

Tilting the playing field: Dynamical loss functions for machine learning

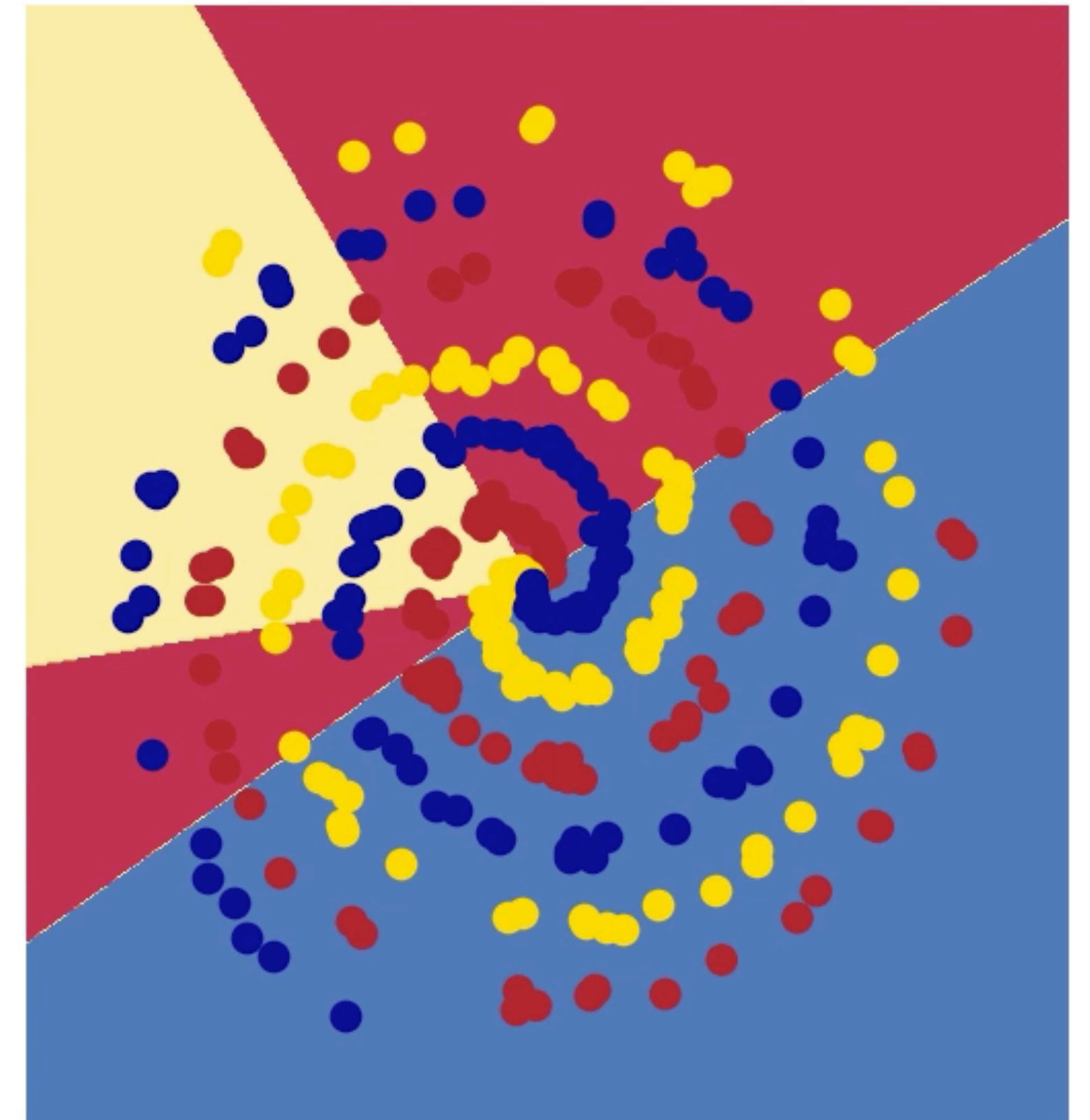
Miguel Ruiz-García^{1,2}, Ge Zhang³, Samuel S. Schoenholz⁴, and Andrea J. Liu³

¹Universidad Politécnica de Madrid

²Universidad Carlos III de Madrid

³University of Pennsylvania

⁴Google Research: Brain Team



uc3m

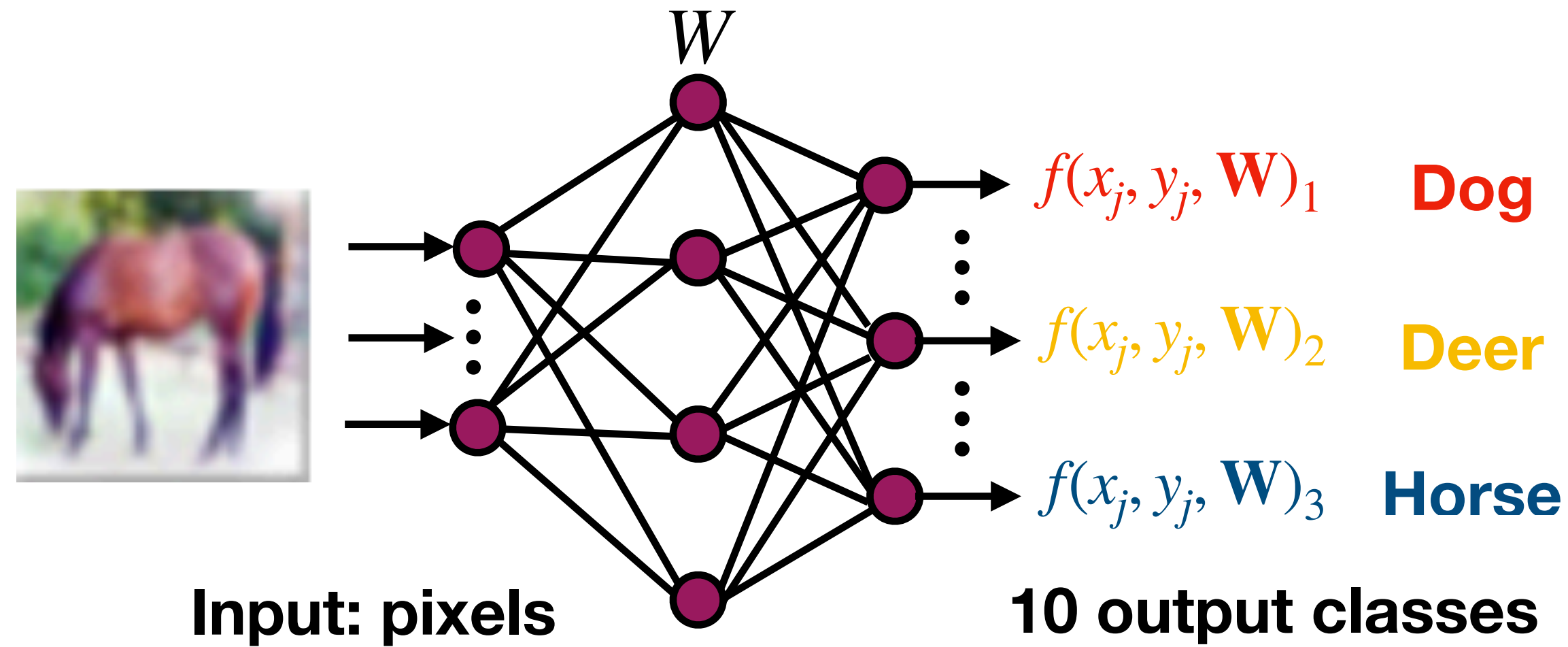
Universidad
Carlos III
de Madrid



POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID

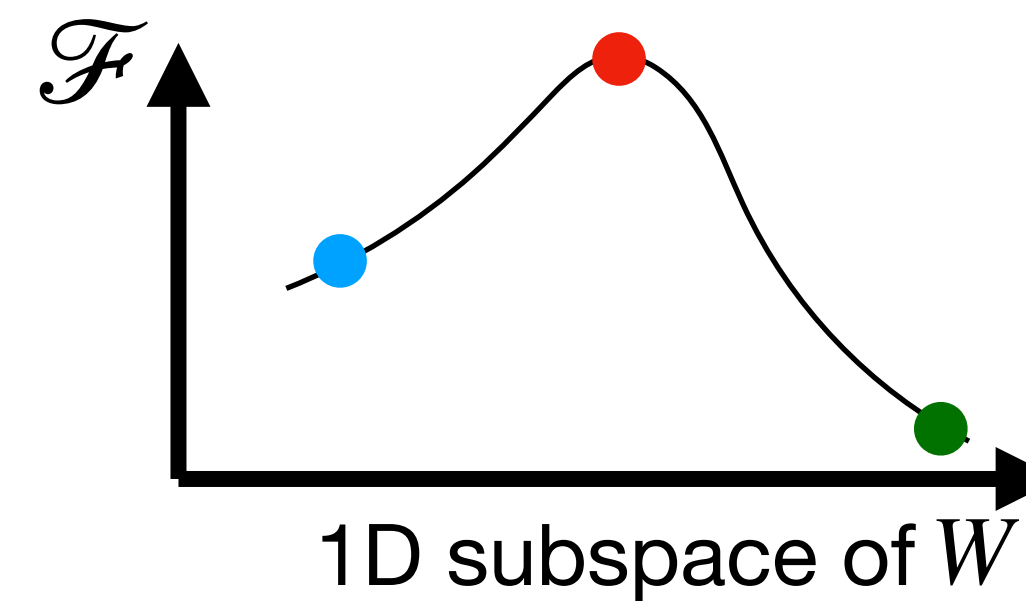
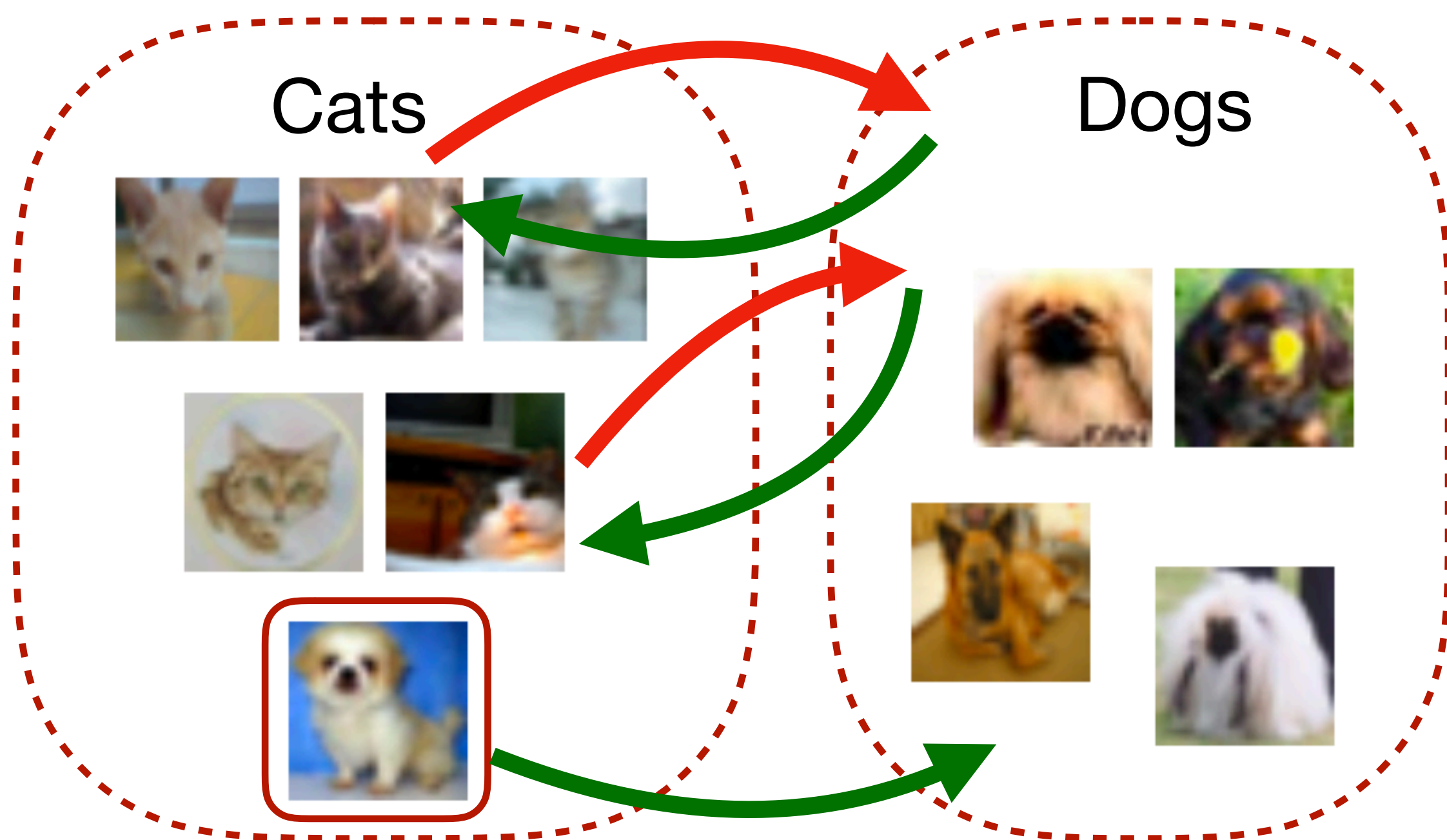
Supervised learning: data classification



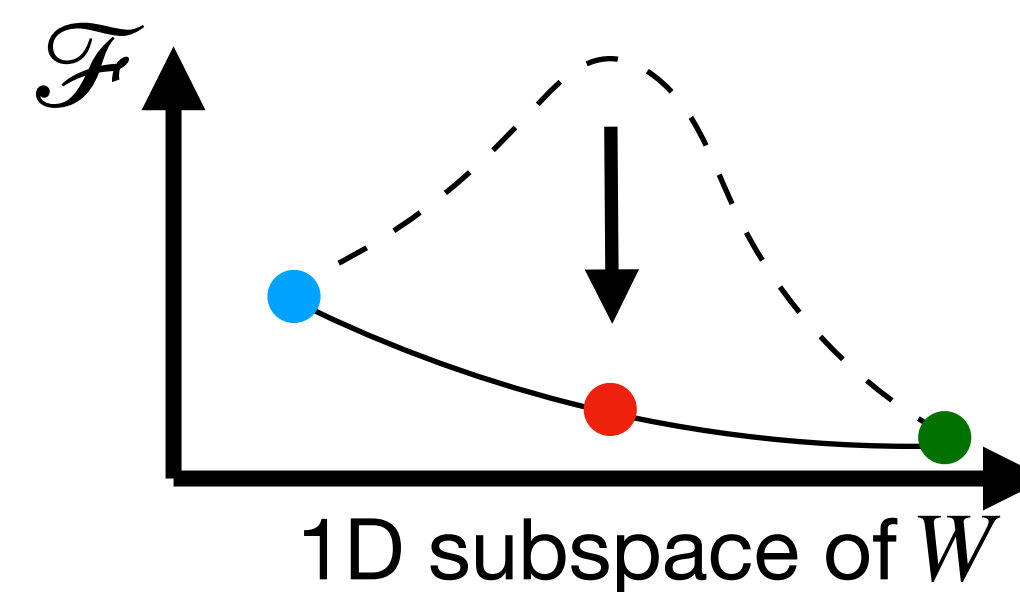
Loss function:

$$\mathcal{F}_i = \sum_{j \in P} \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$

Probability of sample belonging to correct class





Correcting the misclassified dog implies the temporary misclassification of two cats -> Frustration



If we temporarily decreased the penalty for misclassifying cats we could ease the frustration.

Borrowing simple ideas from human learning:

Topics inside every subject are progressively introduced

Grades K–1 	Grades 2–3 	Grades 4–6 
Early Math Concepts	Place Value	Fractions
Whole Number Concepts	Whole Number Operations	Decimals

Deep learning analogue: Curriculum learning

Train the model using samples organized in a *meaningful order*

Difficult to implement: One needs to assess the “difficulty” of each sample inside each class

Bengio et al. ICML 2009

Something much simpler: Subjects are alternated during the week

Language B	Music	Language A	Language B	Humanities
Language B	Language A	Language A	Technology	Humanities
RECESS	RECESS	RECESS	RECESS	RECESS
Math	Language A	P.E.	Language A	Language B
Math	Language B	Language B	Language A	Language B
LUNCH	LUNCH	LUNCH	LUNCH	LUNCH
Technology	Math	Math	Math	Language A

This is much simpler than creating a curriculum!

Is there a machine learning analogy?

Can we alternatively focus more on each class?

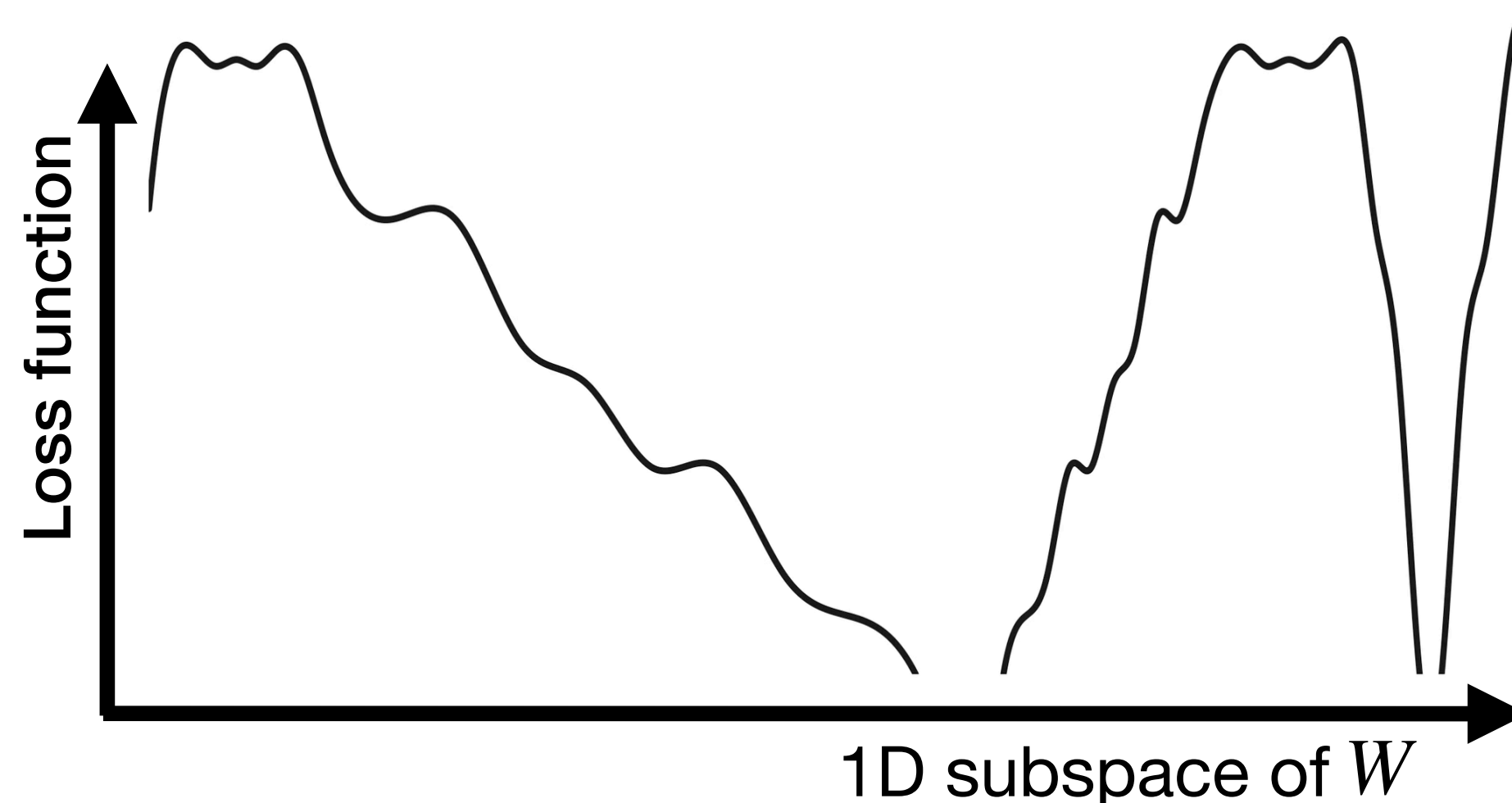
Creating a dynamical loss function

$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$

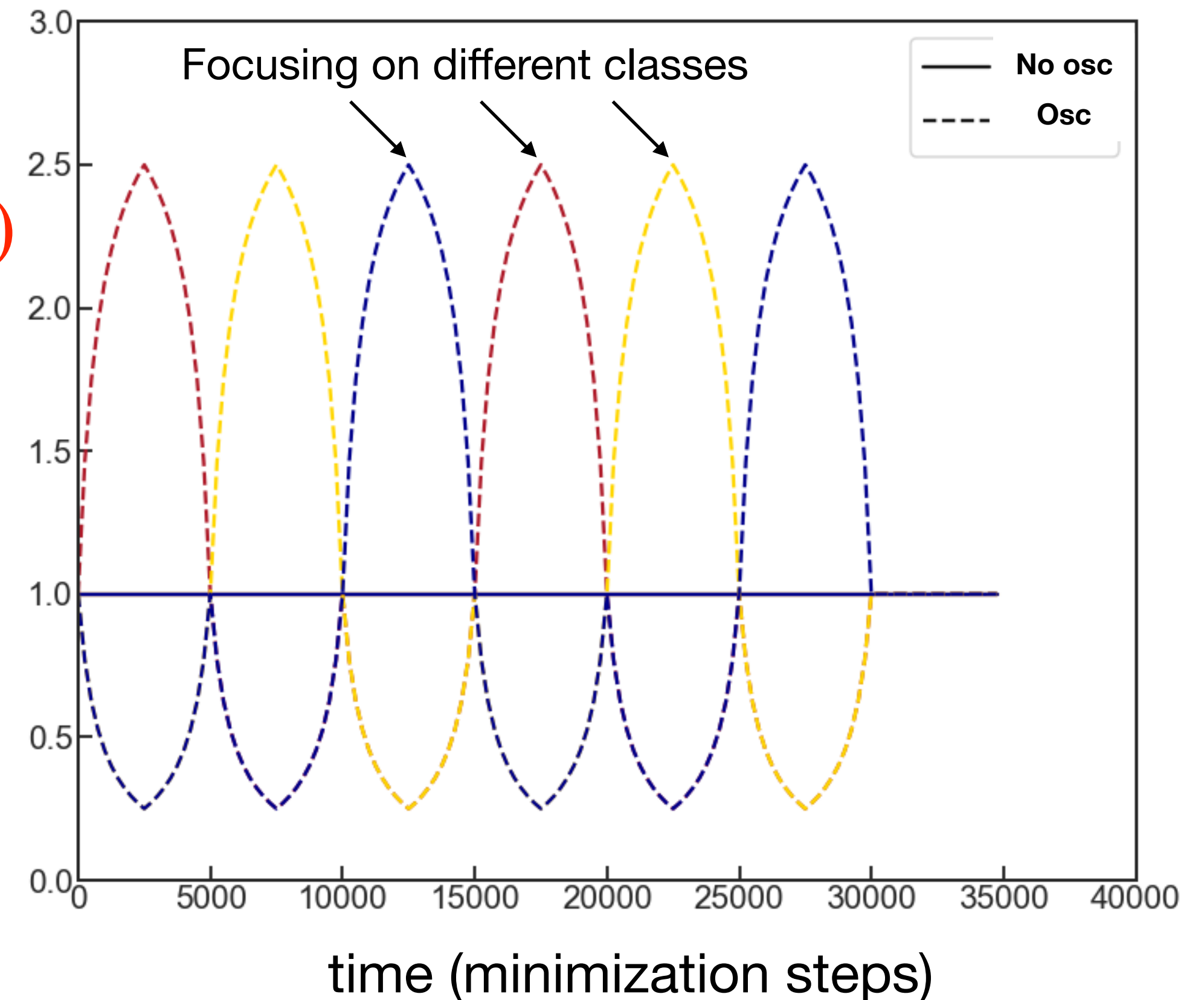
The contribution of each **class** to the loss function oscillates with time

Extremely easy to implement (data already classified in classes)
 Two new hyper-parameters: period and amplitude
 Conserves the global minima

Toy picture:



$\Gamma_c(t)$

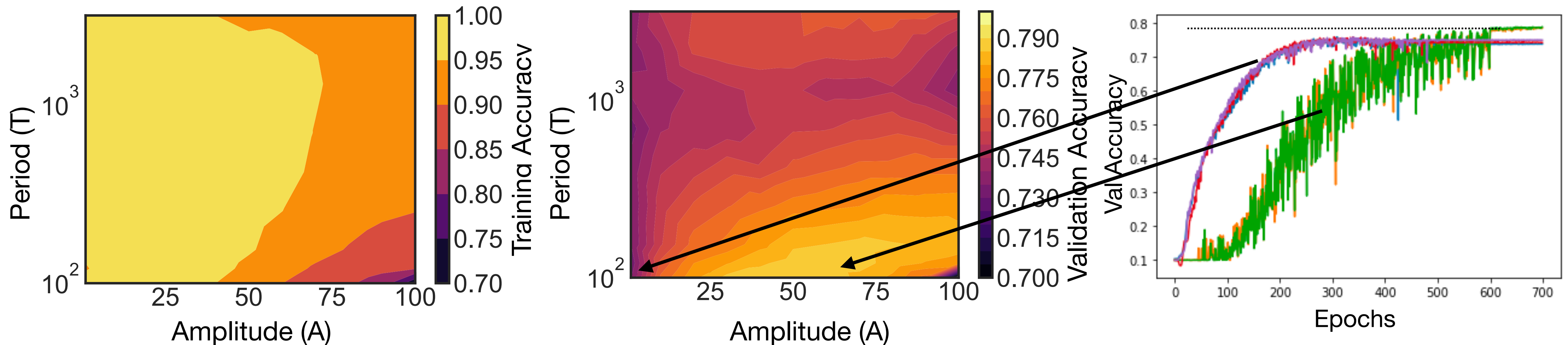


Oscillates the landscape without changing the global minima

Will this improve learning?

Dynamical loss function and CIFAR10

(Myrtle5-64channels, SGD + Momentum, Learning rate schedule...)

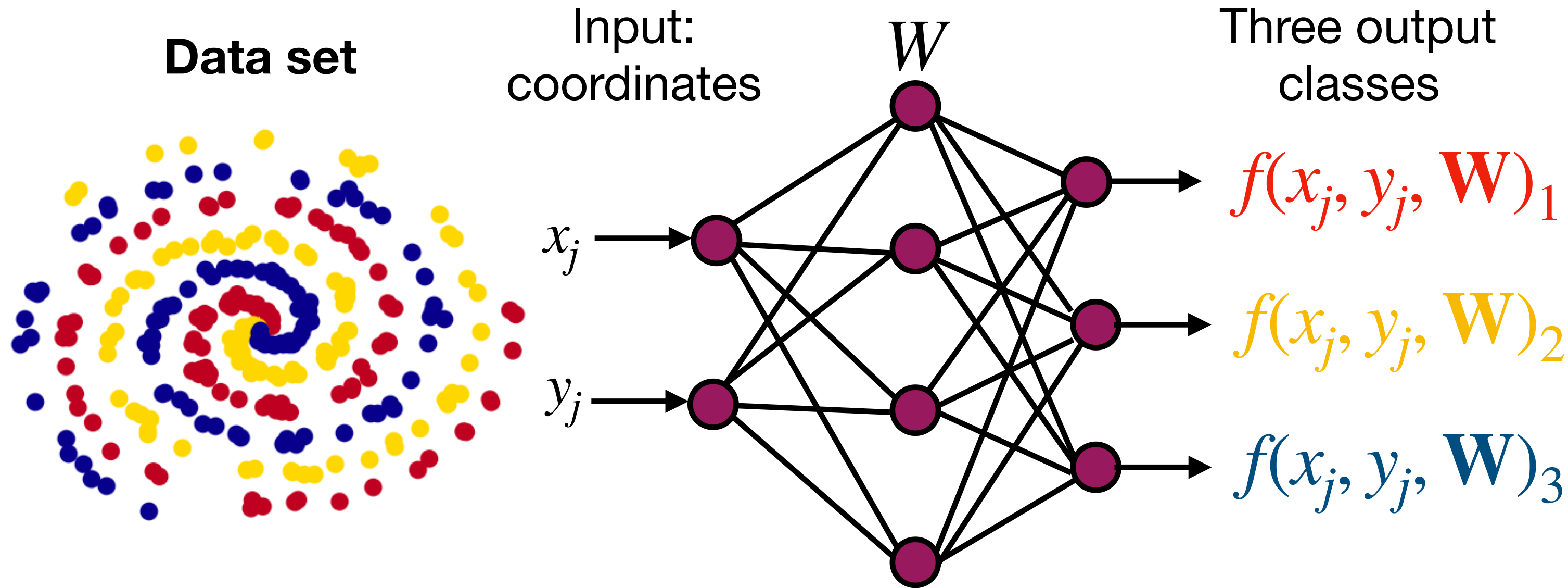


In the overparametrized regime the dynamical loss function improves generalization!

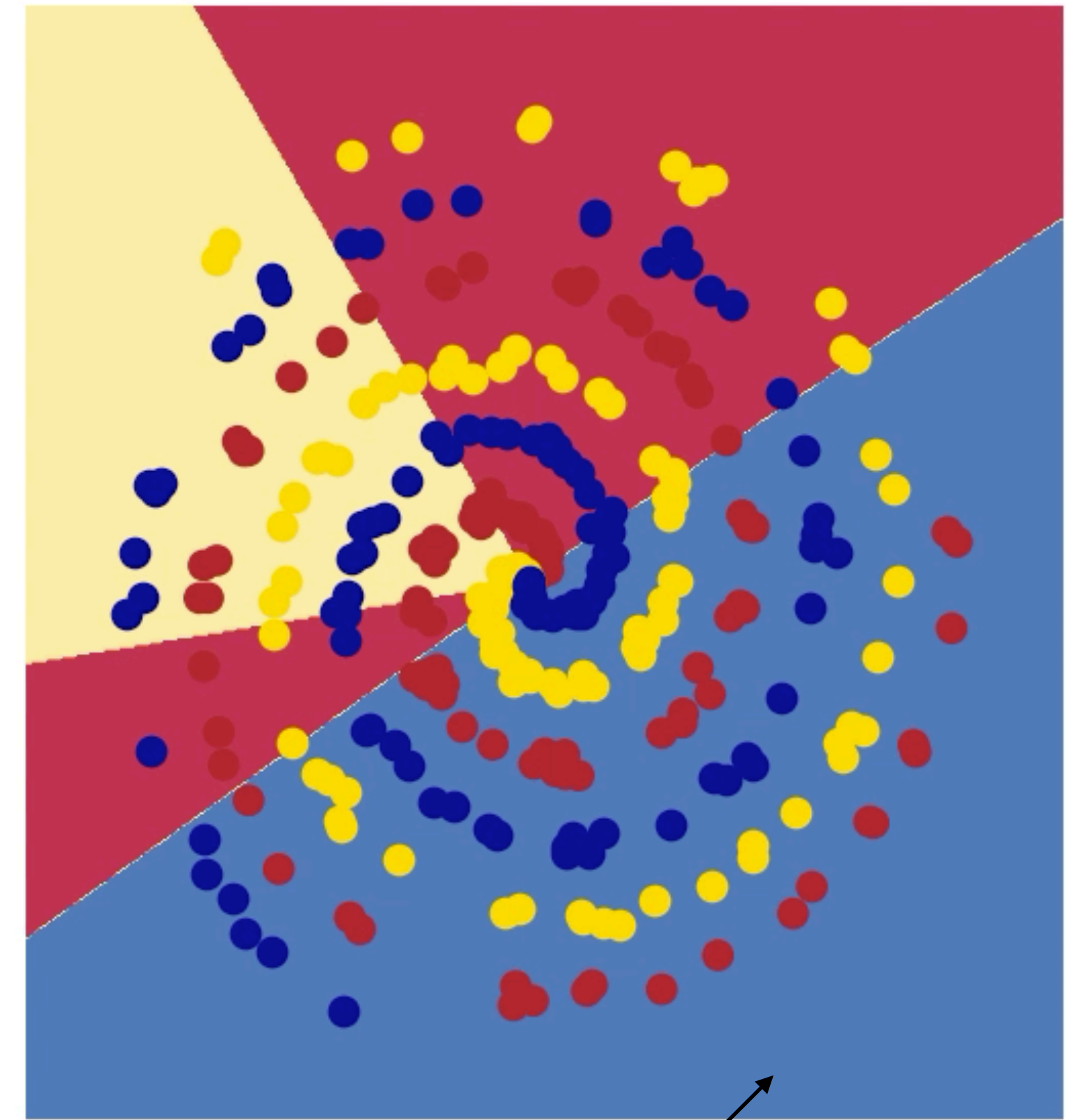
Cycling over classes -> learning slower but better

Simple model and dataset

(One hidden layer, full batch gradient descent)

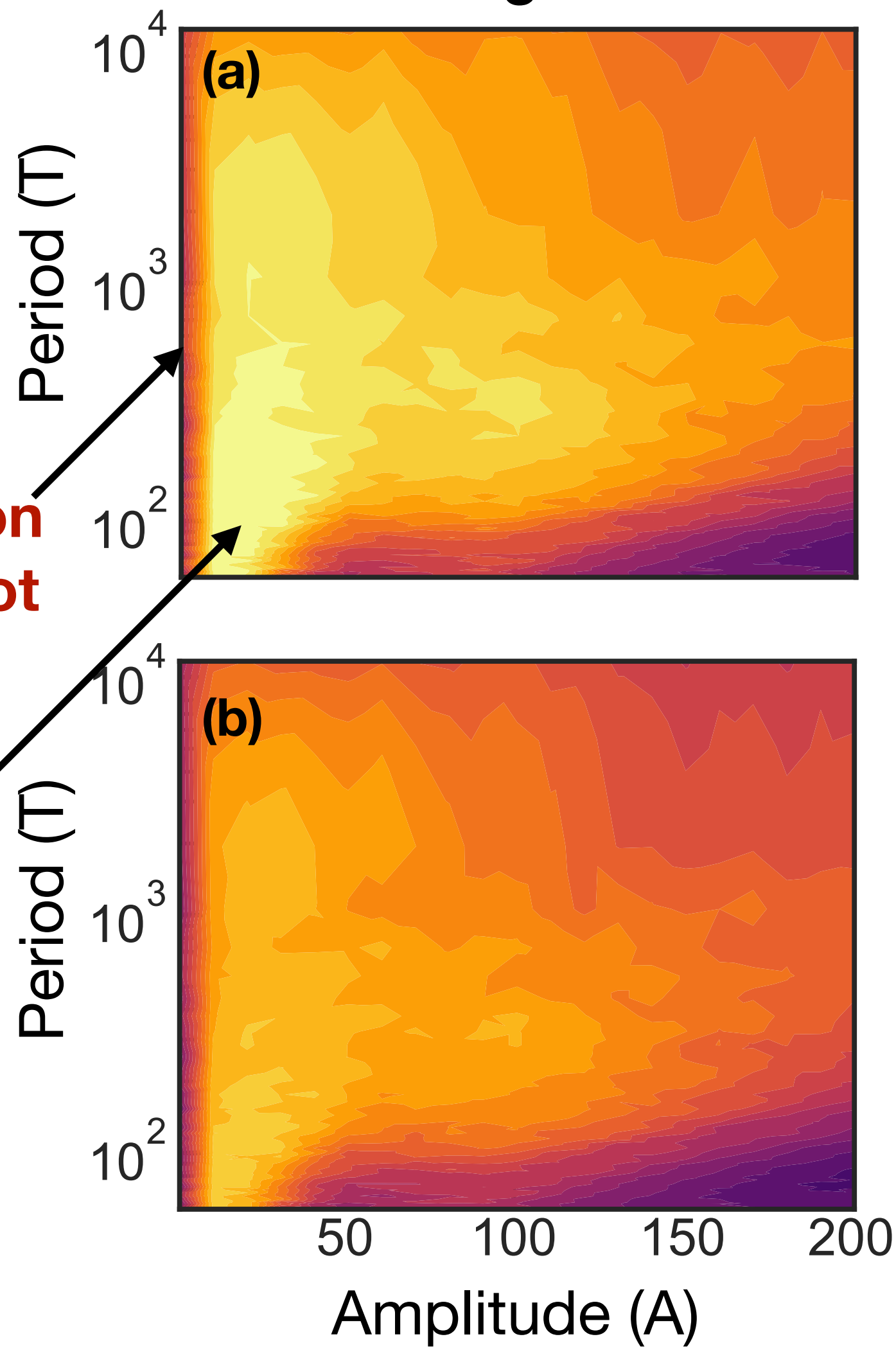


$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$

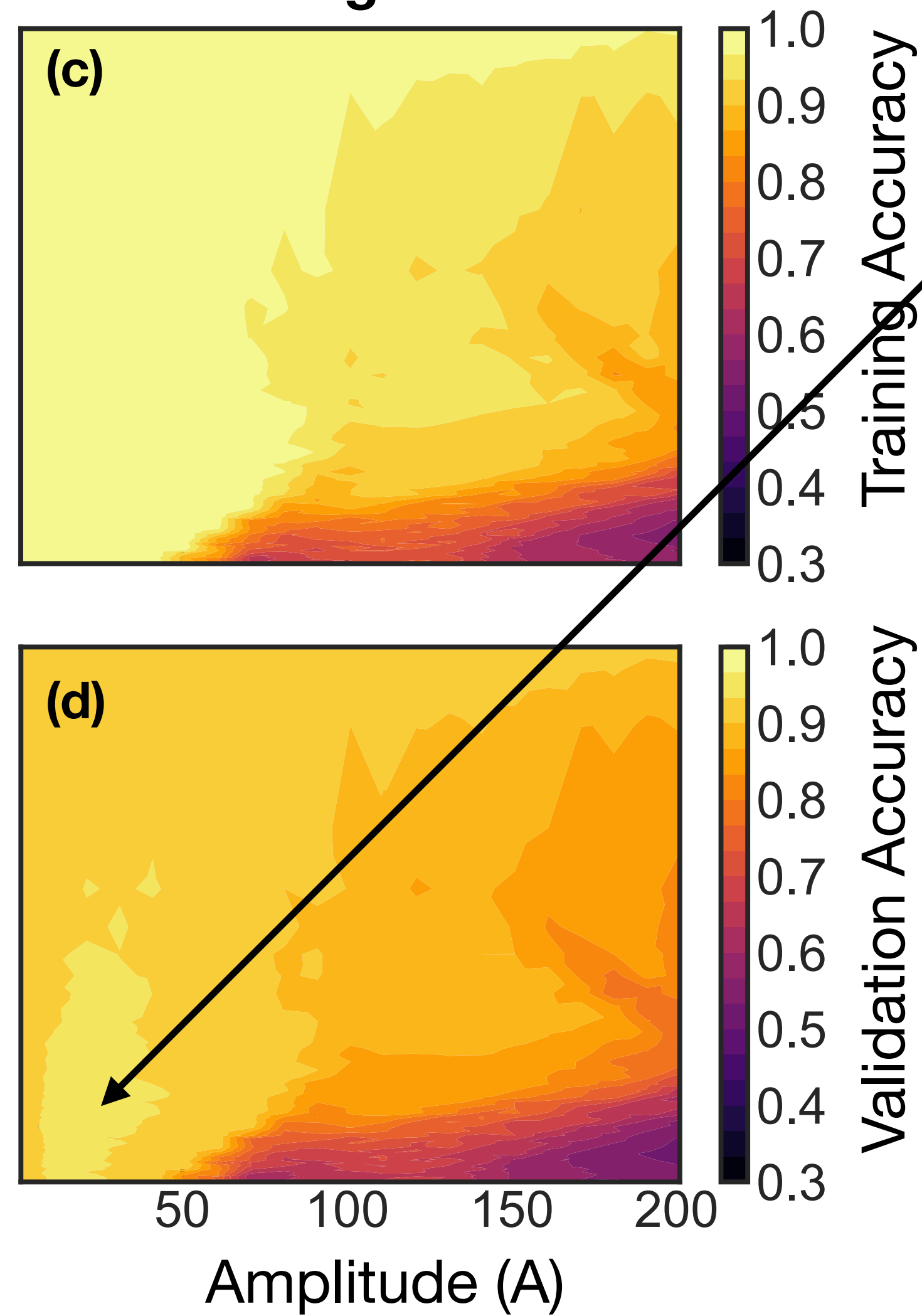


Background color -> predicted class
for any point in the plane

**100 hidden units
Underparametrized
regime**



**1000 hidden units
Overparametrized
regime**



**With a standard
(static) loss function
the model could not
learn**

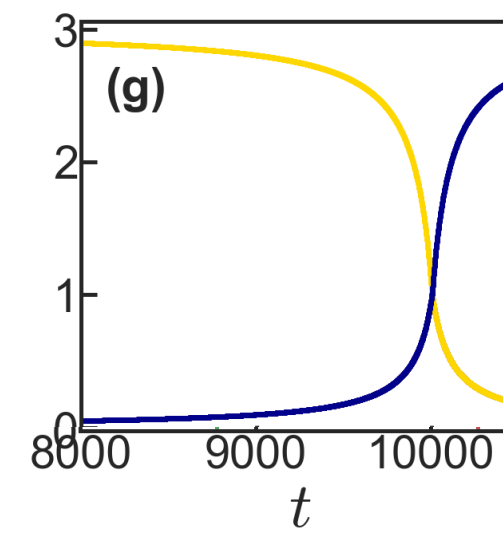
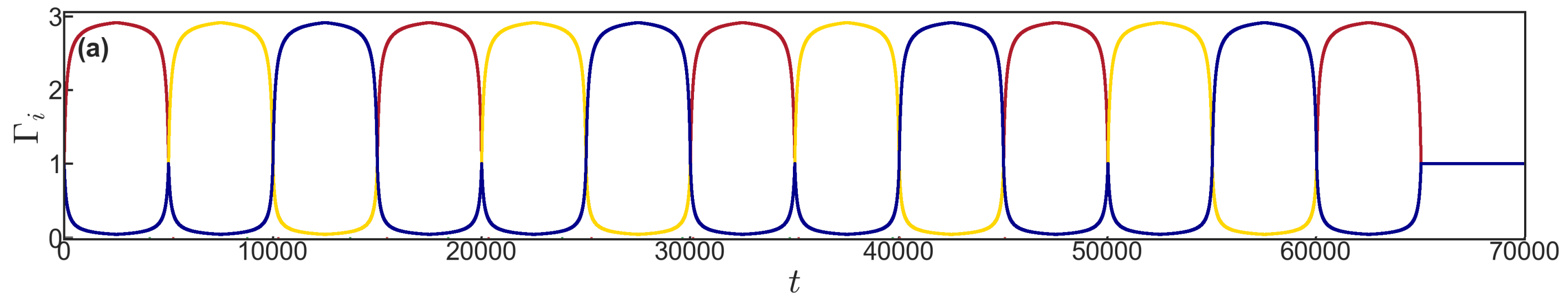
**But the
dynamical loss
function enables
learning!**

**There is a region
where
generalization
improves!**

Training Accuracy

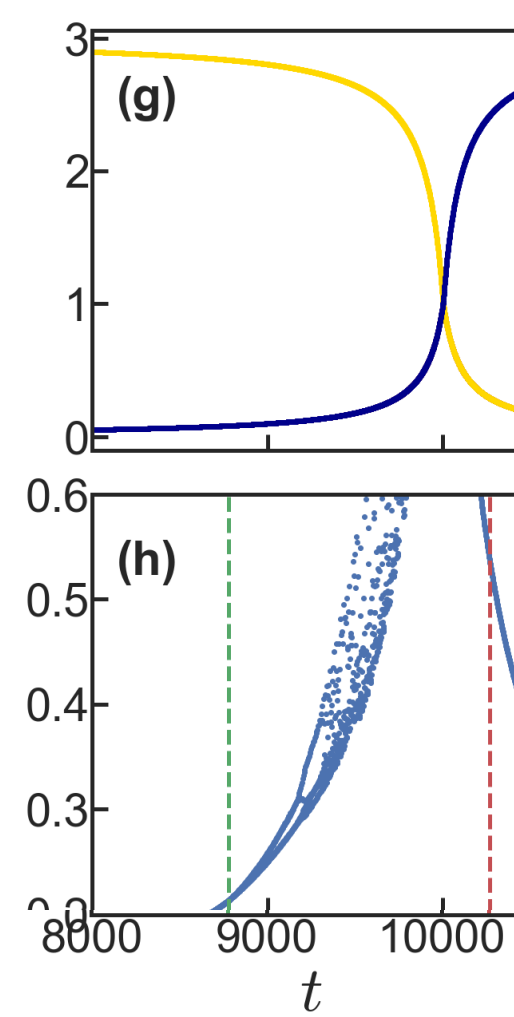
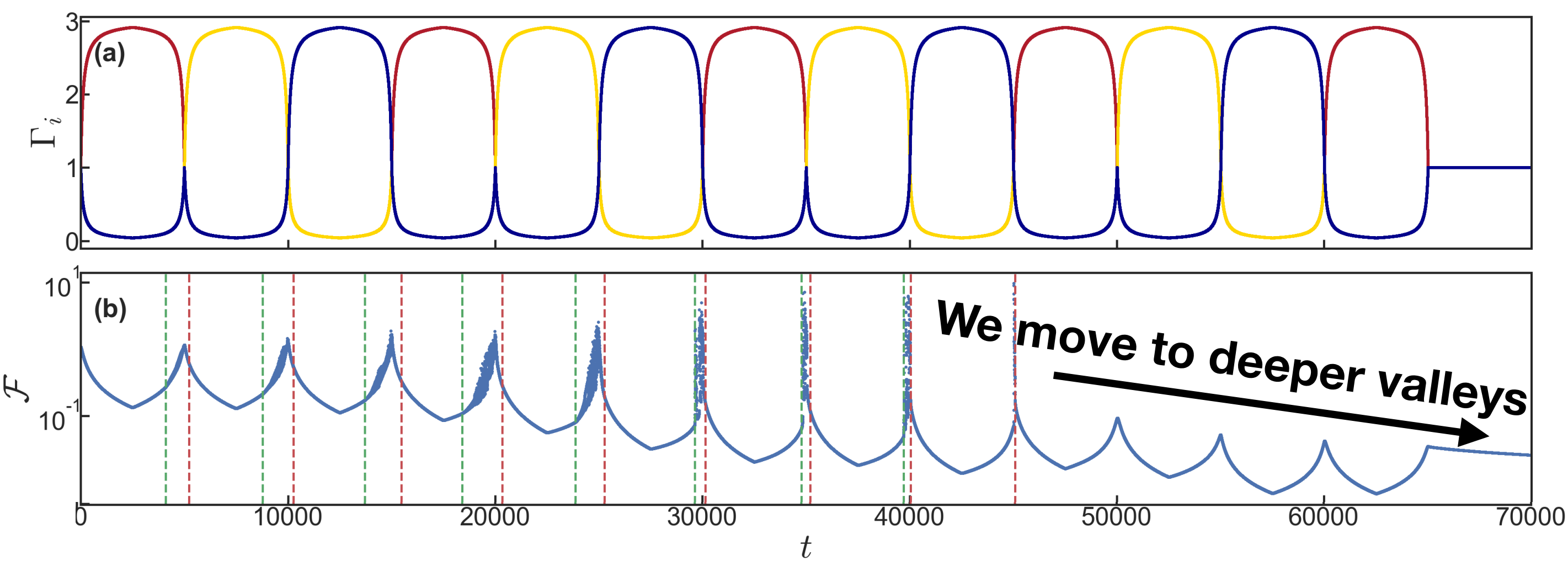
Validation Accuracy

Coupled dynamics: the system is descending a landscape that is oscillating...



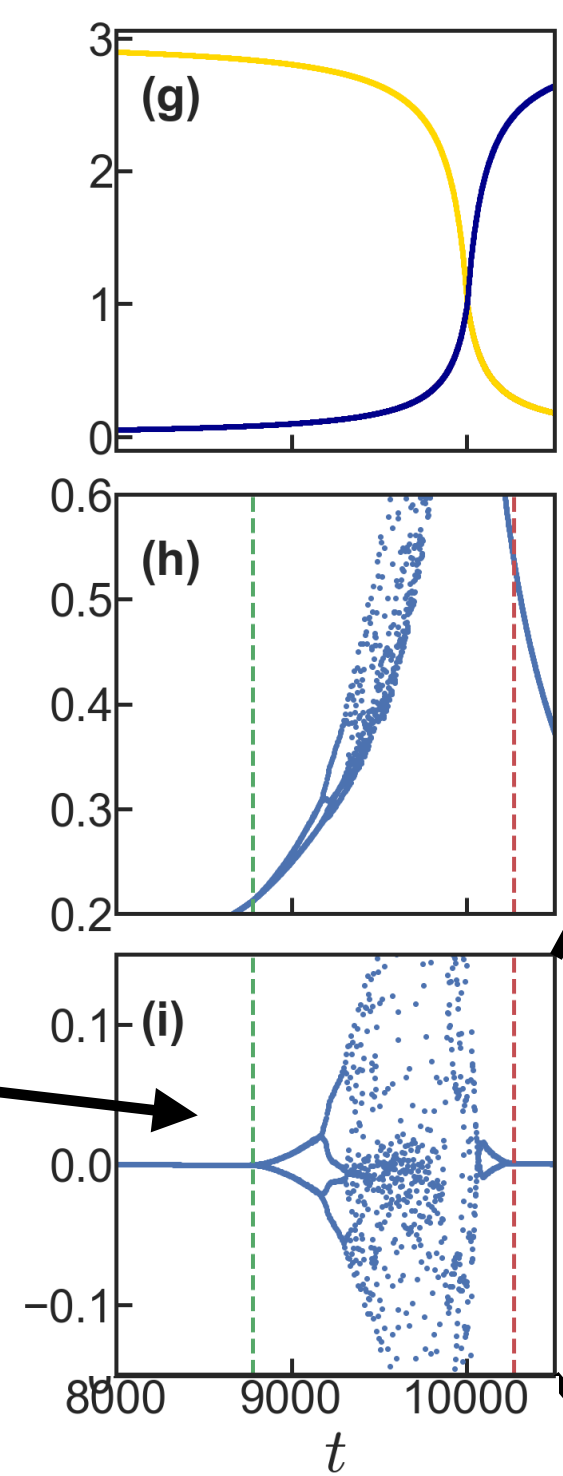
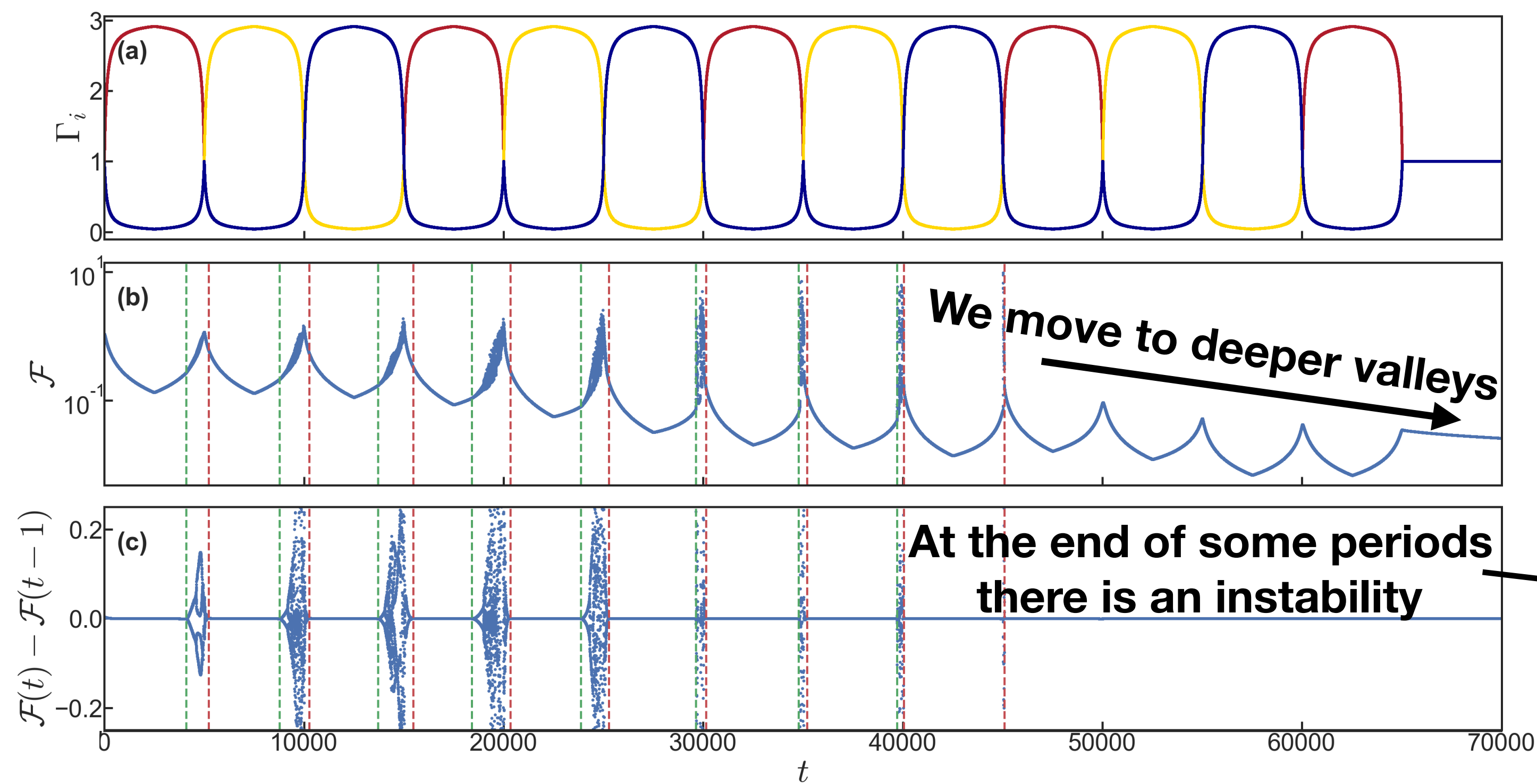
$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$

Coupled dynamics: the system is descending a landscape that is oscillating...

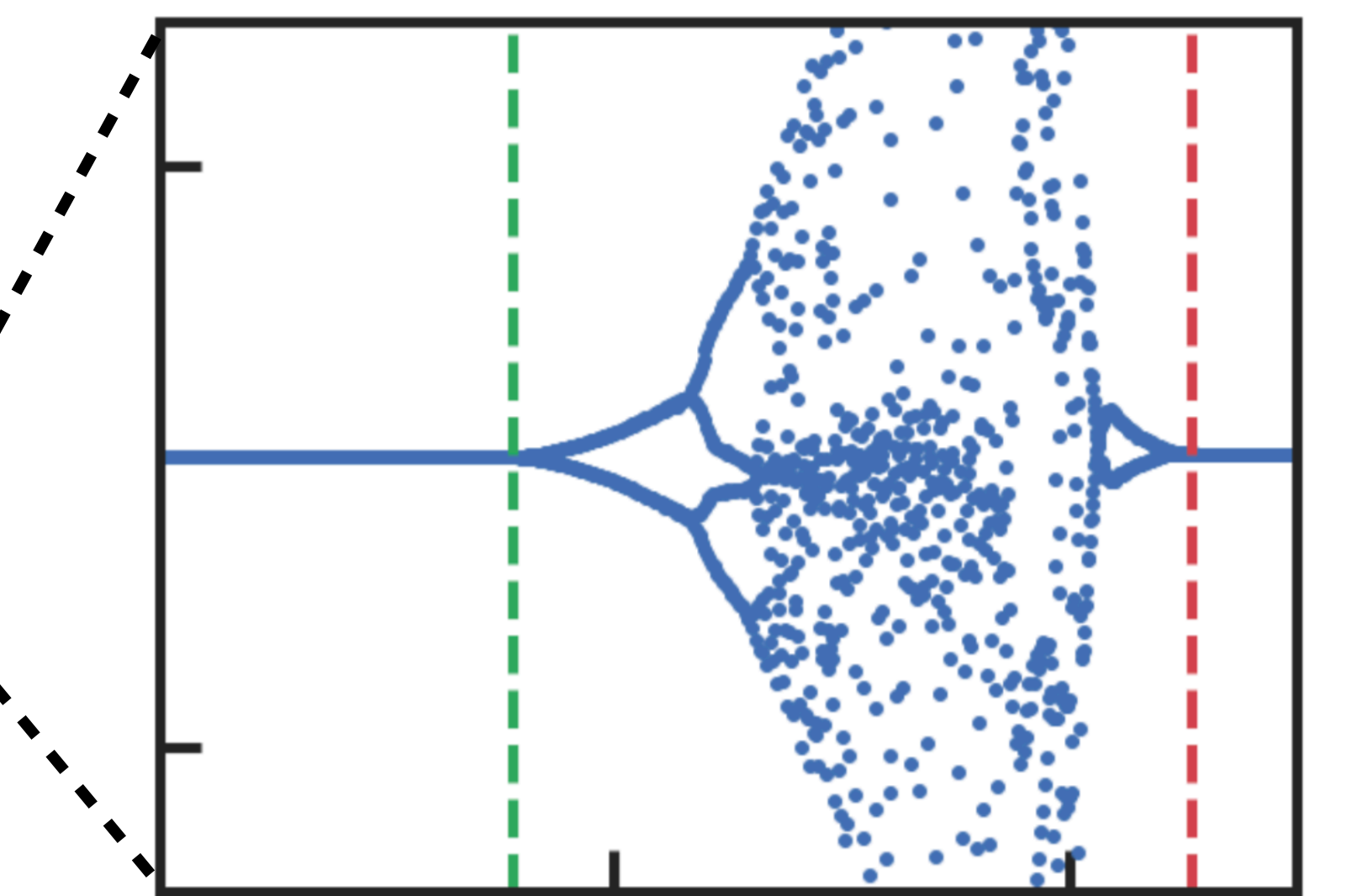


$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$

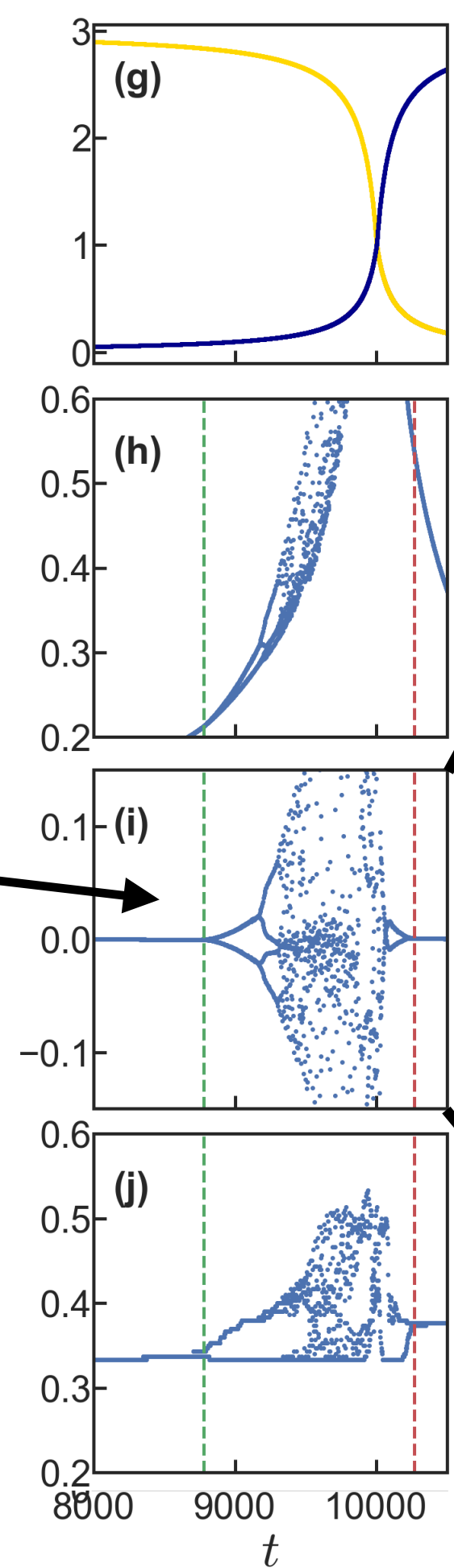
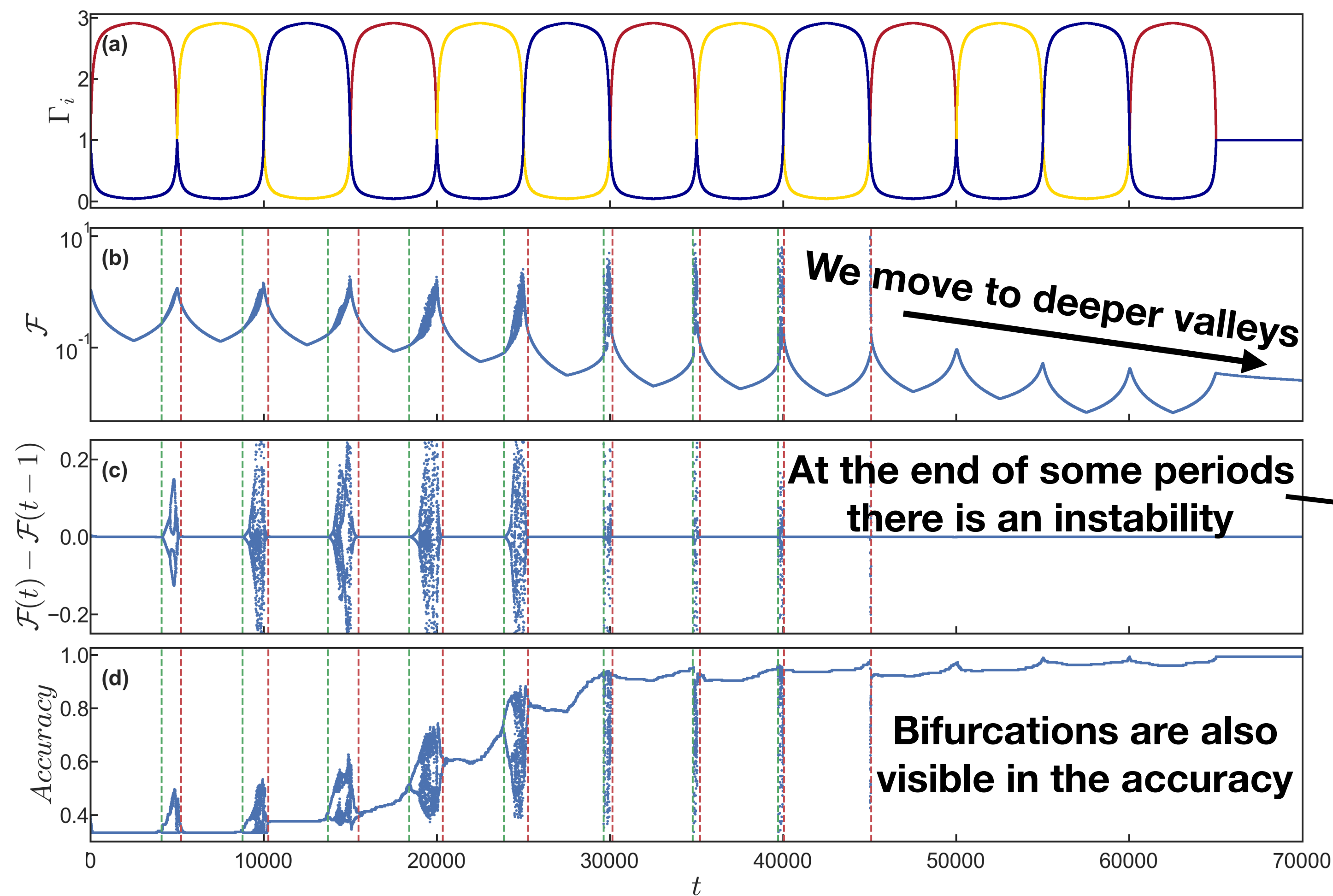
Coupled dynamics: the system is descending a landscape that is oscillating...



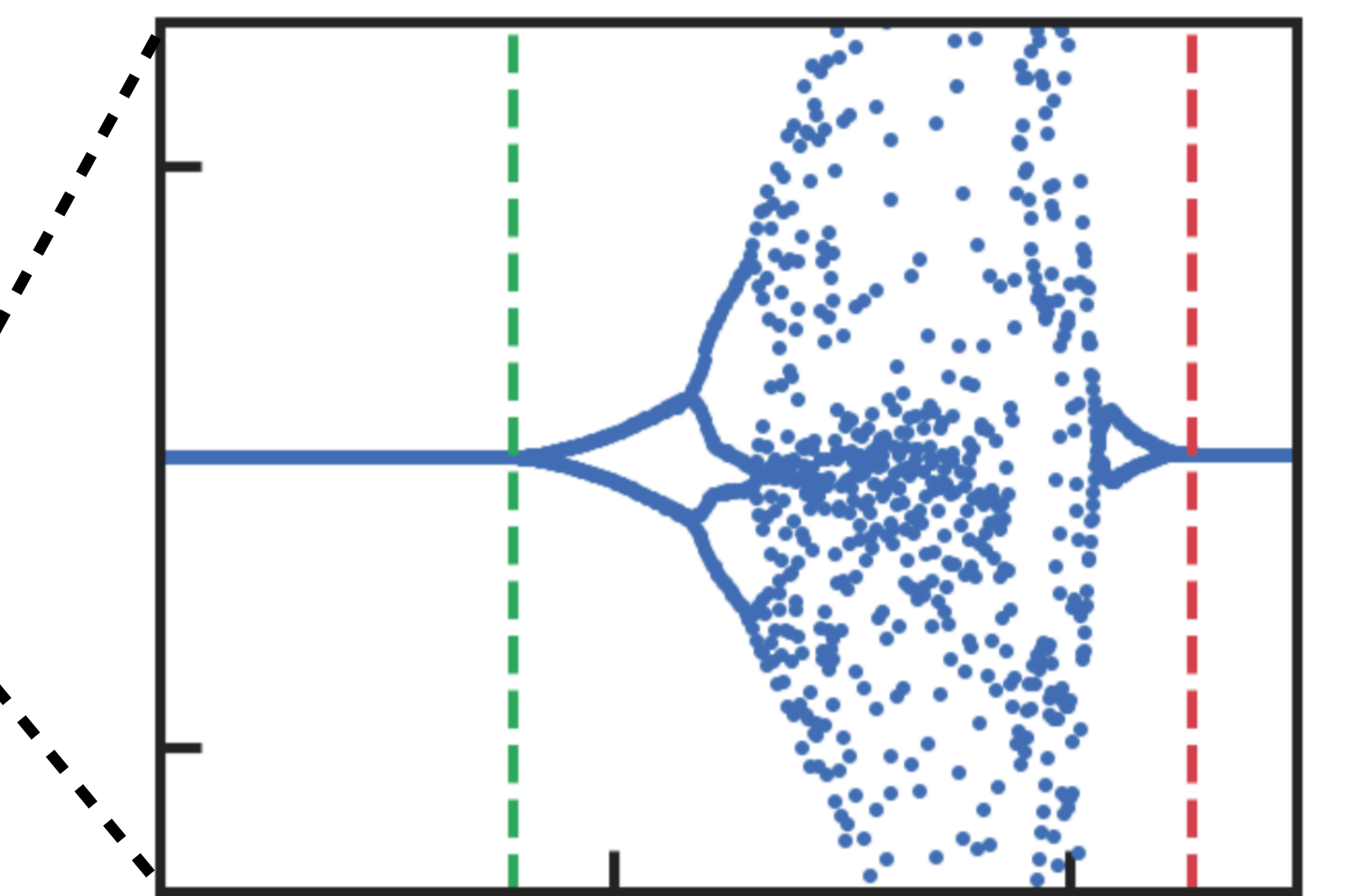
$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$



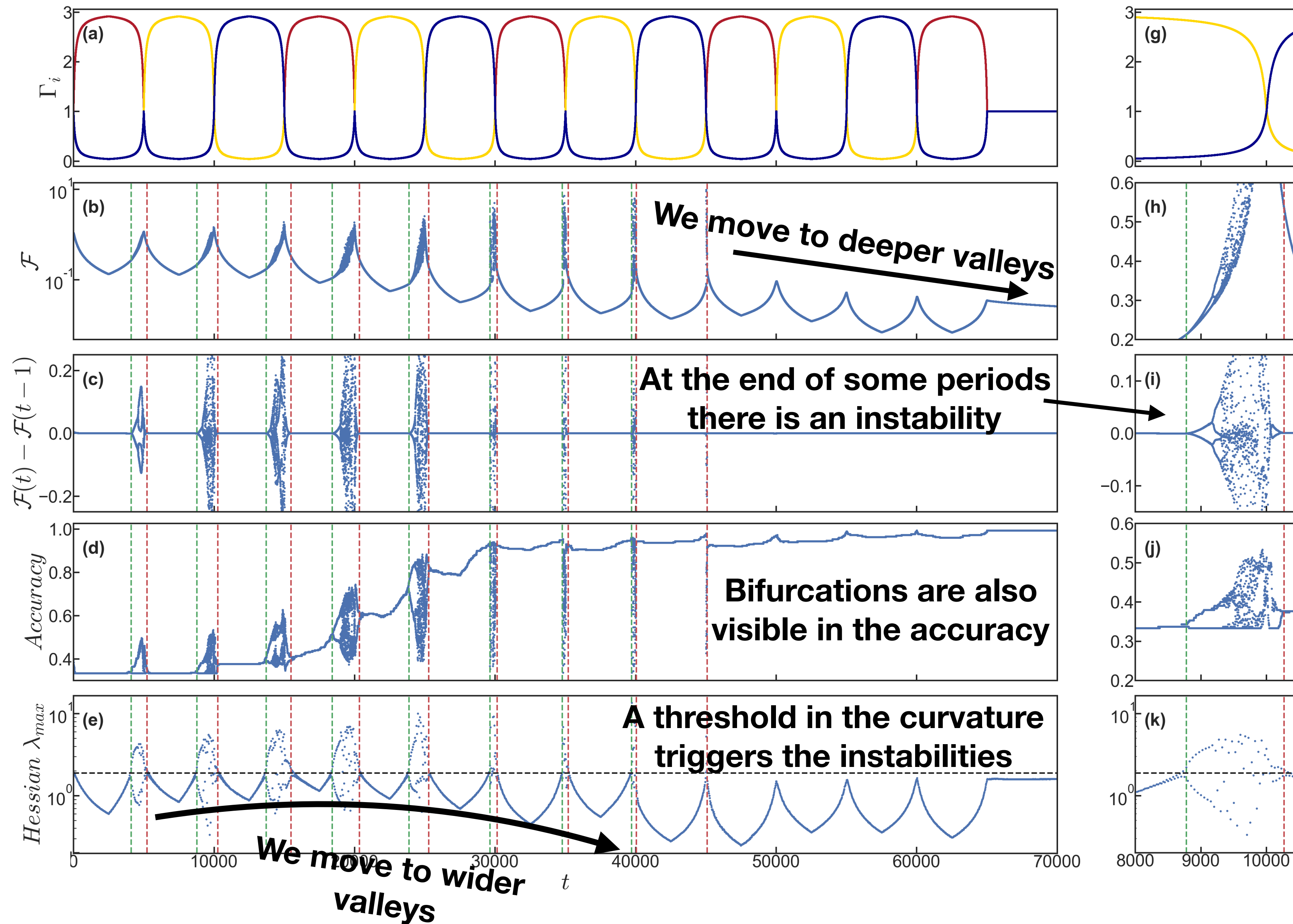
Coupled dynamics: the system is descending a landscape that is oscillating...



$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$



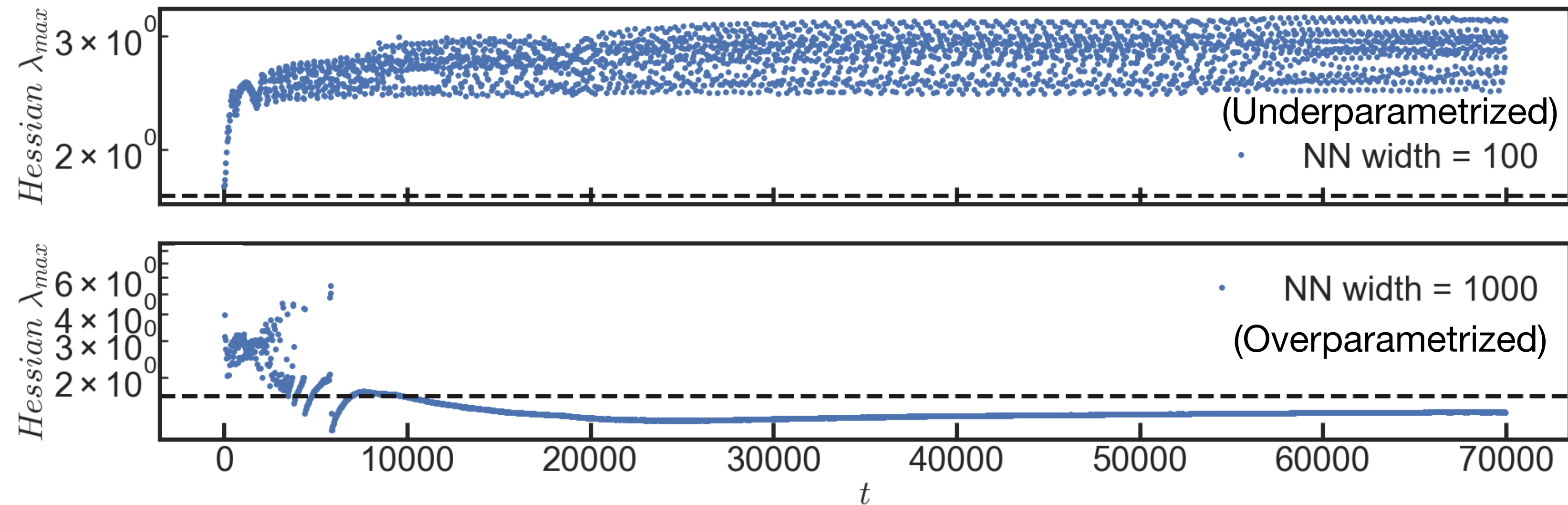
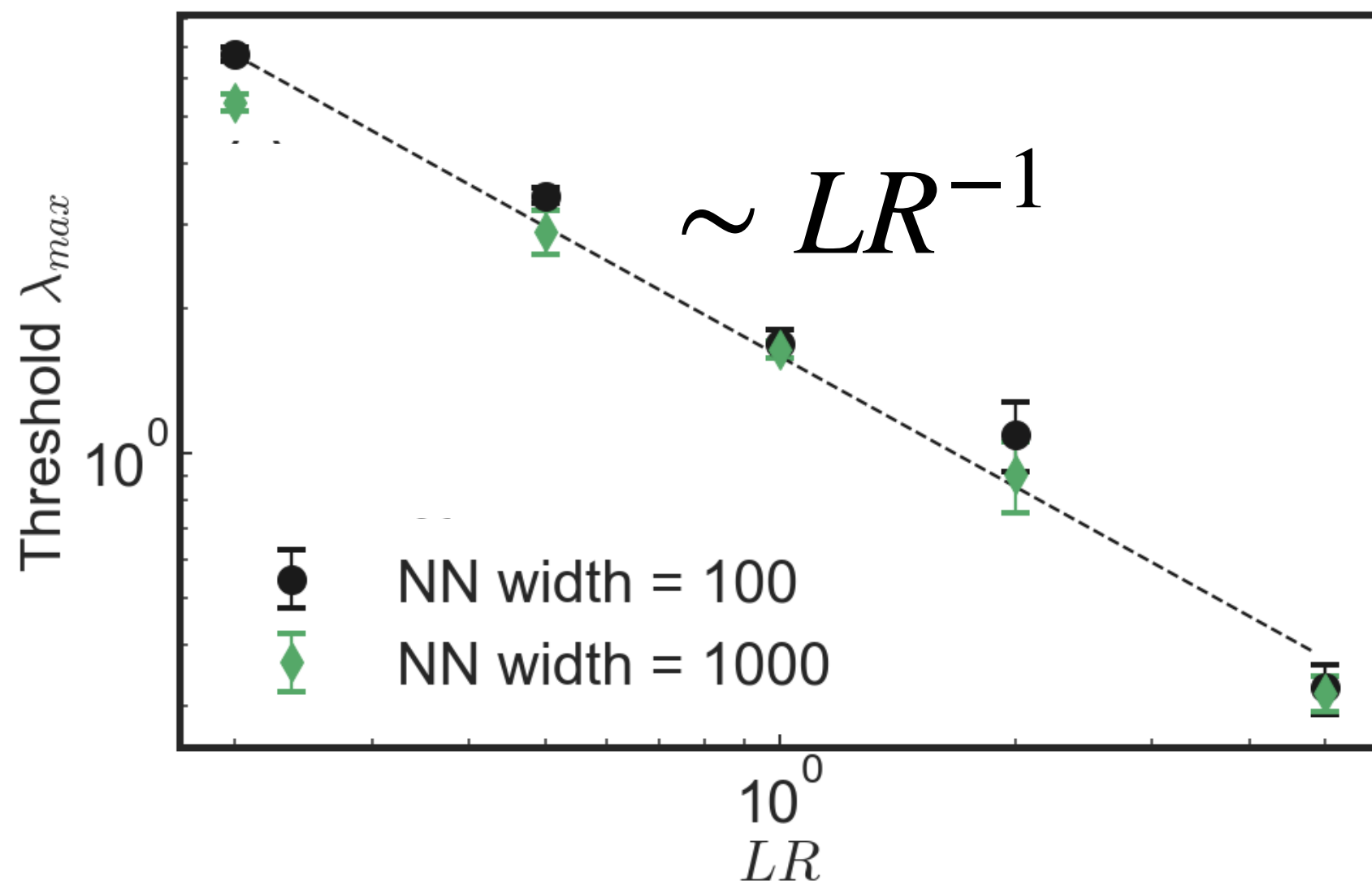
Coupled dynamics: the system is descending a landscape that is oscillating...



$$\mathcal{F} = \sum_{j \in P} \Gamma_{c_j}(t) \left(-\log \left(\frac{e^{f(x_j, y_j, \mathbf{W})_{c_j}}}{\sum_i e^{f(x_j, y_j, \mathbf{W})_i}} \right) \right)$$

The dynamical loss function leads to a landscape with valleys that oscillate. Valleys alternatively become deeper and wider and higher and narrower. This "peristaltic" movement pushes the system towards better minima.

The threshold depends on the learning rate but it does not depend on the width of the NN:



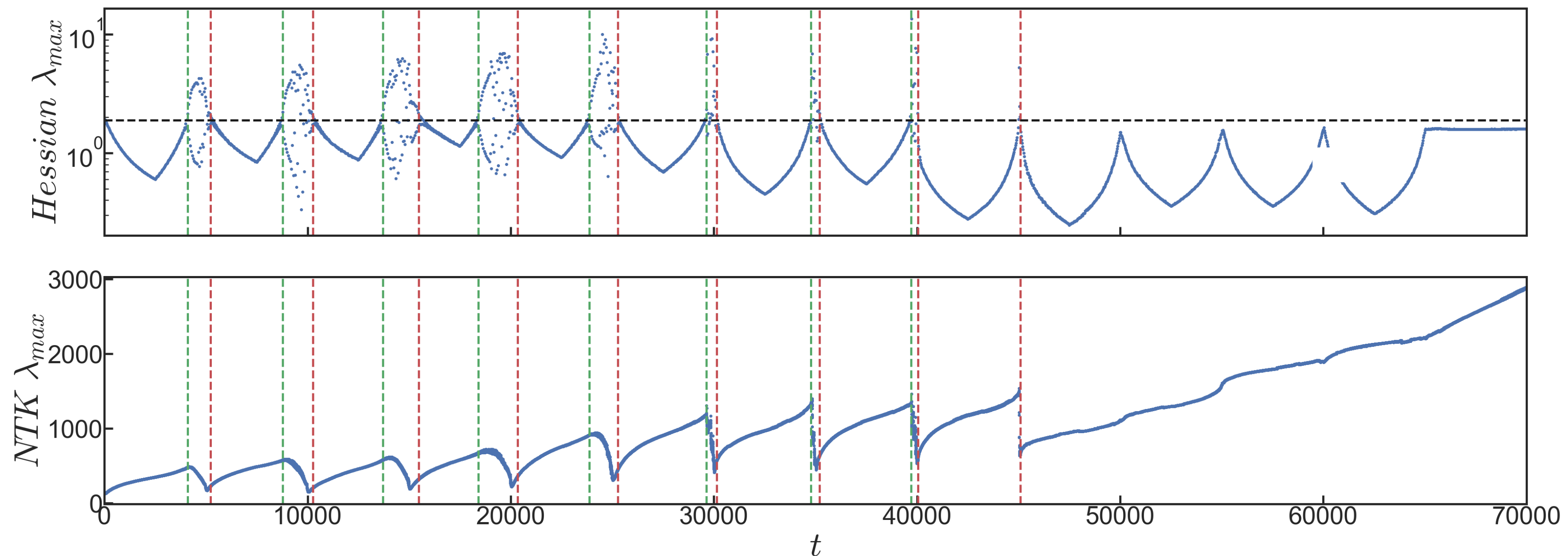
The threshold is computed with the dynamical loss function but it explains why the model was (not) able to learn with the standard static loss function!

$$\mathcal{F}(\vec{x}) \sim \mathcal{F}(\vec{a}) + \nabla \mathcal{F}(\vec{a})(\vec{x} - \vec{a}) + \frac{1}{2}(\vec{x} - \vec{a})^T H \mathcal{F}(\vec{a})(\vec{x} - \vec{a}), \quad (\vec{x} - \vec{a}) \propto LR$$

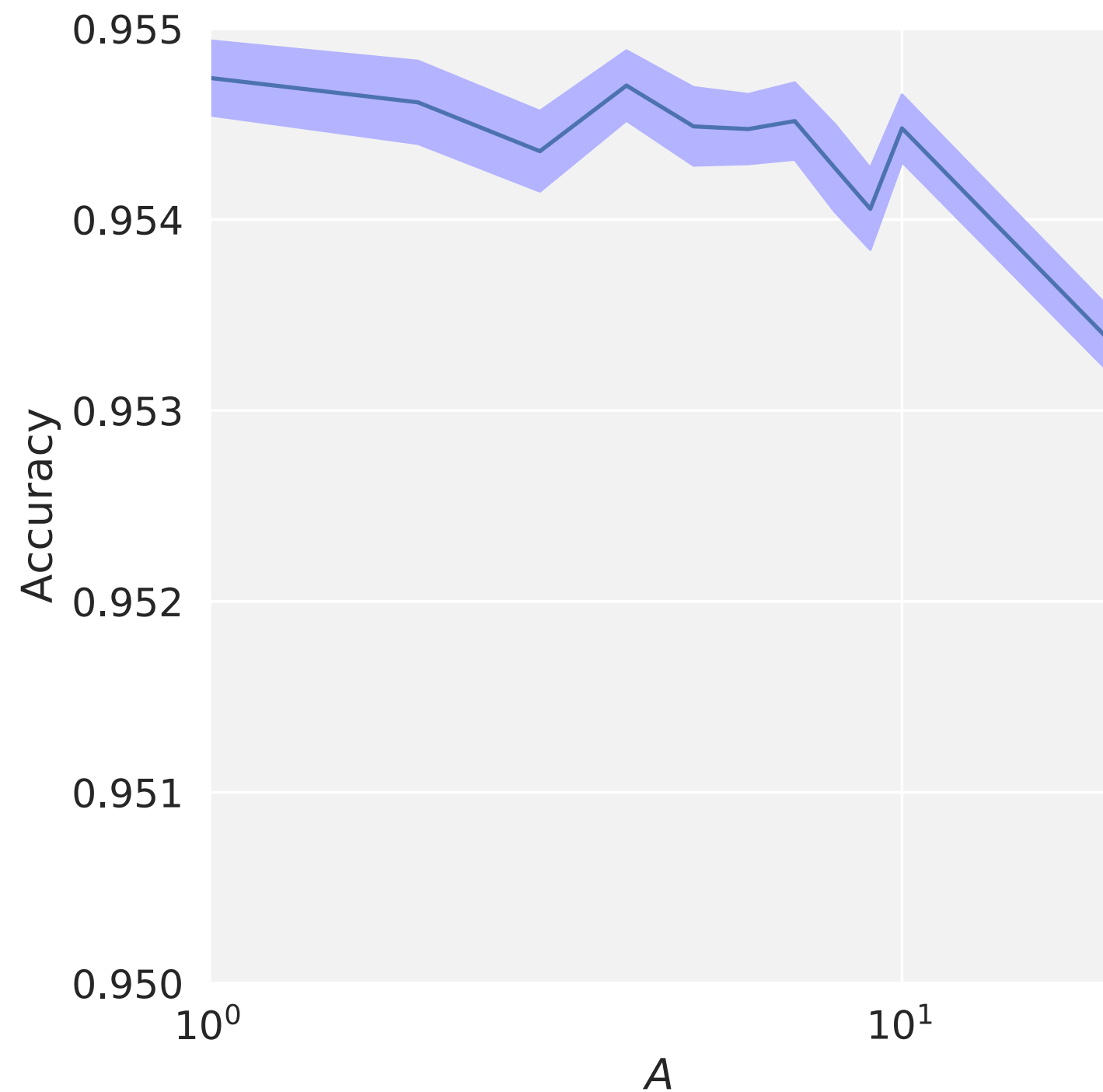
GD breaks down when the first and second order terms are proportional \rightarrow Hessian $\lambda_{max} \propto LR^{-1}$

Bonus slide

Bifurcations can also be understood in terms of the NTK. Largest eigenvalue of the NTK drops during instabilities:



Test accuracy of a Wide Residual Network and CIFAR10. We did not see an improvement over our limited set of experiments.



Why was the dynamical loss unhelpful in this case?

- 1) Interaction of the dynamical loss with batch normalization and regularization terms?
- 2) The network is already well-conditioned and so the oscillations may not lead to further improvements?
- 3) Do we need to retune other hyperparameters?
- 4) Can we use a variable learning rate that takes advantage of the changing curvature of the dynamical loss?

We can define dynamical loss functions for deep neural networks taking advantage of the different classes in the dataset

Dynamical loss functions can improve training in the underparametrized regime and generalization in the overparametrized regime

We understand the complex dynamics and its instabilities in terms of the curvature of the landscape

**Thank
you!**



HR EXCELLENCE IN RESEARCH



uc3m

Universidad
Carlos III
de Madrid



BURROUGHS
WELLCOME
FUND 

XSEDE

Extreme Science and Engineering
Discovery Environment

SIMONS
FOUNDATION