partial L}{\partial y_{h_1}^1}\underbrace{\frac{\partial y_{h_1}^1}{\partial W_{11}^1}}_{0} + \\ & \frac{\partial L}{\partial y_{h_1+1}^1}\frac{\partial y_{h_1+1}^1}{\partial W_{11}^1} + \frac{\partial L}{\partial y_{h_1+2}^1}\underbrace{\frac{\partial y_{h_1+2}^1}{\partial W_{11}^1}}_{0} + \cdots + \frac{\partial L}{\partial y_{2h_1}^1}\underbrace{\frac{\partial y_{2h_1}^1}{\partial W_{11}^1}}_{0} + \\ & \frac{\partial L}{\partial y_{2h_1+1}^1}\frac{\partial y_{2h_1+1}^1}{\partial W_{11}^1} + \frac{\partial L}{\partial y_{2h_1+2}^1}\underbrace{\frac{\partial y_{2h_1+2}^1}{\partial W_{11}^1}}_{0} + \cdots + \frac{\partial L}{\partial y_{3h_1}^1}\underbrace{\frac{\partial y_{3h_1}^1}{\partial W_{11}^1}}_{0} \\ = {} & \frac{\partial L}{\partial y_1^1}x_1^1 + \frac{\partial L}{\partial y_{h_1+1}^1}x_1^2 + \frac{\partial L}{\partial y_{2h_1+1}^1}x_1^3 \end{aligned}$$

$$\frac{\partial L}{\partial W_{12}^1} = \frac{\partial L}{\partial y_1^1}x_2^1 + \frac{\partial L}{\partial y_{h_1+1}^1}x_2^2 + \frac{\partial L}{\partial y_{2h_1+1}^1}x_2^3$$

$$\frac{\partial L}{\partial W_{ij}^1} = \frac{\partial L}{\partial y_i^1}x_j^1 + \frac{\partial L}{\partial y_{h_1+i}^1}x_j^2 + \frac{\partial L}{\partial y_{2h_1+i}^1}x_j^3$$

$$\rightarrow \boxed{\underset{h_1 \times v}{\frac{\partial L}{\partial \mathbf{W}^1}} = \sum_{i=1}^{D} \underset{h_1 \times 1}{\left(\frac{\partial L}{\partial \mathbf{y}^1}[i]\right)} \underset{1 \times v}{\mathbf{x}_i^{\mathsf{T}}}}$$

Mini-batch implementation

Model.c: backprop

```c
void backPropagate(WordNet *model) {
  int i, j, k, l;
```

```c
  subtractMatrix(model->outputStateBatch, model->yt, model->vocabSize, model->N,
                 model->dLdy3_);
  transposeMatrix(model->dLdy3_, model->vocabSize, model->N, model->dLdy3_t);
```

$$\frac{\partial L}{\partial \mathbf{y}^{3-}}$$

```c
  transposeMatrix(model->layer2StateBatch, model->h2, model->N, model->y2t);
  multiplyMatrix(model->dLdy3_, model->y2t, model->vocabSize, model->N,
                 model->h2, model->dLdW3);
```

$$\frac{\partial L}{\partial \mathbf{W}^{3}}$$

```c
  sumColumn2Vector(model->dLdy3_, model->vocabSize, model->N, model->dLdb3);
```

$$\frac{\partial L}{\partial \mathbf{b}^{3}}$$

```c
  transposeMatrix(model->W3, model->vocabSize, model->h2, model->W3t);
  multiplyMatrix(model->W3t, model->dLdy3_, model->h2, model->vocabSize,
                 model->N, model->dLdy2);
```

$$\frac{\partial L}{\partial \mathbf{y}^{2+}}$$

```c
  j = model->h2 * model->N;
  for (i = 0; i < j; ++i) {
    model->dLdy2_[i] = model->dLdy2[i] * model->layer2StateBatch[i] *
                       (1.0 - model->layer2StateBatch[i]);
  }
```

$$\frac{\partial L}{\partial \mathbf{y}^{2-}}$$

```c
  sumColumn2Vector(model->dLdy2_, model->h2, model->N, model->dLdb2);
```

$$\frac{\partial L}{\partial \mathbf{b}^{2}}$$

```c
  transposeMatrix(model->layer1StateBatch, model->h1 * model->D, model->N,
                  model->y1t);
  multiplyMatrix(model->dLdy2_, model->y1t, model->h2, model->N,
                 model->h1 * model->D, model->dLdW2);
```

$$\frac{\partial L}{\partial \mathbf{W}^{2}}$$

```c
  transposeMatrix(model->W2, model->h2, model->h1 * model->D, model->W2t);
  multiplyMatrix(model->W2t, model->dLdy2_, model->h1 * model->D, model->h2,
                 model->N, model->dLdy1);
```

$$\frac{\partial L}{\partial \mathbf{y}^{1}}$$

```c
  memset(model->dLdW1, 0, model->h1 * model->vocabSize * sizeof(double));
  for (k = 0; k < model->D; ++k) {
    for (i = 0; i < model->vocabSize; ++i) {
      for (j = k, l = 0; j < model->D * model->N; j += model->D, ++l) {
        model->xit[l * model->vocabSize + i] =
          model->inputWordVectorBatch[i * model->D * model->N + j];
      }
    }
    for (i = k * model->h1, l = 0; l < model->h1; ++i, ++l) {
      for (j = 0; j < model->N; ++j) {
        model->dLdy1i[l * model->N + j] = model->dLdy1[i * model->N + j];
      }
    }
    multiplyAddMatrix(model->dLdy1i, model->xit, model->h1, model->N,
                      model->vocabSize, model->dLdW1);
  }
}
```

$$\frac{\partial L}{\partial \mathbf{W}^{1}}$$

$$\frac{\partial L}{\partial \mathbf{y}^{3-}} = \mathbf{y}^{3+} - \mathbf{y}^{t}$$

$$\frac{\partial L}{\partial \mathbf{b}^{3}} = \frac{\partial L}{\partial \mathbf{y}^{3-}}$$

$$\frac{\partial L}{\partial \mathbf{W}^{3}} = \frac{\partial L}{\partial \mathbf{y}^{3-}} (\mathbf{y}^{2+})^{\mathsf{T}}$$

$$\frac{\partial L}{\partial \mathbf{y}^{2+}} = (\mathbf{W}^{3})^{\mathsf{T}} \frac{\partial L}{\partial \mathbf{y}^{3-}}$$

$$\frac{\partial L}{\partial \mathbf{y}^{2-}} = \mathbf{y}^{2+} \circ (\mathbf{1} - \mathbf{y}^{2+}) \circ \frac{\partial L}{\partial \mathbf{y}^{2+}}$$

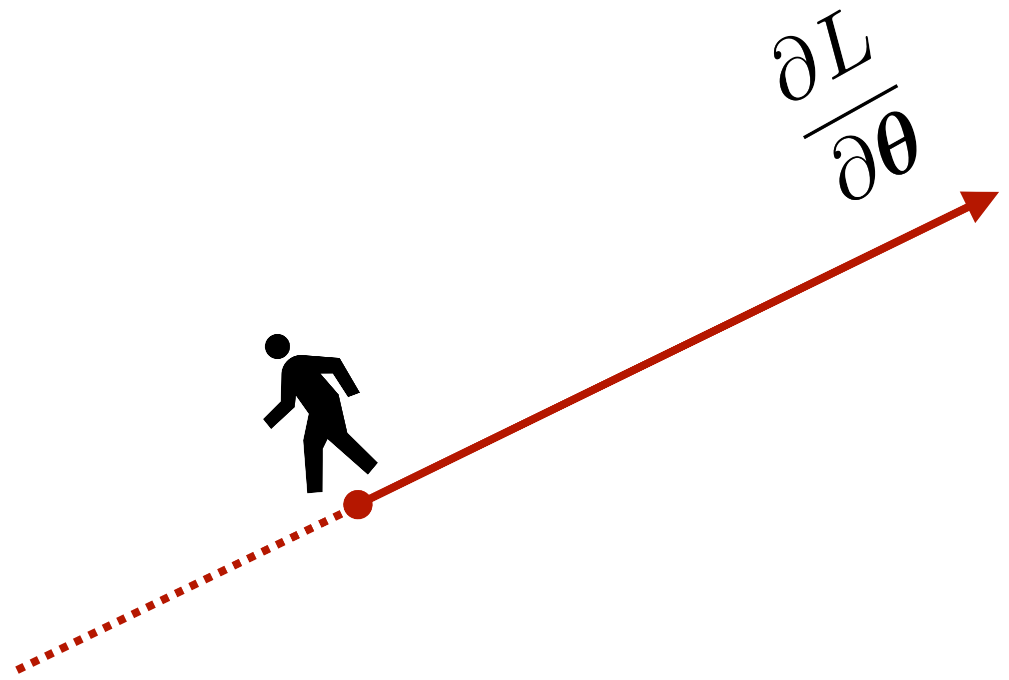
$$\frac{\partial L}{\partial \mathbf{b}^{2}} = \frac{\partial L}{\partial \mathbf{y}^{2-}}$$

$$\frac{\partial L}{\partial \mathbf{W}^{2}} = \frac{\partial L}{\partial \mathbf{y}^{2-}} (\mathbf{y}^{1})^{\mathsf{T}}$$

$$\frac{\partial L}{\partial \mathbf{y}^{1}} = (\mathbf{W}^{2})^{\mathsf{T}} \frac{\partial L}{\partial \mathbf{y}^{2-}}$$

$$\frac{\partial L}{\partial \mathbf{W}^{1}} = \sum_{i=1}^{D} \left(\frac{\partial L}{\partial \mathbf{y}^{1}}[i] \right) \mathbf{x}_i^{\mathsf{T}}$$

Update parameters



How far should I go?

- Basic update rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}}$$

Learning rate

- Constant?
- AdaGrad?
- RMSProp?
- Adam?

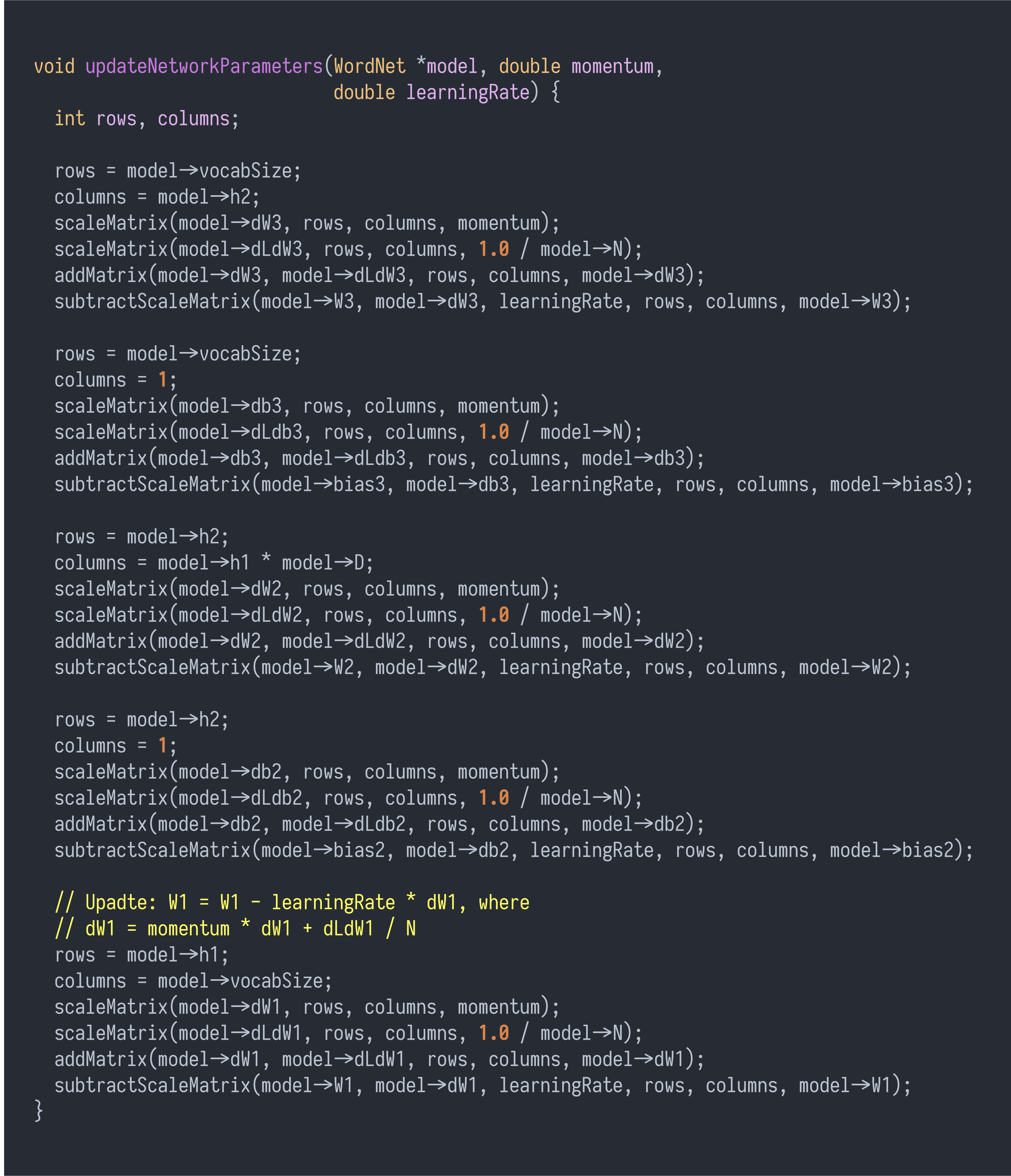
- With momentum

[0,1) Contribution of previous gradient?

$$\mathbf{v} \leftarrow \beta \cdot \mathbf{v} - \alpha \cdot \frac{\partial L}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}$$

- Nesterov?



```c
void updateNetworkParameters(WordNet *model, double momentum,
                             double learningRate) {
  int rows, columns;

  rows = model->vocabSize;
  columns = model->h2;
  scaleMatrix(model->dW3, rows, columns, momentum);
  scaleMatrix(model->dLdW3, rows, columns, 1.0 / model->N);
  addMatrix(model->dW3, model->dLdW3, rows, columns, model->dW3);
  subtractScaleMatrix(model->W3, model->dW3, learningRate, rows, columns, model->W3);

  rows = model->vocabSize;
  columns = 1;
  scaleMatrix(model->db3, rows, columns, momentum);
  scaleMatrix(model->dLdb3, rows, columns, 1.0 / model->N);
  addMatrix(model->db3, model->dLdb3, rows, columns, model->db3);
  subtractScaleMatrix(model->bias3, model->db3, learningRate, rows, columns, model->bias3);

  rows = model->h2;
  columns = model->h1 * model->D;
  scaleMatrix(model->dW2, rows, columns, momentum);
  scaleMatrix(model->dLdW2, rows, columns, 1.0 / model->N);
  addMatrix(model->dW2, model->dLdW2, rows, columns, model->dW2);
  subtractScaleMatrix(model->W2, model->dW2, learningRate, rows, columns, model->W2);

  rows = model->h2;
  columns = 1;
  scaleMatrix(model->db2, rows, columns, momentum);
  scaleMatrix(model->dLdb2, rows, columns, 1.0 / model->N);
  addMatrix(model->db2, model->dLdb2, rows, columns, model->db2);
  subtractScaleMatrix(model->bias2, model->db2, learningRate, rows, columns, model->bias2);

  // Upadte: W1 = W1 - learningRate * dW1, where
  // dW1 = momentum * dW1 + dLdW1 / N
  rows = model->h1;
  columns = model->vocabSize;
  scaleMatrix(model->dW1, rows, columns, momentum);
  scaleMatrix(model->dLdW1, rows, columns, 1.0 / model->N);
  addMatrix(model->dW1, model->dLdW1, rows, columns, model->dW1);
  subtractScaleMatrix(model->W1, model->dW1, learningRate, rows, columns, model->W1);
}
```