Machine Learning: Algorithms and Applications

Philip O. Ogunbona

Advanced Multimedia Research Lab
University of Wollongong

Artificial Neural Networks and Deep Learning: Regularization
Autumn 2023

Outline

Regularization for Deep Learning: A Taxonomy

Jan Kukačka, Vladimir Golkov, and Daniel Cremers
{jan.kukacka, vladimir.golkov, cremers}@tum.de
Computer Vision Group
Department of Informatics
Technical University of Munich

Machine learning tasks - examples

- Tasks are described in terms of how the machine learning system should process the example
- Examples are collection of *features* quantitatively measured from some object or event that machine learning system will process - $\boldsymbol{x} \in \mathbb{R}^n$
- Classification
  - Required to specify, to which of k categories some input belongs
  - A function of the following form is to be learned from data:

$$f : \mathbb{R}^n \rightarrow \{1, \ldots, k\} \tag{1}$$

  - $y = f(\boldsymbol{x})$ assigns feature $\boldsymbol{x}$ to category identified by y
  - Function could also output a probability distribution over classes (categories)
- Regression
  - Required to predict a numerical value given some input
  - A function of the following form is to be learned from data:

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \tag{2}$$

  - Contrast with classification where output is categorical data type

Machine learning tasks - examples

- Transcription
  - Required to observe a relatively unstructured representation of some kind of data and transcribe it into discrete textual form
  - A function of the following form is to be learned from data:

$$f : \mathbb{R}^n \rightarrow \mathbb{A}^{k(m)} \tag{3}$$

where $\mathbb{A}$ is the set of some language alphabets (English, Yoruba, etc) and $k(m)$ is some variable number that indicates variable lengths of alphabets and depends on the application. Speech recognition is an example.

- Machine translation
  - Required to convert sequence of symbols (or alphabets) in some language to sequence of symbols (alphabets) in another language
  - A function of the following form is to be learned from data:

$$f : \mathbb{A}_x^{k(n)} \rightarrow \mathbb{A}_y^{k(m)} \tag{4}$$

where $\mathbb{A}_x$ is the set of some source language alphabets (English, Punjabi, Urdu, Mandarin, Yoruba, etc.) and $k(m)$ is some variable number that indicates variable lengths of alphabets, $\mathbb{A}_y$ is the set of some target language alphabets (German, French, Russian, etc.).

Machine learning tasks - examples

1. The task examples can also be regarded as the learning of concepts
2. We do not know in a compact analytical form, what function can be used to achieve the mapping underlying the concept.
3. Imagine having a set of functions from which we can select (this is the Hypothesis space, $\mathcal{H}$).
4. The idea is to find $h \in \mathcal{H}$ that minimises the error in generalising our knowledge of the concept to unseen samples.
5. A neural network is a universal approximator and can be used as a "template" for the function to be learned.
6. A neural network is a parameterised function, f_w, whose parameters (i.e. weights and biases, etc.) can be learned from data (this is termed training the network).

$$f_w : x \mapsto y$$

$w \in W$ are the parameters (or weights.)

We need to learn the parameters of the network that create a "function" or "model" that generalizes knowledge of the concept.

Regularization

- Training a neural network entails finding the $w \in W$, usually written a w^*, from among the several possibilities, that minimises a loss function $\mathcal{L} : W \mapsto \mathbb{R}$;

$$w^* = \arg\min_w \mathcal{L}(w). \tag{5}$$

- The loss function ,$\mathcal{L}$, is written in general in the form of an expected risk:

$$\mathcal{L} = \mathbb{E}_{(x,t)\sim P}\left[E(f_w(x), t) + R(\cdots)\right] \tag{6}$$

  where E is the error function (or fidelity term) and R is the regularization term. Note that $E(x, t)$ is a function of (x, t), while $R()$ is a penalty on the model, designed to constrain the set from which we are selecting the function (network) parameters.

- Empirical risk $\hat{\mathcal{L}}$, is normally used as approximation for Equation (6) to obtain optimal weight:

$$w^* = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,t_i)\in\mathcal{D}} (E(f_w(x), t) + R(\cdots)) \tag{7}$$

Definition: Regularization (Kukačka et al. 2017)

Any supplementary technique aimed at making a model generalize better is a form of regularization.

Regularization

Elements responsible for learned weights, w^*

1. $\mathcal{D}$: The training set
2. f: The selected model family (i.e. hypothesis space $\mathcal{H}$)
3. E: The error function selected
4. $\mathcal{R}$: The regularization term
5. The optimization technique selected to find the optimum w

Regularization

REGULARIZATION VIA DATA

1. Transformation from given dataset $\mathcal{D}$, to a new and contextually appropriate dataset $\mathcal{D}_R$
   - Feature extraction (PCA, KPCA, Fourier transform, wavelet transform, autoencoder, word model, etc.)
   - Preprocessing
   - Feature space transformation
   - Distribution remodelling
2. Data generation (augmented dataset)
3. Both dataset transformation and data augmentation can be performed with stochastic (i.e. probabilistic) parameters.

Regularization

REGULARIZATION VIA DATA

- Transformation of dataset with stochastic parameter could be of the form :

$$\tau_\theta(x) = x + \theta, \quad \theta \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \tag{8}$$

- In general θ could be generated in three (3) ways:
  - deterministically (i.e fixed by some equation)
  - randomly according to some distribution
  - adaptively obtained through optimization (constrained or unconstrained or stochastic) and with a specific objective.
- The data transformation could be seen as:
  - representation-preserving (both feature space and distribution)
  - representation-modifying (possibly different distribution and feature space); map the original factors (features) to a space where learning is easier

Regularization

REGULARIZATION VIA DATA

- Transformation could happen in different data space (or format):
  - input space - apply transformation to data x
  - hidden-feature space - e.g. modifying output of layers
  - the target (label) could be modified during training
- The data transformation could happen during training or testing (training samples or test samples)

Regularization I

REGULARIZATION VIA NETWORK ARCHITECTURE

1. Merely selecting a network architecture is a form of regularization because it prescribes the type of function that is used for the approximation.
   - " ... using deep architectures does indeed express a useful prior over the space of functions the model learns" (Goodfellow et al. 2016, p.200).
2. As we have indicated earlier, the network is a function that maps input to output according to the number of layers and neurons; this already constrains the type of mapping.
3. The weight structure that the architecture imposes works with the selected optimization technique to dictate nature of the search space. Recall that during optimization we are traversing an error surface that is a function of the weights.
4. Certain invariances of the network are "baked in" or "hardwired" - e.g convolution. Table 3 of Kukačka et al. (2017) provides a list of methods on the network architecture and the assumptions they encode roughly.

Regularization I

REGULARIZATION VIA NETWORK ARCHITECTURE

Weight sharing: Reuses certain trainable parameter in several parts of the network.

- Model complexity is reduced
- Inherent encoding of prior knowledge about the underlying process of the data (convolution - shift-equivariance and locality of feature extraction).

Activation functions: The choice of activation function is important because of how it impacts the output of a neuron.

- ReLu is meant to ameliorate the vanishing gradient problem.
- Drpout and Maxout allows the approximation of the geometric mean of the model ensemble predictions at test time
- GeLu bridges combines stochastic regularizers, such as dropout, with non-linearities, i.e., activation functions

Regularization

REGULARIZATION VIA ERROR FUNCTION

1. Error functions usually depict the quality of approximation achieved by the network (mean-squared error, cross-entropy, etc.)
2. The Dice coefficient (when used as error function) could help combat data imbalance. It is commonly used as a metric in segmentation tasks but has been adapted to serve as a loss function:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{9}$$

where $|X|$ is number of correctly segmented pixels and $|Y|$ is the number of incorrectly segmented pixels.

In tensorflow it is implemented as:

```
def DiceLoss(targets, inputs, smooth=1e-6):
  #flatten label and prediction tensors
    inputs = K.flatten(inputs)
    targets = K.flatten(targets)

    intersection = K.sum(K.dot(targets, inputs))
    dice = (2*intersection + smooth) / (K.sum(targets) +
            K.sum(inputs) + smooth)
    return 1 - dice
```

Regularization

REGULARIZATION VIA REGULARIZATION TERM

1. An obvious way of achieving regularization is to add a regularizer to the loss function (i.e. as in Equation 6)
2. Usually this regularizer does not incorporate the target term; its a penalty on, for example, the weights. Recall the regularizer used in ridge regression (i.e. $||w||_2^2$) or the one used in LASSO (i.e. $||w||_1$)
3. Weight decay is another regularizer:

$$R(w) = \lambda \frac{1}{2} ||w||_2^2 \tag{10}$$

where λ controls the importance of the regularization in comparison to the consistency (fidelity) term. It promotes error reduction on training set.

Regularization

REGULARIZATION VIA REGULARIZATION TERM

1. "Smoothness" constrain allows the specifications of "landscape" of the learned mapping. The concept of "smoothness" implies that if two inputs x_1 and x_2 are close to each other, their resultant mapping $f_w(x_1)$ and $f_w(x_2)$ are also close. Formulated as:

$$R(f_w, x) = ||J_{f_w}(x)||_F^2 \tag{11}$$

where $||\cdot||_F$ denotes the Frobenius norm and $J_{f_w}(x)$ is the Jacobian of the neural network input-output mapping f_w for some fixed weights w.

2. This regularization penalises mappings with large derivatives (i.e. no big changes in the output of mapping); recall contractive autoencoder.

3. Note that Frobenius norm for a matrix $M \in \mathbb{R}^{n \times m}$ is defined as

$$||M||_F = \sqrt{\sum_{j=1}^{m} \sum_{k=1}^{n} |m_{jk}|^2} \tag{12}$$

i.e. the square root of the sum of the absolute values of its elements.

4. We have seen this regularization in lecture slides "Artificial Neural Networks and Deep Learning: An introduction (III)" Slide 10 - Contractive Autoencoder: $\Omega(\boldsymbol{h}) = \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2$

Regularization

REGULARIZATION VIA OPTIMIZATION

1. Stochastic gradient is the most frequently used optimization algorithm in deep neural network implementation.
2. In its basic form the weight updates are obtained as:

$$w_{t+1} = w_t - \eta_t \nabla_w \mathcal{L}(w_t, d_t) \tag{13}$$

   where $\nabla_w \mathcal{L}(w_t, d_t)$ is the gradient of the loss $\mathcal{L}$ evaluated on a mini-batch d_t of datat from the training set.
3. Recall the various extensions of the basic algorithm, Adam, RMSProp, AdaGrad etc., from lecture slides "Artificial Neural Networks and Deep Learning: An introduction (I)" Slides 48 -53
4. Optimization methods generally involve (i) initialization (ii) update and (iii) termination. Optimization-based regularization can be considered through these lenses.

Regularization

REGULARIZATION VIA OPTIMIZATION

1. Initial selection of model weights; sample from a distribution and keep variance of activations in all layers around one (1) to avoid vanishing or exploding activations and gradients
2. Prime the learning by pre-training network on data from a different task from the same domain
3. A good example: Transformer-based language models, such as OpenAI's GPT (Generative Pre-trained Transformer) series, utilize pre-training techniques (on large corpus of web documents or books). The resulting pre-trained models can be fine-tuned on specific NLP tasks like text classification, question-answering, or text generation, exhibiting strong language understanding capabilities.
4. Initialization can be with or without pre-training.
   - Without pre-training: use (i) random weight initialization, (ii) orthogonal weight matrices (iii) data-dependent weight initialization
   - With weight pre-training: (i) greedy layer-wise pre-training, (ii) Curriculum learning, (iii) spatial contrasting, (iv) subtask splitting

Regularization

REGULARIZATION VIA OPTIMIZATION

1. Update methods can be categorised as (i) update rules and (ii) gradient and weight filters
2. Update rules:
   - Momentum (see the lecture slides ANN&DL: An Introduction I slides 48-53)
   - Learning rate schedules
   - Online batch selection
   - SGD alternatives - L-BFGS, Hessian-free method, etc.
3. Gradient and weight filters:
   - Dropout (stochastically zeroing out activation)
   - Annealed Langevin noise (i.e. they added noise to gradient)
   - AnnealSGD (use annealing schedule for adjusting the learning rate; reduce learning rate according to (say) exponential decay, linear decay or step decay); leads to effective exploration of parameter space and better convergence

Regularization

REGULARIZATION VIA OPTIMIZATION

1. Termination of the optimization at the right moment may improve generalization by reducing discrepancy between minimizers of expected and empirical risk. A validation set could be used to determine stopping "moment".
   - Termination with validation set: (i) early stopping, (ii) choice of validation set.
   - Termination without validation set: (i) fixed number of iteration, (ii) optimized approximation algorithm

tensorflow - early stopping

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
# This callback will stop the training when there is no improvement in
# the loss for three consecutive epochs.
model = tf.keras.models.Sequential([tf.keras.layers.Dense(10)])
model.compile(tf.keras.optimizers.SGD(), loss='mse')
history = model.fit(np.arange(100).reshape(5, 20), np.zeros(5),
                    epochs=10, batch_size=1, callbacks=[callback],
                    verbose=0)
len(history.history['loss'])  # Only 4 epochs are run.
```

Bibliography

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press.

Kukačka, J., Golkov, V. & Cremers, D. (2017), Regularization for deep learning: A taxonomy, Technical Report : arXiv:1710.10686[cs.LG].
URL: *https://arxiv.org/abs/1710.10686v1*