Deep Learning course

Session 7 – Regularization

E. Francisco Roman-Rangel
edgar.roman@alumni.epfl.ch

CInC-UAEM. Cuernavaca, Mexico. September 22nd, 2018.

Outline

Training and validation sets

- Tune parameters (Ω) using training set.
- Tune hyperparameters using validation set:
  Reduce overfitting.
- **Condition:** both sets must come from the same distribution.

Split

- Training size: 80%.
- Validation size: 20%.

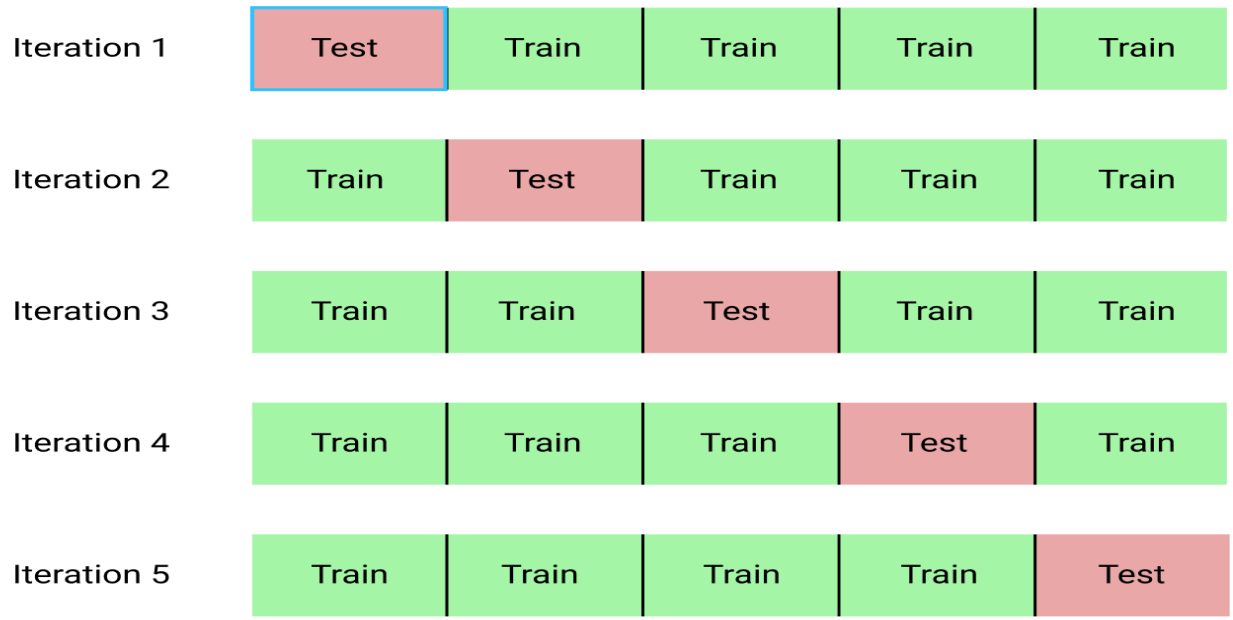
Test set

Additional set for evaluating the model on real data.

- Split: 70%, 20%, 10%.

k-folds cross validation

- It might be difficult to get enough data to populate all three sets.
- Divide training data in k folds (80%-20%), and train k times.
- "Evaluate the best model on the test set": (report average).



Leave-one-out

"Full cross-validation"

- Validate only on a single example.
- Repeat training as many times as there are training examples.
- Choose best model for production (evaluate on the test set).
- Report average.

Different underlying distribution

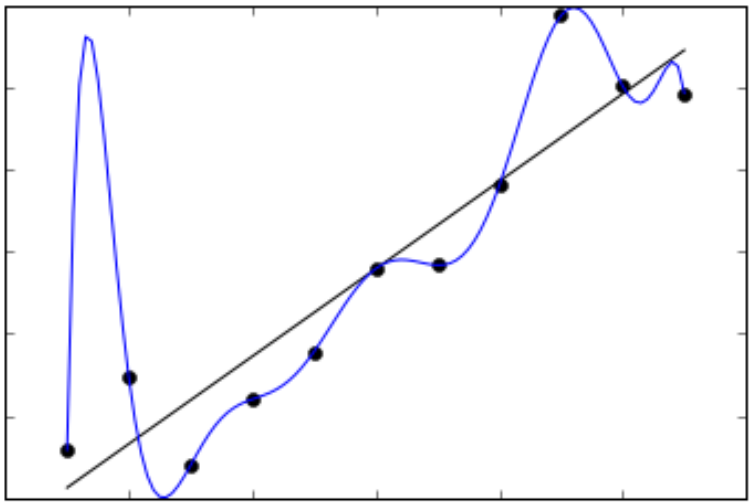
What if it happens that the test set looks different from the training and validations sets?

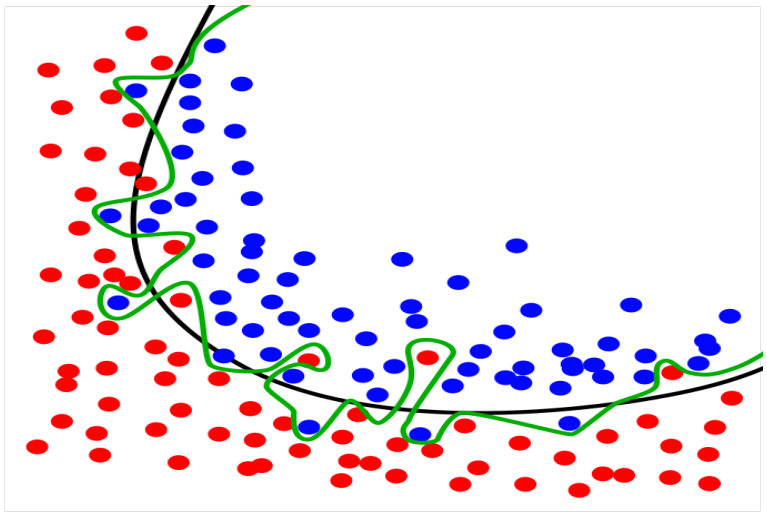
- Report average on training and test sets.
- That's the best we can do (expect).
- We cannot know much more.

Outline

Overfitting

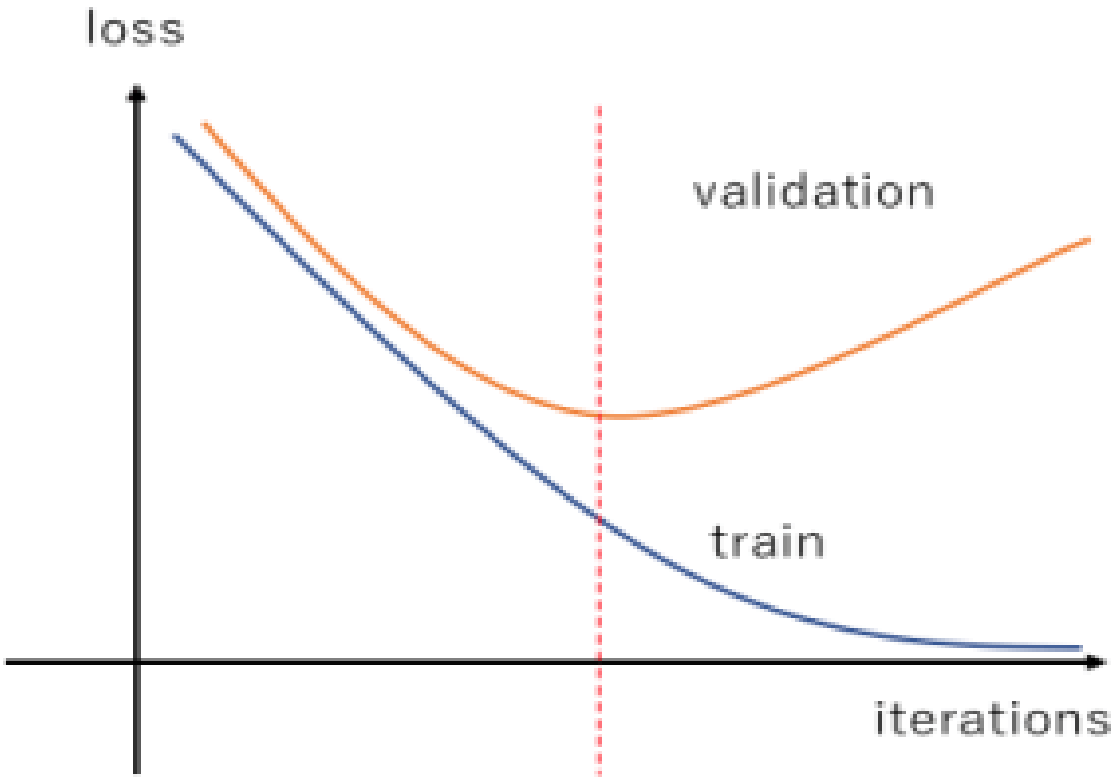
Having a model that performs well on a training set, but fails when challenged by unseen data samples.





- Approach for solution: give up a bit of training performance in favor of a better generalization.

Overfitting



- The model learns the expected output of the training set rather than its underlaying general distribution (even noise).
- Memorizing is not learning.

Over- and under-fitting

Memorizing training data vs not learning at all.
(not even learning to model the data).

High bias

Both training and validation error are high: **underfitting**.
Parameters are biased and do not fit well the data.

High variance

Training error is low whereas validation error is high: **overfitting**.
There are variations between the underlaying structure of the training and validation sets.

Preventing overfitting: regularizers

- Train longer.
- Gather more data.
- Decrease the number of features.
- L2, L1, (L0).
- Dropout.
- Batch normalization.

Preventing underfitting

- More capacity.
- More training epochs.
- More training data.
- More features.

Example (MNIST)

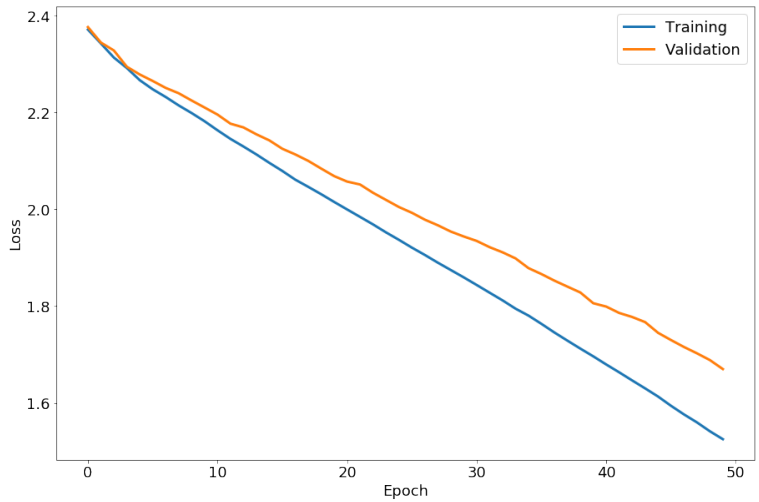
- 10 classes.
- Training on 60K examples.
- 784-D.
- Validation $= 20\%$.
- batch size $= 32$.

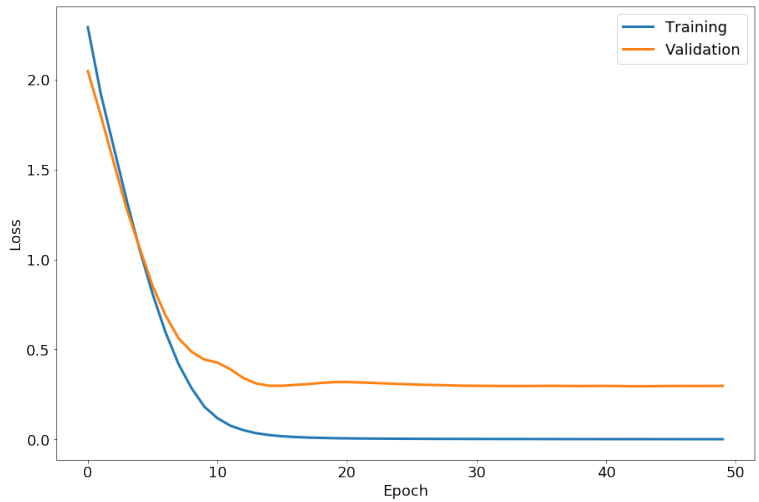
Example (MNIST): size of training data

100 vs 60K training examples.

Layer (type)	Output Shape	Param #
dense_23 (Dense)	(None, 200)	157000
dense_24 (Dense)	(None, 100)	20100
dense_25 (Dense)	(None, 50)	5050
dense_26 (Dense)	(None, 10)	510

Total params: 182,660
Trainable params: 182,660
Non-trainable params: 0

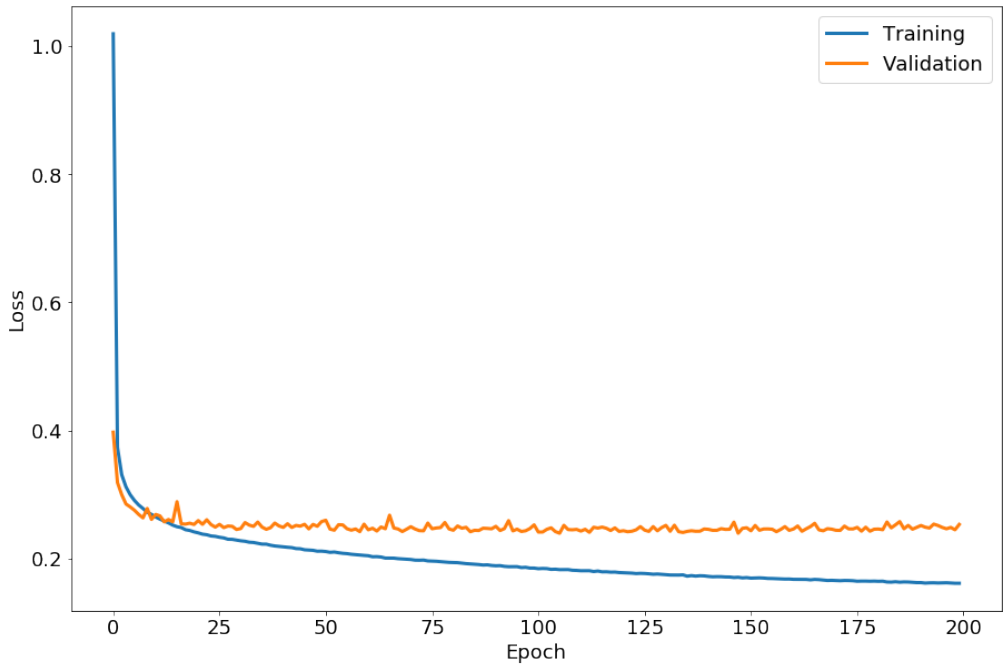




Example (MNIST): low capacity

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 10)	7850
dense_2 (Dense)	(None, 10)	110

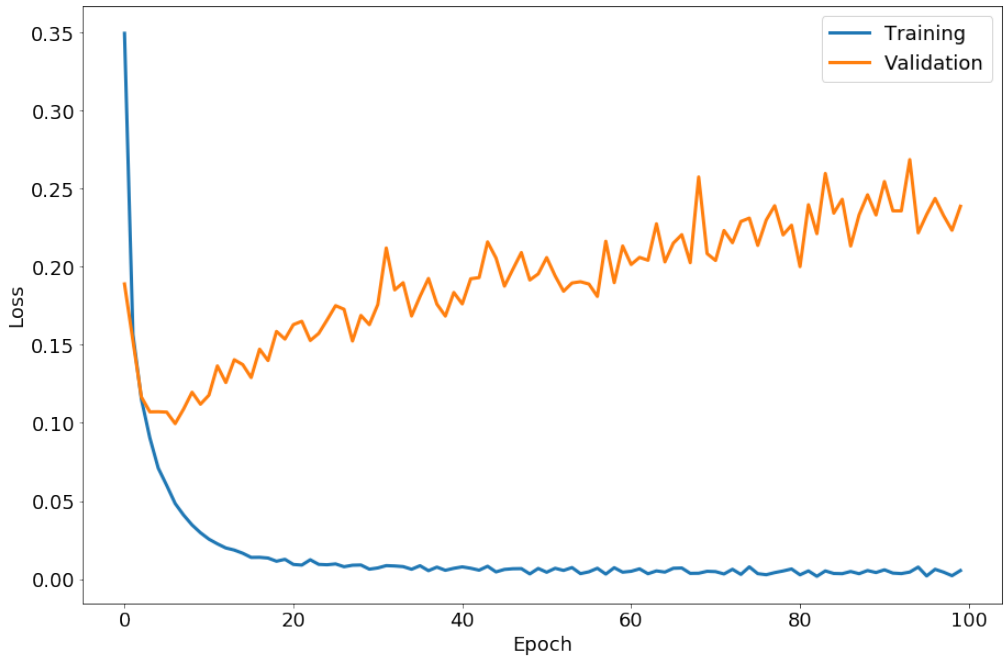
Total params: 7,960
Trainable params: 7,960
Non-trainable params: 0



Example (MNIST): enough capacity

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 100)	78500
dense_2 (Dense)	(None, 20)	2020
dense_3 (Dense)	(None, 10)	210

Total params: 80,730
Trainable params: 80,730
Non-trainable params: 0

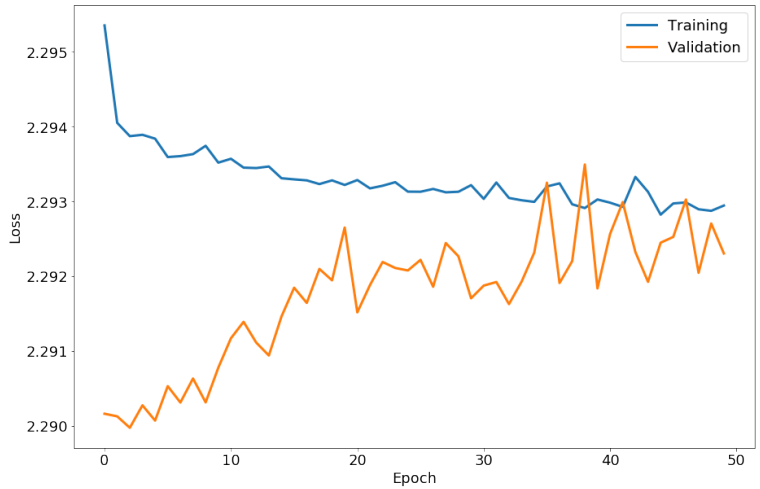


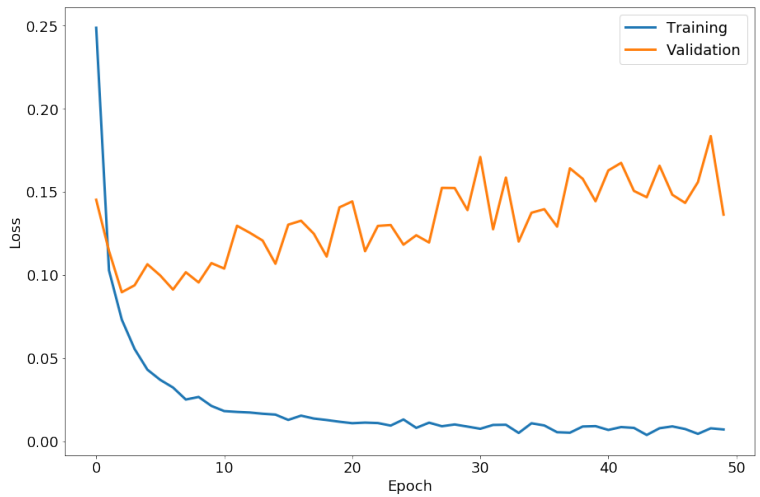
Example (MNIST): number of features

50-D vs 784-D.

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 200)	157000
dense_6 (Dense)	(None, 100)	20100
dense_7 (Dense)	(None, 50)	5050
dense_8 (Dense)	(None, 10)	510

Total params: 182,660
Trainable params: 182,660
Non-trainable params: 0





Goal

	Low *Training* Error	High *Training* Error
Low *Testing* Error	The model is learning!	Probably some error in your code. Or you've created a *psychic* AI.
High *Testing* Error	OVERFITTING	The model is not learning.

Outline

Regularization

- Limit model capacity.
- Adding a norm penalty to parameters.

$$\mathcal{L}_T(\mathbf{X}, \mathbf{y}; \mathbf{\Omega}) = \mathcal{L}(\mathbf{X}, \mathbf{y}; \mathbf{\Omega}) + \alpha \mathcal{P}(\mathbf{\Omega})$$

where, $\mathcal{L}(\cdot)$ is the standard loss function, $\mathcal{P}(\cdot)$ is the *penalty* function, and α is a hyperparameter weighting the relative contribution of the norm penalty term with respect to $\mathcal{L}$.

Parameter values are forced to remain small, thus keeping the function as a polynomial of low degree.

- Bias term is not subject of regularization, i.e., $\mathbf{\Omega} = \mathbf{\Omega}_{[1:N]}$.

L2 (weight decay)

Ridge regression or Tikhonov regularization.

$$\mathcal{P}(\mathbf{\Omega}) = \frac{1}{2}\|\mathbf{\Omega}\|_2^2$$

Derivative:

$$\mathbf{\Omega}$$

L1

Lasso regression.

$$\mathcal{P}(\mathbf{\Omega}) = \|\mathbf{\Omega}\|_1 = \sum_i |\omega_i|$$

Derivative:

$$\text{sign}(\mathbf{\Omega})$$

Besides keep small values for the weights, it also induces sparsity.

Sparsity

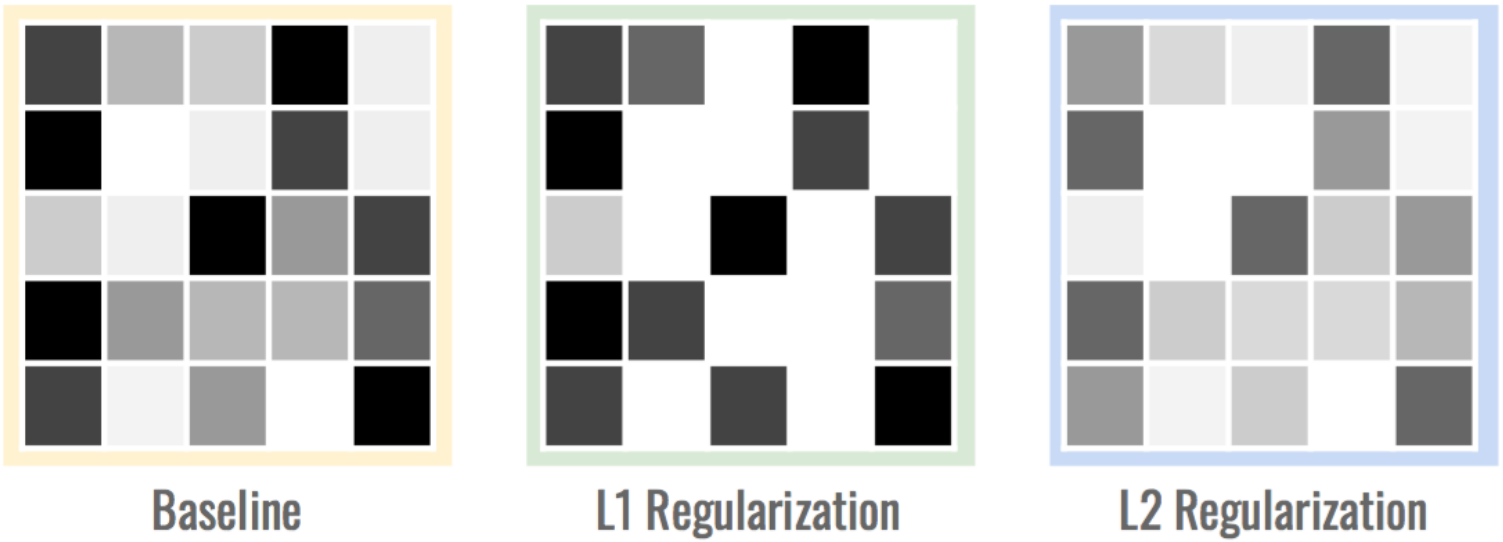


Image from:
https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

L2L1

Elastic regression (ridge + lasso).

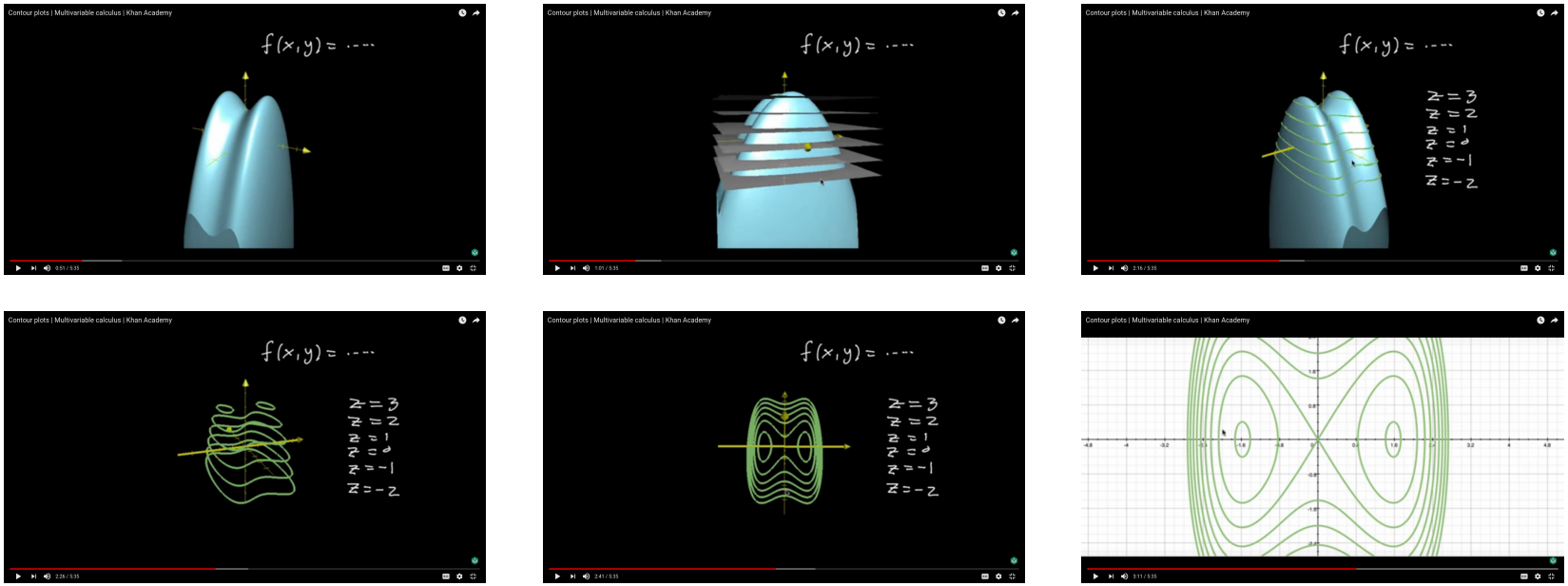
$$\mathcal{P}(\mathbf{\Omega}) = \alpha_R \|\mathbf{\Omega}\|_2^2 + \alpha_L \|\mathbf{\Omega}\|_1$$

Derivative:

$$\mathbf{\Omega} + \text{sign}(\mathbf{\Omega})$$

Contour plots

(Background)



Images from Video: Contour plots - Multivariable calculus - Khan Academy

https://www.youtube.com/watch?v=WsZj5Rb6do8

Normal balls

Keep parameter values at the intersection between the Loss space and the Penalty space.

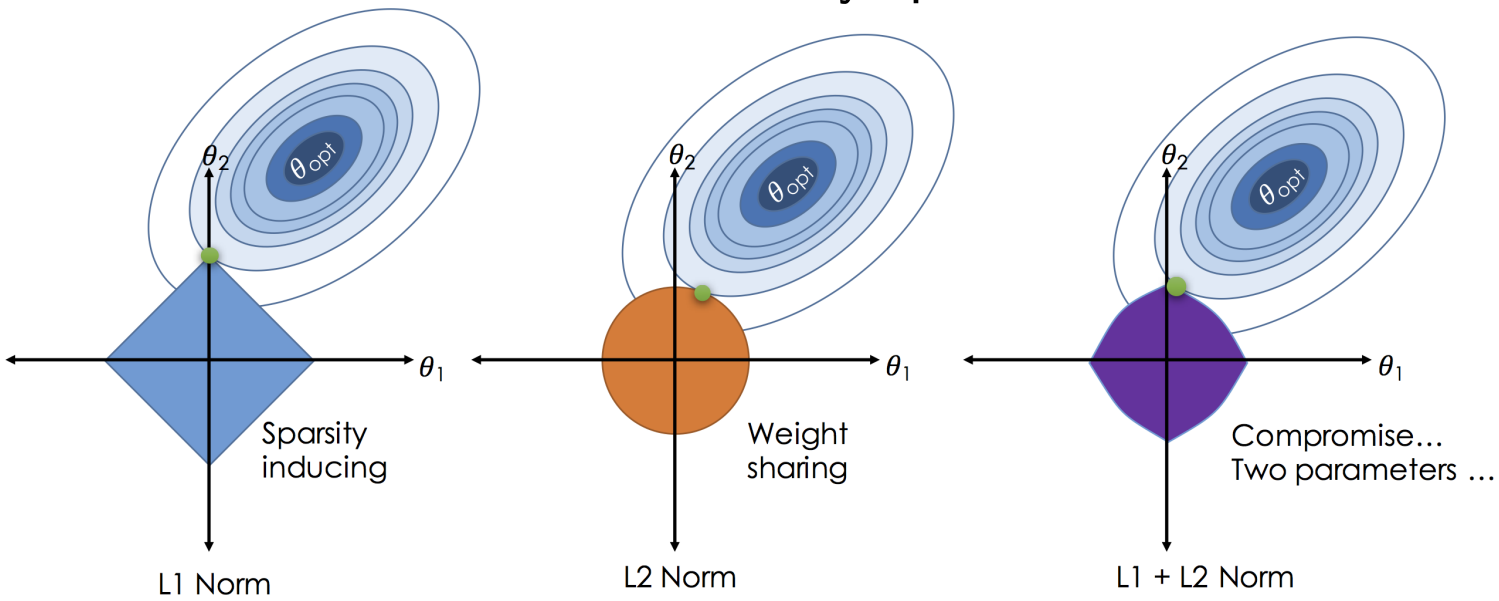


Image from:

http://www.ds100.org/sp17/assets/notebooks/linear_regression/Regularization.html

We might not reach the minimum possible loss, but remain close enough while still giving room for generalization.

L1 contours

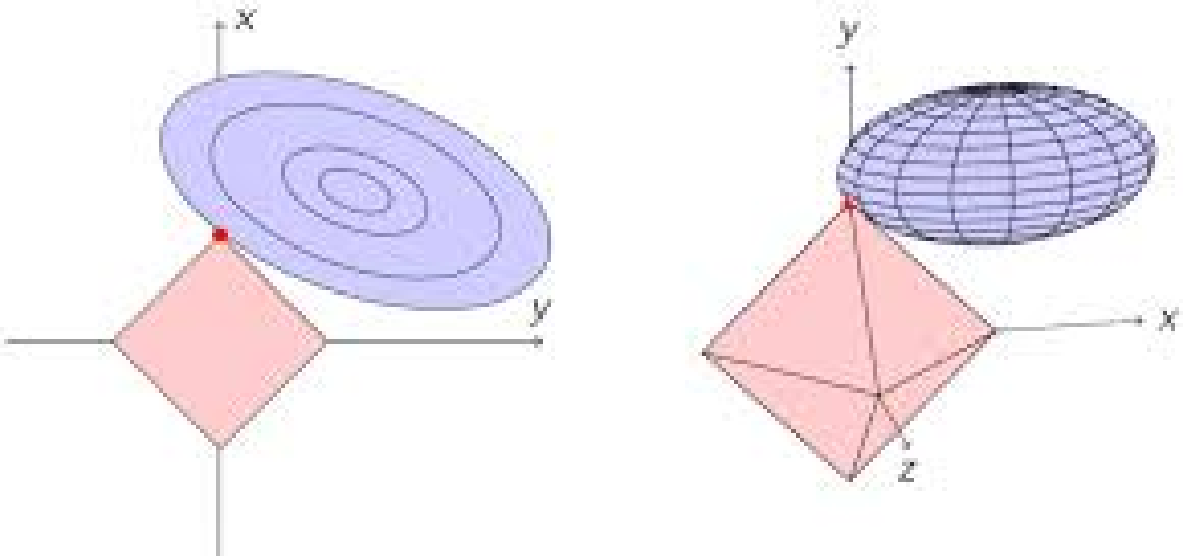


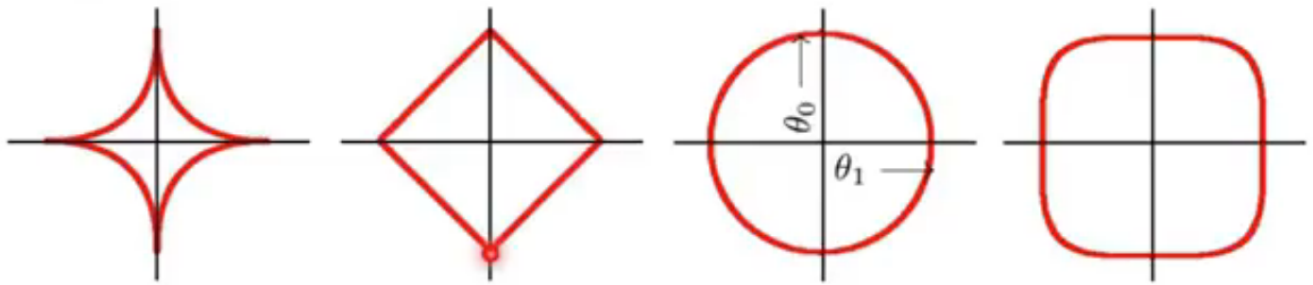
Image from Oxford ML course:
http://www.robots.ox.ac.uk/ az/lectures/ml/lect4.pdf

Other contours

Different regularization functions

- More generally, for the L_p regularizer: $\left(\sum_i |\theta_i|^p \right)^{\frac{1}{p}}$

Isosurfaces: $\|\theta\|_p = \text{constant}$

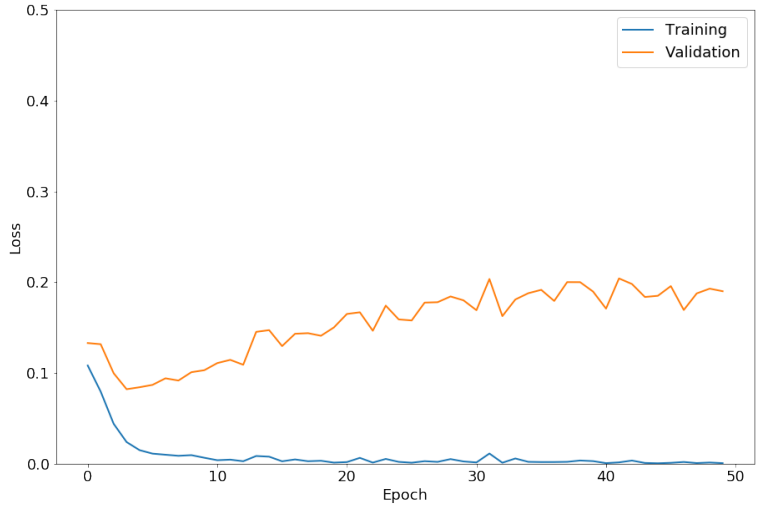


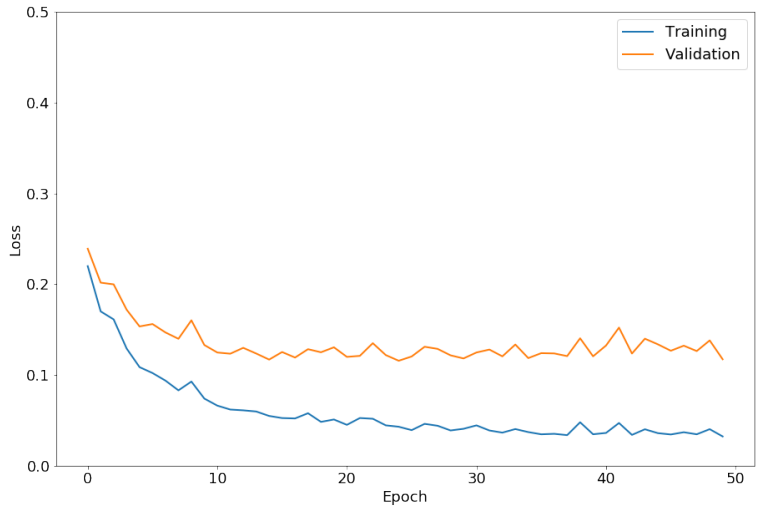
L_0 = limit as $p \to 0$: "number of nonzero weights", a natural notion of complexity

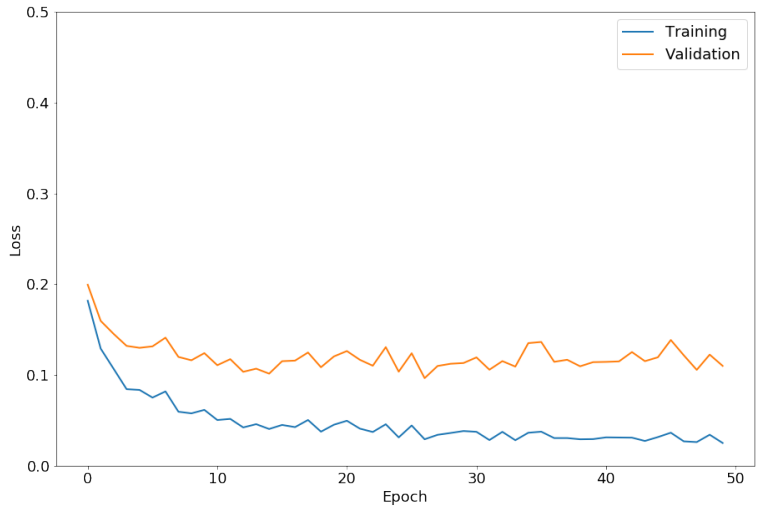
Image from Alexander Ihler youtube channel:
https://www.youtube.com/watch?v=sO4ZirJh9ds

Examples

No regularizer vs l1 vs l2

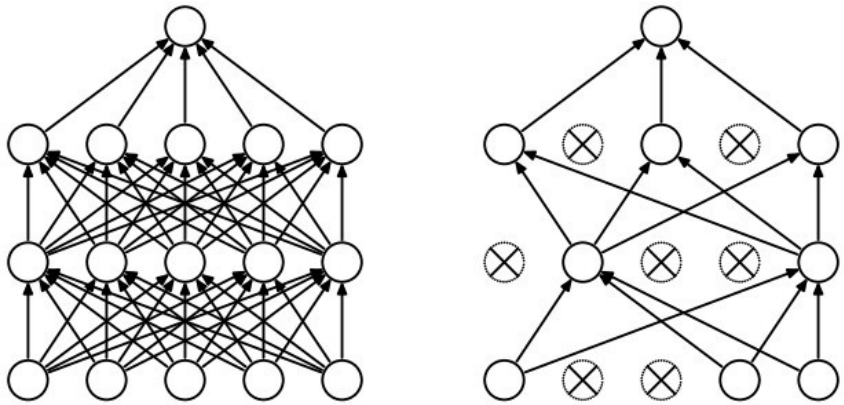






Dropout

- Add stochasticity.
- During training, limit the capacity of the model at random.
- Deactivate neurons (*dropout*) or weights (*dropconnect*).
- Forces the network to become redundant.
- Also helps to make the model robust against variations.



(a) Standard Neural Net (b) After applying dropout.

Image from Srivastava et al., 2014, JMLR.

Batch normalization

- Normalization is often applied to input layer (accelerates learning).
- We can do the same to hidden layers.
- Add noise and learn to be robust against it.
- Induces independence between layers.

Ioffe & Szegedy, 2015. ICML. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift".

Batch normalization

Compute the batch mean and variance:

$$\mu_B = \frac{1}{M}\sum_{m=1}^{M} x_m, \qquad \sigma_B^2 = \frac{1}{M}\sum_{m=1}^{M}(x_m - \mu_B)^2.$$

Normalize and fit to next layer:

$$\hat{x}_i = \gamma \cdot \frac{x_m - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

This might be applied to different layers, i.e., $x_m^l \leftarrow a_m^l$.

Suggested readings

- Goodfellow, DL Book. Regularization chap.
- https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42
- https://www.youtube.com/watch?v=sO4ZirJh9ds
- https://www.youtube.com/watch?v=WsZj5Rb6do8
- Hinton et al., 2012. CVPR. "Improving neural networks by preventing co-adaptation of feature detectors".
- Srivastava et al., 2014. JMLR. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting".

Thank you.

Q&A