Lecture 5: Mapping Models to Brain and Behavioral Data
2018.10.08
NB: This lecture should be read in conjunction with the tutorial at:

Linking Hypotheses

Neural Recordings from IT and V4

V1
IT
V2
V4
...
LN
LN...
LN
LN...
LN
LN
LN...
LN...
LN
LN
...
...

Neural Recordings from IT and V4
Linking Hypotheses

Three levels of comparison:

1. Behavioral: does the model (or brain feature set) explain the patterns of behavior?

2. Population similarity: is the population code in the model the same as in the neural data?

3. Single-neuron predictivity: can each neuron in the brain be matched by a synthetic neuron from the model? and vice versa?
Linking Hypotheses

Three levels of comparison:

1. Behavioral: does the model (or brain feature set) explain the patterns of behavior?

2. Population similarity: is the population code in the model the same as in the neural data?

3. Single-neuron predictivity: can each neuron in the brain be matched by a synthetic neuron from the model? and vice versa?
Models are linking hypotheses:

Pixel representation  encoding  IT neural representation  decoding  Behavioral patterns

“animal”
“boat”
“car”
Models are linking hypotheses:

*IT neural representation*

"animal"

"boat"

"car"

Behavioral Linking Hypothesis
Models are linking hypotheses:

**IT neural representation**

**Behavioral patterns**

- “animal”
- “boat”
- “car”

> decoding
Behavioral Linking Hypothesis

Models are linking hypotheses:

IT neural representation → Behavioral patterns

decoding

“animal”
“boat”
“car”
...
...
...

sensory system → motor system
Behavioral Linking Hypothesis

Models are linking hypotheses:

\[ y_k = \phi \left( \sum_{j=0}^{m} w_{k,j} x_j + b_j \right) \]

sensory system \( \rightarrow \) IT neural representation

decoding

Behavioral patterns

“animal”
“boat”
“car”

\( \rightarrow \) motor system

McCulloch-Pitts artificial neuron
Behavioral Linking Hypothesis

Models are linking hypotheses:

\[ y_k = \phi \left( \sum_{j=0}^{m} w_{kj} x_j + b_j \right) \]

\[ \text{pred}(s_j) = \text{sign} \left( \sum_i w_i \cdot \text{neurons}_i(s_j) + b \right) \]
Behavioral Linking Hypothesis

Models are linking hypotheses:

\[ y_k = \phi \left( \sum_{j=0}^{m} w_{kj} x_j + b_j \right) \]

pred\( (s_j) = \text{sign} \left( \sum_i w_i \cdot \text{neurons}_i(s_j) + b \right) \)

\[ \phi = \text{sign fn.} \]
Behavioral Linking Hypothesis

Models are linking hypotheses:

\[ y_k = \phi \left( \sum_{j=0}^{m} w_{k,j} x_j + b_j \right) \]

\[ \text{pred}(s_j) = \text{sign} \left( \sum_i w_i \cdot \text{neurons}_i(s_j) + b \right) \]

McCulloch-Pitts artificial neuron

What would it mean for an idea like this be really true?

At the very least, the **patterns of errors** of behavior would have to be correctly predicted by the decoding model.
At the very least, the **patterns of errors** of behavior would have to be correctly predicted by the decoding model.

We’ll reproduce a key result from:

**Simple Learned Weighted Sums of Inferior Temporal Neuronal Firing Rates Accurately Predict Human Core Object Recognition Performance**

Najib J. Majaj, Ha Hong, Ethan A. Solomon, and James J. DiCarlo

*Journal of Neuroscience* 30 September 2015, 35 (39) 13402-13418; DOI: [https://doi.org/10.1523/JNEUROSCI.5181-14.2015](https://doi.org/10.1523/JNEUROSCI.5181-14.2015)
Behavioral Linking Hypothesis

Procedure for comparison

1. Measure Human Behavior on Battery of Categorization Tasks

2. Compute a not-too-noisy metric of success for each task

3. Train classifiers on the same tasks and compute the same metrics

4. Compare the patterns of success across tasks.
Human Behavioral Data

[IPYNB: Human Behavioral Dataset]
Human Behavioral Data

Low-Variation
Basic Categorization

Medium-Variation
Animal Identification
1. Measure Human Behavior on Battery of Categorization Tasks

2. Compute a not-too-noisy metric of success for each task

3. Train classifiers on the same tasks and compute the same metrics

4. Compare the patterns of success across tasks.
1. Measure Human Behavior on Battery of Categorization Tasks

2. Compute a *not-too-noisy metric of success* for each task

3. Train classifiers on the same tasks and compute the same metrics

4. Compare the patterns of success across tasks.

Ok, but what metric specifically?
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one
What metric specifically?

1. Literally look at each image one by one

![Diagram: Image-by-category confusion matrix](image.png)
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one

image-by-category confusion matrix from behavior

image-by-category confusion matrix from neural decode
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one

image-by-category confusion matrix

from behavior

distractors

how similar?

distractors

from neural decode

images
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one

Too noisy because almost all confusion matrix elements are empty

image-by-category confusion matrix

from behavior

distractors

how similar?

distractors

from neural decode
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one
What metric specifically?

1. Literally look at each image one by one

2. Category-level confusion matrices
Linking Hypotheses

2. Category-level confusion matrices

- from behavior
- from neural decode
2. Category-level confusion matrices

How similar?

From behavior

From neural decode
2. Category-level confusion matrices

from behavior

from neural decode

Linking Hypotheses

how similar?

Still too noisy.
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one

2. Category-level confusion matrices

2. Metric of per-option accuracy

→ compute eight dprimes summarizing accuracy and false alarm rates
Linking Hypotheses

What metric specifically?

1. Literally look at each image one by one

2. Category-level confusion matrices

2. Metric of per-option accuracy

→ compute eight dprimes summarizing accuracy and false alarm rates

theses aren’t that badly noisy.
[IPYNB:  Split-half reliability of human data]
Linking Hypotheses

Procedure for comparison

1. Measure Human Behavior on Battery of Categorization Tasks

2. Compute a not-too-noisy metric of success for each task

3. Train classifiers on the neural data on the same tasks and compute the same metrics

4. Compare the patterns of success across tasks between humans and neural ...
Human Behavioral Data
Linking Hypotheses

Procedure for comparison

1. Measure Human Behavior on Battery of Categorization Tasks

2. Compute a not-too-noisy metric of success for each task

3. Train classifiers on the same tasks and compute the same metrics

4. Compare the patterns of success across tasks.
Procedure for comparison

1. Measure Human Behavior on Battery of Categorization Tasks

2. Compute a not-too-noisy metric of success for each task

3. Train classifiers on the same tasks and compute the same metrics

4. Compare the patterns of success across tasks.

As we’ve done before, look at the correlations

In this case, between the vectors of dprimes
Ok, so our main metric of comparison is:

$$\text{corr}(\text{human dprimes, neural dprimes})$$
Ok, so our main metric of comparison is:

\[
\text{corr(\text{human dprimes, neural dprimes})}
\]

But actually neither dprime vector is perfectly non-noisy.

How do we correct for noise?
Ok, so our main metric of comparison is:

$$\text{corr}(\text{human dprimes}, \text{neural dprimes})$$

But actually neither dprime vector is perfectly non-noisy.

How do we correct for noise?

$$\frac{\text{corr}(\text{human dprimes}, \text{neural dprimes})}{\sqrt{\text{split-half}(\text{human, human}) \cdot \text{split-half}(\text{neural, neural})}}$$
Ok, so our main metric of comparison is:

$$\text{corr}(\text{human dprimes, neural dprimes})$$

But actually neither dprime vector is perfectly non-noisy.

How do we correct for noise?

$$\text{corr}(\text{human dprimes, neural dprimes}) \cdot \sqrt{\text{split-half}(\text{human, human}) \cdot \text{split-half}(\text{neural, neural})}$$

each one is spearman-brown corrected itself
Human Behavioral Data

[IPYNB: Actually make the comparison]
Result 1:
Linking hypothesis pretty good, especially at basic categorization
Result 2: IT >> V4 at matching pattern of human behavior across all tasks vs IT vs V4

... and especially clear from high-variation identification tasks
Eg. the blue fat arrow path is much more relevant than a potential green V4 pathway.
Result 3: Potentially understand *which* specific decoder is better …

SVM vs Correlation Classifier
Result 3: Potentially understand *which* specific decoder is better . . .

\[ \text{SVM vs Correlation Classifier} \]

... can be isolated statistically:

Simple Learned Weighted Sums of Inferior Temporal Neuronal Firing Rates Accurately Predict Human Core Object Recognition Performance

Najib J. Majaj, Ha Hong, Ethan A. Solomon, and James J. DiCarlo

*Journal of Neuroscience* 30 September 2015, 35 (39) 13402-13418; DOI: https://doi.org/10.1523/JNEUROSCI.5181-14.2015
Linking Hypotheses

Result 3: Potentially understand *which* specific decoder is better . . .

SVM vs Correlation Classifier

- Human/IT-SVM
  - Corrected Correlation: 0.86
  - Corrected R-squared: 0.75

- Human/IT-Corr
  - Corrected Correlation: 0.77
  - Corrected R-squared: 0.60

sensory system → motor system

e.g. *maybe* this is implementing more like SVM learning rather than correlation classifier
Monkey Neurons vs Humans

(a) IT, V4, V1, and Pixels scatter plots showing decoder performance against human performance.

(b) Bar graph comparing behavioral consistency across IT, V4, V1, and Pixels.
Behavior = Feature space + Simple decision rule
= encoding + decoding

e.g. maybe this is implementing more like SVM learning rather than correlation classifier
Behavioral match between models and data at category confusion level is pretty good …
Behavioral match between models and data at category confusion level is pretty good …

“O1” = behavior at the object level, cross-distractor average

“O2” = behavior at the object level, on a per-distractor basis

different aggregates of the confusion matrix
Behavioral match between models and data at category confusion level is pretty good ...

“O1” = behavior at the object level, cross-distractor average

vector length num_categories

matrix of shape num_categories x num_categories

“O2” = behavior at the object level, on a per-distractor basis

different aggregates of the confusion matrix
Beamrual match between models and data at category confusion level is pretty good ...

“O1” = behavior at the object level, cross-distractor average

Somewhat less detailed, smallish number of trials to estimate well

“O2” = behavior at the object level, on a per-distractor basis

Somewhat more detailed, more trials to estimate well

different aggregates of the confusion matrix
Behavioral match between models and data at category confusion level is pretty good …
Performance Comparison

But NOT perfect!

"i1" = behavior at the image level, cross-distractor average

vector of length num_images

matrix of shape num_image x num_categories

"i2" = behavior at the image level, on a per-distractor basis
Performance Comparison

But NOT perfect!

“i1” = behavior at the image level, cross-distractor average

“i2” = behavior at the image level, on a per-distractor basis

requires even more trials to estimate — but worth it! because it separates models

f-load of trials to estimate. worth it?
Performance Comparison

But NOT perfect!
1. Behavioral: does the model (or brain feature set) explain the patterns of behavior?

pros:
   i. easy to measure
   ii. pretty discriminative if done right

cons:
   i. too task-specific?
   ii. not high resolution enough?
Linking Hypotheses

Three levels of comparison:

1. Behavioral: does the model (or brain feature set) explain the patterns of behavior?

2. Population similarity: is the population code in the model the same as in the neural data?

3. Single-neuron predictivity: can each neuron in the brain be matched by a synthetic neuron from the model? and vice versa?
Neural Data as Feature Representations

Pixel space: $\mathbb{R}^{1000000}$

IT feature space: $\mathbb{R}^{4000(?)}$
Neural Data as Feature Representations

IT feature space

$\mathbf{r}_i = \text{vector of neural responses to stimulus } i$

$\mathbf{r}_j = \text{vector of neural responses to stimulus } j$
IT feature space

$\mathbf{r}_i = \text{vector of neural responses to stimulus } i$

$\mathbf{r}_j = \text{vector of neural responses to stimulus } j$

$$dist(i, j) = \text{euclidean}(\mathbf{r}_i, \mathbf{r}_j)$$

$$= \sqrt{\sum_{k=0}^{N-1} (r_{ik} - r_{jk})^2}$$

$N = \# \text{ of neurons (}\# \text{ of dimensions)}$

$k = \text{variable ranging over neurons}$
Neural Data as Feature Representations

IT feature space

$\mathbf{r}_i = \text{vector of neural responses to stimulus } i$

$\mathbf{r}_j = \text{vector of neural responses to stimulus } j$

Instead of euclidean distance:

$\text{dist}(i, j) = 1 - \text{correlation}(\mathbf{r}_i, \mathbf{r}_j)$

$= 1 - \frac{\text{cov}(\mathbf{r}_i, \mathbf{r}_j)}{\sqrt{\text{var}(\mathbf{r}_i) \cdot \text{var}(\mathbf{r}_j)}}$

$\mathbb{E}_k[\mathbf{r}_i \mathbf{r}_j] - \mathbb{E}_k[\mathbf{r}_i] \mathbb{E}_k[\mathbf{r}_j]$

$\mathbb{E}_k[\mathbf{r}_i^2] - \mathbb{E}_k[\mathbf{r}_i]^2$

expectations over neurons
$M_{ij} = 1 - \text{correlation}(r_i, r_j)$

Low $M_{ij}$ (blue) means neurons think the stimuli are similar.

High $M_{ij}$ (red) means neurons think the two stimuli are different.
RDMs allow comparison of different neural representations on a common stimulus set.

Their structure echoes key features of the functionality of the population code.
RDMs allow comparison of different neural representations on a common stimulus set.

Their structure echoes key features of the functionality of the population code.
Their structure echoes key features of the functionality of the population code.
Representational Similar Analyses

Hierarchical processing in the ventral stream.

Pixels  V1-like  V2-like  V4  IT
Representational Similar Analyses

Model captures diagonal and off-diagonal structure effectively.

![Monkey V4](image1.png)

![HMO Model](image2.png)

![Monkey IT](image3.png)

- animals
- boats
- cars
- chairs
- faces
- fruits
- planes
- tables
Hierarchical processing in the ventral stream.

Representational Similar Analyses

Pixels  V1 like  V2 like  V4  IT

DNN Model
The Population Code

Image Generalization

Object Generalization

Category Generalization

Similarity to IT at High Variation

Spearman r of RDMs

- Controls
- HMO
- V4 Neurons
- IT Neuron Split-Half
The Population Code

Also works for comparing low layers of models to early visual areas

representational similarity

human V1-V3

HCNN Layers

pixels 1 2 3 4 5 6 7

Seyed Khaligh-Razavi

Niko Kriegeskorte
2. Population similarity: is the population code in the model the same as in the neural data?

pros:
   i. task-independent
   ii. easy to measure

cons:
   i. is NOT invariant to linear transform
      —> can easily mismatch for “silly” reasons
Three levels of comparison:

1. Behavioral: does the model (or brain feature set) explain the patterns of behavior?

2. Population similarity: is the population code in the model the same as in the neural data?

3. Single-neuron predictivity: can each neuron in the brain be matched by a synthetic neuron from the model? and vice versa?
Optimize for Performance, Test Against Neurons

**Step 1: Optimize for Task**

- Spatial Convolution over Image Input

**Step 2: Compare to Neural Data**

Visual Recognition Task

- V1
- IT
- V2
- V4

100ms Visual Presentation

Layer 1

Layer 2

Layer 3

Layer 4
Neural predictivity: the ability of a model to predict each individual neural site's activity.
Neural Response Prediction

Some kind of mapping is necessary.
Here, we use linear regression.

\[ T = M \times S \]
Neural predictivity: the ability of model to predict each individual neural site’s activity.

Neural site unit ~ sparse linear combination of model units

Linear regression with fixed training images.

Accuracy = goodness-of-fit on held-out testing images (Cross validated)

Neural predictivity = median accuracy over all units.
Simple linear regression is a case when there's one regressor:

\[ y \sim mx + b \]

Find \( m, b \) that minimize sum-of-squares error:

\[
\sum_{i=0}^{N-1} (y_i - mx_i - b)^2
\]

\[
b = \frac{Cov(x, y)}{Var(x)} = corr(x, y) \cdot \frac{\sigma_y}{\sigma_x}
\]

\( N \) = number of training points
Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

\[ \begin{array}{c}
(n, m) \\
(m, k) \\
(n, k)
\end{array} \]

\(k = \text{number of regressors}\)
\(m = \text{number of dimensions to regress}\)
\(n = \text{number of data points}\)

… and accordingly, has a simple explicit formula solution:

$$\left( \mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{Y}$$

“pseudo-inverse” of \(\mathbf{X}\)

Of course, actually computing the pseudo-inverse is computationally intensive.
Multivariate linear regression is very similar:

\[ \mathbf{Y} = W \cdot \mathbf{X}^T + b \]

\( (n, m) \) \hspace{1cm} \( (m, k) \) \hspace{1cm} \( (n, k) \)

... and accordingly, has a simple explicit formula solution:

\[
\begin{bmatrix}
\begin{bmatrix}
\mathbf{X}^T \mathbf{X}
\end{bmatrix}^{-1} \\
\mathbf{X}^T
\end{bmatrix}
\begin{bmatrix}
\mathbf{X}^T \\
\mathbf{Y}
\end{bmatrix}
\sim \frac{\text{cov}(\mathbf{X}, \mathbf{Y})}{\text{Var}(\mathbf{X})}
\]

"pseudo-inverse" of \( \mathbf{X} \)

\( k = \) number of regressors

\( m = \) number of dimensions to regress

\( n = \) number of data points

Of course, actually computing the pseudo-inverse is computationally intensive.
Multivariate linear regression is very similar:

\[ \mathbf{Y} = \mathbf{W} \cdot \mathbf{X}^T + \mathbf{b} \]

\[ \begin{array}{c}
(\mathbf{n}, \mathbf{m}) \\
(\mathbf{m}, \mathbf{k}) \\
(\mathbf{n}, \mathbf{k})
\end{array} \]

\( k = \text{number of regressors} \)
\( m = \text{number of dimensions to regress} \)
\( n = \text{number of data points} \)

The goal was to minimize:

\[
\sum_{i=0}^{n-1} (Y_i - x_i^T W - b)^2 = \| \mathbf{Y} - \mathbf{X}^T \cdot \mathbf{W} - \mathbf{b} \|
\]

... and accordingly, had the simple explicit formula solution:

\[ \mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \]
Ridge Regression

Multivariate linear regression is very similar:

\[ \mathbf{Y} = \mathbf{W} \cdot \mathbf{X}^T + b \]

- \( k \) = number of regressors
- \( m \) = number of dimensions to regress
- \( n \) = number of data points

Regularization can help prevent overfitting, minimize this loss instead:

\[ \| \mathbf{Y} - \mathbf{X}^T \cdot \mathbf{W} - b \| + \alpha \| \mathbf{W} \|^2 \]

- fitting error
- regularization
Ridge Regression

Multivariate linear regression is very similar:

\[
Y = W \cdot X^T + b
\]

k = number of regressors
m = number of dimensions to regress
n = number of data points

Regularization can help prevent overfitting, minimize this loss instead:

\[
||Y - X^T \cdot W - b|| + \alpha ||W||^2
\]

fitting error regularization

Again, has a simple explicit solution:

\[
W = (X^T X + \alpha \mathbb{I})^{-1} X^T Y
\]
Multivariate linear regression is very similar:

\[ Y = W \cdot X^T + b \]

- \( k \) = number of regressors
- \( m \) = number of dimensions to regress
- \( n \) = number of data points

Regularization can help prevent overfitting, minimize this loss instead:

\[ \| Y - X^T \cdot W - b \| + \alpha \| W \|^2 \]

fitting error  \hspace{1cm} \text{regularization}

Again, has a simple explicit solution: \textbf{Ridge Regression} (aka Tikhonov regression)

\[ W = (X^T X + \alpha I)^{-1} X^T Y \]
Ridge Regression

Multivariate linear regression is very similar:

\[ \mathbf{Y} = \mathbf{W} \cdot \mathbf{X}^T + \mathbf{b} \]

\( k = \) number of regressors
\( m = \) number of dimensions to regress
\( n = \) number of data points

Regularization can help prevent overfitting, minimize this loss instead:

\[ \| \mathbf{Y} - \mathbf{X}^T \cdot \mathbf{W} - \mathbf{b} \| + \alpha \| \mathbf{W} \|^2 \]

\( \| \mathbf{Y} - \mathbf{X}^T \cdot \mathbf{W} - \mathbf{b} \| \) fitting error
\( \alpha \) cross validating the regularization strength

Again, has a simple explicit solution:

\[ \mathbf{W} = (\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^T \mathbf{Y} \]

\textbf{RidgeCV Regression}

(aka Tikhonov regression)
Ridge Regression

\[
W = (X^TX + \alpha \mathbb{I})^{-1} X^T Y
\]

effectively like adding uncorrelated noise of variance \(\alpha\)

makes 0 contribution to the numerator term

like adding a “ridge” along the diagonal of the regressor covariance matrix — that’s why it’s called “ridge”
Lasso Regression

Multivariate linear regression is very similar:

\[ Y = W \cdot X^T + b \]

\((n, m)\) \quad \((m, k)\) \quad \((n, k)\)  

k = number of regressors  
m = number of dimensions to regress  
n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

\[ ||Y - X^T \cdot W - b|| + \alpha||W|| \]

fitting error  
regularization
Multivariate linear regression is very similar:

\[ \mathbf{Y} = \mathbf{W} \cdot \mathbf{X}^T + b \]

\( (n, m) \quad (m, k) \quad (n, k) \)

k = number of regressors
m = number of dimensions to regress
n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

\[ ||\mathbf{Y} - \mathbf{X}^T \cdot \mathbf{W} - b|| + \alpha ||\mathbf{W}|| \]

fitting error  regularization

Does not have a simple solution.

Lasso leads to \textit{sparse} weights.
Lasso Regression

Multivariate linear regression is very similar:

\[
Y = W \cdot X^T + b
\]

\[ (n, m) \quad (m, k) \quad (n, k) \]

\[ k = \text{number of regressors} \]
\[ m = \text{number of dimensions to regress} \]
\[ n = \text{number of data points} \]

For reasons just like in categorization, can help prevent overfitting to minimize:

\[
\|Y - X^T \cdot W - b\| + \alpha \|W\|
\]

fitting error regularization

Does not have a simple solution.

Lasso leads to \textit{sparse} weights.

Lasso Regression

LassoCV Regression
Multivariate linear regression is very similar:

\[ Y = W \cdot X^T + b \]

- \( k = \) number of regressors
- \( m = \) number of dimensions to regress
- \( n = \) number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

\[ ||Y - X^T \cdot W - b|| + \alpha ||W|| \]

- fitting error
- regularization

Does not have a simple solution.

Lasso leads to \textit{sparse} weights.

Lasso Regression

LassoCV Regression

which matters if you care about interpretability of coefficients — not so relevant in neuroscience, but relevant in (e.g.) psychology or economics
Multivariate linear regression is very similar:

\[
\mathbf{Y} = \mathbf{W} \cdot \mathbf{X}^T + \mathbf{b}
\]

\(k = \text{number of regressors}\)
\(m = \text{number of dimensions to regress}\)
\(n = \text{number of data points}\)

For reasons just like in categorization, can help prevent overfitting to minimize:

\[
\|\mathbf{Y} - \mathbf{X}^T \cdot \mathbf{W} - \mathbf{b}\| + \alpha \|\mathbf{W}\|^2 + \beta \|\mathbf{W}\|
\]

fitting error
two regularization terms

Potential advantage:
Can be as good at fitting as Ridge while still allowing some sparse interpretability

**ElasticNet regression**

**ElasticNetCV regression**
Multivariate linear regression is very similar:

\[
Y = W \cdot X^T + b
\]

\(k = \text{number of regressors}\)

\(m = \text{number of dimensions to regress}\)

\(n = \text{number of data points}\)

In Ridge Regression we had:

\[
||Y - X^T \cdot W - b|| + \alpha||W||^2
\]
Partial Least Squares (PLS) Regression

\[ X = TP^T \]
\[ (n, k) \]

\[ Y = UQ^T \]
\[ (n, m) \]

\( k \) = number of regressors

\( m \) = number of dimensions to regress

\( n \) = number of data points
Partial Least Squares (PLS) Regression

\[ X = TP^T \quad (n, k) \]
\[ Y = UQ^T \quad (n, m) \]

- \( T \) (n, L) matrix
- \( U \) (n, L) matrix

- \( k \) = number of regressors
- \( m \) = number of dimensions to regress
- \( n \) = number of data points
Partial Least Squares (PLS) Regression

\[ \mathbf{X} = TP^T \quad \text{(n, k)} \]
\[ \mathbf{Y} = UQ^T \quad \text{(n, m)} \]

\( T \) (n, L) matrix
\( U \) (n, L) matrix
\( P \) (k, L) orthogonal matrix
\( Q \) (m, L) orthogonal matrix

\( k = \text{number of regressors} \)
\( m = \text{number of dimensions to regress} \)
\( n = \text{number of data points} \)
Partial Least Squares (PLS) Regression

\[ \mathbf{X} = T P^T \quad \quad \mathbf{Y} = U Q^T \]

\( (n, k) \) \hspace{1cm} \( (n, m) \)

\( T \) (n, L) matrix \hspace{1cm} \( P \) (k, L) orthogonal matrix

\( U \) (n, L) matrix \hspace{1cm} \( Q \) (m, L) orthogonal matrix

Goal is to choose \( T, U, P \) and \( Q \) to maximize: \( corr(T, U) \)

k = number of regressors

m = number of dimensions to regress

n = number of data points
Partial Least Squares (PLS) Regression

\[ \mathbf{X} = T \mathbf{P}^T \quad \text{(n, k)} \]
\[ \mathbf{Y} = U \mathbf{Q}^T \quad \text{(n, m)} \]

\[ T \text{ (n, L) matrix} \quad P \text{ (k, L) orthogonal matrix} \]
\[ U \text{ (n, L) matrix} \quad Q \text{ (m, L) orthogonal matrix} \]

The goal is to choose \( T, U, P \) and \( Q \) to maximize:\n
\[ \text{corr}(T, U) \]

\( k = \) number of regressors
\( m = \) number of dimensions to regress
\( n = \) number of data points

“trying to put \( X \) and \( Y \) in a common space”
Partial Least Squares (PLS) Regression

\[ \mathbf{X} = T P^T + E \]  \hspace{2cm} \[ \mathbf{Y} = U Q^T + F \]

\begin{align*}
T & \quad (n, L) \text{ matrix} & P & \quad (k, L) \text{ orthogonal matrix} \\
U & \quad (n, L) \text{ matrix} & Q & \quad (m, L) \text{ orthogonal matrix}
\end{align*}

\text{goal is to choose } T, U, P \text{ and } Q \text{ to maximize: } \text{corr}(T, U)

\ldots \text{ and minimize residuals } E \text{ and } F.

\text{k = number of regressors} \\
\text{m = number of dimensions to regress} \\
\text{n = number of data points}
Partial Least Squares (PLS) Regression

\[ X = TP^T + E \quad \text{and} \quad Y = UQ^T + F \]

\( T \) (n, L) matrix  \( P \) (k, L) orthogonal matrix  
\( U \) (n, L) matrix  \( Q \) (m, L) orthogonal matrix

\( \text{corr}(T, U) \)

\[ k = \text{number of regressors} \]
\[ m = \text{number of dimensions to regress} \]
\[ n = \text{number of data points} \]

\( L = \text{how many common dimensions} \)

\( L \) has to be cross-validated

goal is to choose \( T, U, P \) and \( Q \) to maximize:

\( \text{cor}(T, U) \)

\( \text{... and minimize residuals E and F.} \)
Partial Least Squares (PLS) Regression

\[ X = TP^T + E \]  (n, k)

\[ Y = UQ^T + F \]  (n, m)

\[ T \] (n, L) matrix

\[ P \] (k, L) orthogonal matrix

\[ U \] (n, L) matrix

\[ Q \] (m, L) orthogonal matrix

Goal is to choose \( T, U, P \) and \( Q \) to maximize: \( \text{corr}(T, U) \)

...and minimize residuals \( E \) and \( F \).

\[ L = \text{how many common dimensions} \]

has to be cross-validated
Partial Least Squares (PLS) Regression

\[ X = TP^T + E \]
\[ Y = UQ^T + F \]

\( T \) (n, L) matrix  \( P \) (k, L) orthogonal matrix
\( U \) (n, L) matrix  \( Q \) (m, L) orthogonal matrix

\( k = \) number of regressors  \( m = \) number of dimensions to regress
\( n = \) number of data points

goal is to choose \( T, U, P \) and \( Q \) to maximize: \( corr(T, U) \)

…. and minimize residuals \( E \) and \( F \).

\[ Y \sim XQ^T (P^T)^{-1} \]

\( L = \) how many common dimensions has to be cross-validated

“trying to put \( X \) and \( Y \) in a common space”
Partial Least Squares (PLS) Regression

\[
X = TP^T + E \quad \text{(n, k)} \quad Y = UQ^T + F \quad \text{(n, m)}
\]

- \(T\) (n, L) matrix
- \(P\) (k, L) orthogonal matrix
- \(U\) (n, L) matrix
- \(Q\) (m, L) orthogonal matrix

Goal is to choose \(T, U, P\) and \(Q\) to maximize: \(\text{corr}(T, U)\)

... and minimize residuals \(E\) and \(F\).

\[
Y \sim XQ^T(P^T)^{-1}
\]

- \(L = \) how many common dimensions
- \(L\) has to be cross-validated

k = number of regressors

m = number of dimensions to regress

n = number of data points

“trying to put \(X\) and \(Y\) in a common space”

the effective regression weight matrix — really needs pseudo-inverse of \(P^T\)
Regression from DNN

[IPYNB: Regression from trained network]
Neural fits get better over training time on Imagenet, but most of the action is toward the beginning of training.
Model ImageNet Top1 performance is highly correlated with ability to regress IT neural responses
Regression from DNN

Fit increases through the layers until the second-to-last (fc6) layer
Regression from DNN

Fit increases through the layers until the second-to-last (fc6) layer.

Differentiation between layers by this metric increases through training time.
Regression from DNN

Best-fitting layer also captures most neurons by “which layer fits best” metric.
Best-fitting layer also captures most neurons by “which layer fits best” metric.

And again, differentiation increases over training time.
Regression from DNN

There's the distribution of ability to fit neurons.
Regression from DNN

There's the distribution of ability to fit neurons.

...but there's also a distribution of reliabilities. How do we normalize?
To our knowledge best (in terms of neural prediction) feedforward model is 12 layers deep:

- V4 at 6th convolutional layer
- pIT at 7th convolutional layer
- cIT/aIT at layers 8-10, depending on neurons position on A/P axis
Regression from DNN

\( \mathbf{v}_{Tr} \) vector over \textbf{training} images of true trial mean values of neuron

\( \mathbf{v}_{Te} \) vector over \textbf{testing} images of true trial values of neuron
Regression from DNN

\[ \mathbf{v}_{Tr} \] vector over **training** images of true trial mean values of neuron

\[ \mathbf{v}_{Te} \] vector over **testing** images of true trial values of neuron

\[ s_j(Tr) \] vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\[ s_j(Te) \] vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID
Regression from DNN

\[ v_{Tr} \] vector over **training** images of true trial mean values of neuron

\[ v_{Te} \] vector over **testing** images of true trial values of neuron

\[ s_j(Tr) \] vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\[ s_j(Te) \] vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\( M(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

\[ M(v_{Tr}|Te) \] testing image predictions of regressor trained on true responses

\[ M(s_1(Tr)|Te) \] testing image predictions of regressor trained on sample 1 of testing images

NB: just write \( M(x) \) because predictions always on the same \( y \) (the testing images \( Te \))
Regression from DNN

\[ v_{Tr} \quad \text{vector over training images of true trial mean values of neuron} \]

\[ v_{Te} \quad \text{vector over testing images of true trial values of neuron} \]

\[ s_j(Tr) \quad \text{vector over training images of trial mean of } j\text{-th sample of } n \text{ trials of neuron} \]

\[ s_j(Te) \quad \text{vector over testing images of trial mean of } j\text{-th sample of } n \text{ trials of neuron} \]

assume IID

\[ M(x|y) \quad \text{predictions on images } y \text{ of linear regressor trained on responses to images } x \]

Want: \[ \text{Corr}(M(v_{Tr}|Te), v_{Te}) \]

But can't get it (because trials are limited).
Regression from DNN

\( v_{Tr} \) vector over **training** images of true trial mean values of neuron

\( v_{Te} \) vector over **testing** images of true trial values of neuron

\( s_j(Tr) \) vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\( s_j(Te) \) vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\( \mathcal{M}(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

Want: \( \text{Corr}(\mathcal{M}(v_{Tr}|Te), v_{Te}) \) But can’t get it (because trials are limited).

Quasi-transitivity of correlation:

\( \text{Corr}(\mathcal{M}(v_{Tr}), s_2(Te)) \sim \text{Corr}(\mathcal{M}(v_{Tr}), v_{Te}) \cdot \text{Corr}(v_{Te}, s_2(Te)) \)
Regression from DNN

\(v_{Tr}\) vector over training images of true trial mean values of neuron

\(v_{Te}\) vector over testing images of true trial values of neuron

\(s_j(Tr)\) vector over training images of trial mean of \(j\)-th sample of \(n\) trials of neuron

\(s_j(Te)\) vector over testing images of trial mean of \(j\)-th sample of \(n\) trials of neuron

assume IID

\(M(x|y)\) predictions on images \(y\) of linear regressor trained on responses to images \(x\)

Want: \(\text{Corr}(M(v_{Tr}|Te), v_{Te})\) But can’t get it (because trials are limited).

Quasi-transitivity of correlation:

\(\text{Corr}(M(v_{Tr}), s_2(Te)) \sim \text{Corr}(M(v_{Tr}), v_{Te}) \cdot \text{Corr}(v_{Te}, s_2(Te))\)

So re-arranging:

\(\text{Corr}(M(v_{Tr}), v_{Te}) \sim \frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\text{Corr}(v_{Te}, s_2(Te))}\)
Regression from DNN

\[ v_{Tr} \] vector over **training** images of true trial mean values of neuron

\[ v_{Te} \] vector over **testing** images of true trial values of neuron

\[ s_j(Tr) \] vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\[ s_j(Te) \] vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

Assume IID

\[ M(x|y) \] predictions on images \( y \) of linear regressor trained on responses to images \( x \)

Want: \( \text{Corr}(M(v_{Tr}|Te), v_{Te}) \) But can't get it (because trials are limited).

Quasi-transitivity of correlation:

\[ \text{Corr}(M(v_{Tr}), s_2(TE)) \sim \text{Corr}(M(v_{Tr}), v_{Te}) \cdot \text{Corr}(v_{Te}, s_2(TE)) \]

So re-arranging:

\[ \boxed{\text{Corr}(M(v_{Tr}), v_{Te})} \sim \frac{\text{Corr}(M(v_{Tr}), s_2(TE))}{\text{Corr}(v_{Te}, s_2(TE))} \]

corr. of model and sample
corr. of truth and sample
Regression from DNN

\( \mathbf{v}_{Tr} \) vector over \textbf{training} images of true trial mean values of neuron

\( \mathbf{v}_{Te} \) vector over \textbf{testing} images of true trial values of neuron

\( \mathbf{s}_j(Tr) \) vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\( \mathbf{s}_j(Te) \) vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\( M(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

\[
\frac{\text{Corr}(M(\mathbf{v}_{Tr}), \mathbf{s}_2(Te))}{\text{Corr}(\mathbf{v}_{Te}, \mathbf{s}_2(Te))}
\]
Regression from DNN

\( v_{Tr} \) vector over \textbf{training} images of true trial mean values of neuron

\( v_{Te} \) vector over \textbf{testing} images of true trial values of neuron

\( s_j(Tr) \) vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\( s_j(Te) \) vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\( M(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

\[
\frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\text{Corr}(v_{Te}, s_2(Te))} = \text{Corr}(s_1(x), s_2(x)) \sim \text{Corr}(v_x, s_2(x))^2
\]

but notice:
Regression from DNN

$v_{Tr}$ vector over \textbf{training} images of true trial mean values of neuron

$v_{Te}$ vector over \textbf{testing} images of true trial values of neuron

$s_{j}(Tr)$ vector over training images of trial mean of $j$-th sample of $n$ trials of neuron

$s_{j}(Te)$ vector over testing images of trial mean of $j$-th sample of $n$ trials of neuron

assume IID

$M(x|y)$ predictions on images $y$ of linear regressor trained on responses to images $x$

$$\frac{\text{Corr}(M(v_{Tr}), s_2(Tr))}{\text{Corr}(v_{Te}, s_2(Tr))}$$

$$\frac{\text{Corr}(M(v_{Tr}), s_2(Tr))}{\sqrt{\text{Corr}(s_1(Tr), s_2(Tr))}}$$

but notice:

$$\text{Corr}(s_1(x), s_2(x)) \sim \text{Corr}(v_x, s_2(x))^2$$

thus
Regression from DNN

\( v_{Tr} \) vector over **training** images of true trial mean values of neuron

\( v_{Te} \) vector over **testing** images of true trial values of neuron

\( s_j(T_{Tr}) \) vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\( s_j(T_{Te}) \) vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\( M(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

\[
\frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\text{Corr}(v_{Te}, s_2(Te))}
\]

but notice:

\[
\text{Corr}(s_1(x), s_2(x)) \sim \text{Corr}(v_x, s_2(x))^2
\]

thus

\[
\frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\sqrt{\text{Corr}(s_1(Te), s_2(Te))}}
\]

correlation of two different samples
Regression from DNN

$v_{Tr}$ vector over **training** images of true trial mean values of neuron

$v_{Te}$ vector over **testing** images of true trial values of neuron

$s_j(Tr)$ vector over training images of trial mean of $j$-th sample of $n$ trials of neuron

$s_j(Te)$ vector over testing images of trial mean of $j$-th sample of $n$ trials of neuron

assume IID

$M(x|y)$ predictions on images $y$ of linear regressor trained on responses to images $x$

\[
\frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\text{Corr}(v_{Te}, s_2(Te))}
\]

\[
\sqrt{\frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\text{Corr}(s_1(Te), s_2(Te))}}
\]

correlation of two different samples

but notice:

\[
\text{Corr}(s_1(x), s_2(x)) \sim \text{Corr}(v_x, s_2(x))^2
\]

thus

\[
\text{Corr}(s_1(x), s_2(x)) \sim \text{Corr}(v_x, s_2(x))^2
\]

still, this is not directly available
Regression from DNN

\( v_{Tr} \) vector over **training** images of true trial mean values of neuron

\( v_{Te} \) vector over **testing** images of true trial values of neuron

\( s_j(Tr) \) vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\( s_j(Te) \) vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\( M(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

\[
\frac{\text{Corr}(M(v_{Tr}), s_2(Te))}{\text{Corr}(v_{Te}, s_2(Te))}
\]

but notice:

\[
\text{Corr}(s_1(x), s_2(x)) \sim \text{Corr}(v_x, s_2(x))^2
\]

thus

\[
\sqrt{\text{Corr}(s_1(Te), s_2(Te))}
\]

still, this is not directly available

Apply same sequence of arguments, to model term!
Regression from DNN

\( \mathbf{v}_{Tr} \) vector over **training** images of true trial mean values of neuron

\( \mathbf{v}_{Te} \) vector over **testing** images of true trial values of neuron

\( s_j(Tr) \) vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\( s_j(Te) \) vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

Assume IID

\( \mathbf{M}(x|y) \) predictions on images \( y \) of linear regressor trained on responses to images \( x \)

End up with:

\[
\frac{\text{Corr}(\mathbf{M}(s_1(Tr)), s_2(Te))}{\sqrt{\text{Corr}(\mathbf{M}(s_1(Tr)), \mathbf{M}(s_2(Tr))) \cdot \text{Corr}(s_1(Te), s_2(Te))}}
\]

(1)
Regression from DNN

\[ v_{Tr} \] vector over **training** images of true trial mean values of neuron

\[ v_{Te} \] vector over **testing** images of true trial values of neuron

\[ s_j(Tr) \] vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\[ s_j(Te) \] vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\[ M(x|y) \] predictions on images \( y \) of linear regressor trained on responses to images \( x \)

End up with:

\[ \frac{\text{Corr}(M(s_1(Tr)), s_2(Te))}{\sqrt{\text{Corr}(M(s_1(Tr)), M(s_2(Tr))) \cdot \text{Corr}(s_1(Te), s_2(Te))}} \]

(1)

\text{corr: between testing sample and predictions of regressor trained on a training sample}

\text{corr: between test predictions of two regressors trained on two different training samples}

\text{the data reliability itself}
Regression from DNN

\[ \mathbf{v}_{Tr} \] vector over \textbf{training} images of true trial mean values of neuron

\[ \mathbf{v}_{Te} \] vector over \textbf{testing} images of true trial values of neuron

\[ s_j(Tr) \] vector over training images of trial mean of \( j \)-th sample of \( n \) trials of neuron

\[ s_j(Te) \] vector over testing images of trial mean of \( j \)-th sample of \( n \) trials of neuron

assume IID

\[ M(x|y) \] predictions on images \( y \) of linear regressor trained on responses to images \( x \)

End up with:

\[ \frac{\text{Corr}(M(s_1(Tr)), s_2(Te))}{\sqrt{\text{Corr}(M(s_1(Tr)), M(s_2(Tr))) \cdot \text{Corr}(s_1(Te), s_2(Te))}} \] (1)

corr: between test predictions of two regressors trained on two different training samples

often OK to assume \( \sim 1 \) (but you should check whenever possible)
Regression from DNN

\(v_{Tr}\) vector over **training** images of true trial mean values of neuron

\(v_{Te}\) vector over **testing** images of true trial values of neuron

\(s_j(Tr)\) vector over training images of trial mean of \(j\)-th sample of \(n\) trials of neuron

\(s_j(Te)\) vector over testing images of trial mean of \(j\)-th sample of \(n\) trials of neuron

assume IID

\(M(x|y)\) predictions on images \(y\) of linear regressor trained on responses to images \(x\)

Leading to

\[
\text{Corr}(M(s_1(Tr)), s_2(Te)) \quad \sqrt{\text{Corr}(s_1(Te), s_2(Te))}
\]

(1a)

corr. between testing sample and predictions of regressor trained on a training sample

the data reliability itself
Regression from DNN

each dot is a neuron
correction computed using formula (1a)
Source neural network has many features, so typically need to subsample to make regressions feasible.
Source neural network has many features, so typically need to subsample to make regressions feasible.

also, when comparing between models want to compare with same number of regression features.

Neural Recordings from IT and V4
Source neural network has many features, so typically need to subsample to make regressions feasible.

Also, when comparing between models want to compare with same number of regression features.

1. Random subsampling

   Easy, but bad for conv. layers — not really representative sample.
Source neural network has many features, so typically need to subsample to make regressions feasible

also, when comparing between models want to compare with same number of regression features

1. Random subsampling  
   
   easy, but bad for conv. layers — not really representative sample

2. PCA the model features first or some other dimension reduction procedure

Neural Recordings from IT and V4
Deep convolutional models improve predictions of macaque V1 responses to natural images

Santiago A. Cadena\textsuperscript{1,3,6,\dagger}, George H. Denfield\textsuperscript{4,6}, Edgar Y. Walker\textsuperscript{4,6}, Leon A. Gatys\textsuperscript{1,3}, Andreas S. Tolias\textsuperscript{3,4,5,6,\dagger}, Matthias Bethge\textsuperscript{1,2,3,6,\dagger}, and Alexander S. Ecker\textsuperscript{1,3,6,\dagger}
Regularizing the Regression Better

\[ x = \text{input image} \]
\[ y = \text{neuron's response} \]
\[ f(x) = \text{model response} \]

\[
\log[r(x)] = \sum w_i f_i(x) + b
\]

\[ w_i, b = \text{weights to learn} \]

Optimize via gradient descent:

\[ \mathcal{L} = -\sum y \log r(x) + r(x) \]

cross-entropy loss
Regularizing the Regression Better

\[
\log[r(x)] = \sum w_i f_i(x) + b \quad w_i, b = \text{weights to learn}
\]

\[
\mathcal{L} = - \sum y \log r(x) + r(x)
\]
Regularizing the Regression Better

\[
\log[r(x)] = \sum w_if_i(x) + b \quad \text{\( w_i, b = \text{weights to learn} \)}
\]

\[
\mathcal{L} = -\sum y \log r(x) + r(x) + \mathcal{L}_{\text{sparse}}
\]

\[
\mathcal{L}_{\text{sparse}} = \lambda_{\text{sparse}} \sum |w_i|
\]

Lots of features in early layers of convnet:

\[40 \times 40 \times 64 > 100k\]
Regularizing the Regression Better

\[ \log[r(x)] = \sum w_i f_i(x) + b \quad w_i, b = \text{weights to learn} \]

\[ \mathcal{L} = - \sum y \log r(x) + r(x) + \mathcal{L}_{\text{sparse}} + \mathcal{L}_{\text{laplace}} \]

\[ \mathcal{L}_{\text{sparse}} = \lambda_{\text{sparse}} \sum |w_i| \quad \text{Lots of features in early layers of convnet:} \quad 40 \times 40 \times 64 > 100k \]

\[ \mathcal{L}_{\text{laplace}} = \lambda_{\text{laplace}} \sqrt{\sum (w_{i,j,k} * L)_{i,j}^2} \quad \text{smoothness prior, since V1 is retinotopic} \]

\[ w_i = w_{ijk} \text{ where } i, j \text{ are the spatial dimensions of the kernel and } k = \text{channel} \]

\[ L = \begin{bmatrix}
  0 & -1 & 0 \\
  -1 & 4 & -1 \\
  0 & -1 & 0
\end{bmatrix} \]
Regularizing the Regression Better

\[ \log[r(x)] = \sum w_i f_i(x) + b \quad w_i, b = \text{weights to learn} \]

\[ \mathcal{L} = -\sum y \log r(x) + r(x) + \mathcal{L}_{\text{sparse}} + \mathcal{L}_{\text{laplace}} + \mathcal{L}_{\text{group}} \]

\[ \mathcal{L}_{\text{sparse}} = \lambda_{\text{sparse}} \sum |w_i| \quad \text{Lots of features in early layers of convnet:} \quad 40 \times 40 \times 64 > 100k \]

\[ \mathcal{L}_{\text{laplace}} = \lambda_{\text{laplace}} \sqrt{\sum (w_{i:j:k} \ast L)_{i,j}^2} \quad \text{smoothness prior, since V1 is retinotopic} \]

\[ \mathcal{L}_{\text{group}} = \lambda_{\text{group}} \sum_k \sqrt{\sum_{ij} w_{ijk}^2} \quad \text{encouragement to *pool*} \]
Regularizing the Regression Better

\[
\log[r(x)] = \sum w_i f_i(x) + b \quad \text{w, b = weights to learn}
\]

\[
\mathcal{L} = -\sum y \log r(x) + r(x) + \mathcal{L}_{sparse} + \mathcal{L}_{laplace} + \mathcal{L}_{group}
\]

\[
\mathcal{L}_{sparse} = \lambda_{sparse} \sum |w_i| \quad \text{Lots of features in early layers of convnet: 40 \times 40 \times 64 > 100k}
\]

\[
\mathcal{L}_{laplace} = \lambda_{laplace} \sqrt{\sum (w_{:,\cdot,k} \ast L)^2_{i,j}} \quad \text{smoothness prior, since V1 is retinotopic}
\]

\[
\mathcal{L}_{group} = \lambda_{group} \sum_k \sqrt{\sum_{ij} w_{ijk}^2} \quad \text{encouragement to *pool*}
\]

lambda’s chosen in cross-validated fashion
Regularizing the Regression Better

\[ \log[r(x)] = \sum w_i f_i(x) + b \quad \text{with } w_i, b = \text{weights to learn} \]

\[ L = - \sum y \log r(x) + r(x) + L_{\text{sparse}} + L_{\text{laplace}} + L_{\text{group}} \]

\[ L_{\text{sparse}} = \lambda_{\text{sparse}} \sum |w_i| \quad \text{Lots of features in early layers of convnet:} \quad 40 \times 40 \times 64 > 100k \]

\[ L_{\text{laplace}} = \lambda_{\text{laplace}} \sqrt{\sum (w_{i,j,k} * L)_{i,j}^2} \quad \text{smoothness prior, since V1 is retinotopic} \]

\[ L_{\text{group}} = \lambda_{\text{group}} \sum_k \sqrt{\sum_{i,j} w_{i,j,k}^2} \quad \text{encouragement to *pool*} \]

\[ \log[r(x)] = \sum w_i f_i(x) + b \quad \text{MOST IMPORTANT: assume regression weights are spatial separable.} \]
Neural system identification for large populations separating “what” and “where”

David A. Klindt *, Alexander S. Ecker *, Thomas Euler 1–3, Matthias Bethge 1,2,4,6
* Authors contributed equally
1 Centre for Integrative Neuroscience, University of Tübingen, Germany
2 Bernstein Center for Computational Neuroscience, University of Tübingen, Germany
3 Institute for Ophthalmic Research, University of Tübingen, Germany
4 Institute for Theoretical Physics, University of Tübingen, Germany
5 Max Planck Institute for Biological Cybernetics, Tübingen, Germany
6 Center for Neuroscience and Artificial Intelligence, Baylor College of Medicine, Houston, USA
klindt.david@gmail.com, alexander.ecker@uni-tuebingen.de,
thomas.euler@cin.uni-tuebingen.de, matthias.bethge@bethgelab.org

Figure 2: Our proposed CNN architecture in its simplest form. It consists of a feature space module and a readout layer. The feature space is extracted via one or more convolutional layers (here one is shown). The readout layer computes for each neuron a weighted sum over the entire feature space. To keep the number of parameters tractable and facilitate interpretability, we factorize the readout into a location mask and a vector of feature weights, which are both encouraged to be sparse by regularizing with L1 penalty.
Regularizing the Regression Better

\[ \log[r(x)] = \sum w_i f_i(x) + b \quad w_i, b = \text{weights to learn} \]

**MOST IMPORTANT:** assume regression weights are space / channel separable.

Dramatically reduces number of regression parameters, respecting the retinotopic structure of convolutional layers (and the ventral stream)