Lecture 2: A dynamic view on universal interpolation and gradient descent

Christa Cuchiero based on joint lectures with M. Gambara, J. Teichmann and Hanna Wutte

Institute of Statistics and Mathematics
Vienna University of Economics and Business

Part I

Deep neural networks, generic universal interpolation, and controlled ODEs

based on joint work with M. Larsson and J. Teichmann

Goal

Analysis of deep feedforward neural networks from an optimal control theory point of view:

- deep neural networks as discretizations of certain controlled ODEs
- expressiveness and generic universal interpolation
- randomly generated generic expressiveness $\Rightarrow$ large numbers of parameters can be left untrained, and be chosen randomly

Definition of a deep feedfoward neural network

- Feedfoward neural networks are maps obtained by composing layers consisting of an affine map and a componentwise nonlinearity σ:

$$x(0) \xrightarrow{\ell_1} x(1) \xrightarrow{\ell_2} x(2) \longrightarrow \cdots x(t) \cdots \xrightarrow{\ell_n} x(n),$$

  where $x(t) \in \mathbb{R}^m$ and

$$\ell_t(x) = (\sigma(\langle A_{t,1}, x\rangle + b_{t,1}), \ldots, \sigma(\langle A_{t,m}, x\rangle + b_{t,m})).$$

  There is usually a (linear) readout map R such that

$$x(n) \longrightarrow Rx(n) = y.$$

Definition of a deep feedfoward neural network

- Feedfoward neural networks are maps obtained by composing layers consisting of an affine map and a componentwise nonlinearity σ:

$$x(0) \xrightarrow{\ell_1} x(1) \xrightarrow{\ell_2} x(2) \longrightarrow \cdots x(t) \cdots \xrightarrow{\ell_n} x(n),$$

  where $x(t) \in \mathbb{R}^m$ and

$$\ell_t(x) = (\sigma(\langle A_{t,1}, x\rangle + b_{t,1}), \ldots, \sigma(\langle A_{t,m}, x\rangle + b_{t,m})).$$

  There is usually a (linear) readout map R such that

$$x(n) \longrightarrow Rx(n) = y.$$

- For a given training data set $\{(x_i, y_i), i = 1, \ldots N\}$, supervised learning means selecting the parameters of $(A_t, b_t)_{t\in\{1,\ldots,n\}}$ and R such that

$$y_i \approx R \circ \ell_n \circ \cdots \circ \ell_1(x_i), \quad \forall i$$

Residual networks

- Define $V(x, \theta_t) := \ell_t(x) - x$, where θ_t collects all the parameters of ℓ_t.
- Then $x(t) = x(t-1) + V(x(t-1), \theta_t)$, which is sometimes called residual network (see, e.g. He et al. ('15)).
- This is nothing else than a discretization of an ODE

$$dX_t^x = V(X_t^x, \theta_t)dt, \quad X_0^x = x.$$

- The feedforward neural network is then modeled by

$$x \mapsto R(X_1^x)$$

  and can be interpreted as a network of continuous depth. The discrete parameter counting the layers is replaced by $t \in [0, 1]$.
- This perspective on neural networks can also be found in e.g. E('17); Chang et al.('17); Chen et al.('18); Grathwohl et al.('18); Dupont et al.('19), Liu and Markowich('19).

Supervised learning as a control problem

- For a given training data set $\{(x_i, y_i), i = 1, \dots N\}$ supervised learning now means selecting $V(\cdot, \theta_t)$ and R so that

$$y_i \approx R(X_1^{x_i}) \quad \forall i.$$

- We view this training task as a (deterministic) control problem: the N inputs x_i should be directed to their respective outputs y_i, all using the same vector fields.

Supervised learning as a control problem

- For a given training data set $\{(x_i, y_i), i = 1, \dots N\}$ supervised learning now means selecting $V(\cdot, \theta_t)$ and R so that

$$y_i \approx R(X_1^{x_i}) \quad \forall i.$$

- We view this training task as a (deterministic) control problem: the N inputs x_i should be directed to their respective outputs y_i, all using the same vector fields.
- Indeed, recognize $dX_t^x = V(X_t^x, \theta_t)dt$ as controlled ordinary differential equation (CODE) by supposing that

$$V(x, \theta) = u^1 V_1(x) + \quad + u^d V_d(x)$$

where $u^1, \dots, u^d$ are scalars and $V_1, \dots, V_d$ are smooth vector fields on $\mathbb{R}^m$.
- We think of $u^1, \cdots, u^d$ as the only d trainable parameters (part of θ) that will be t-dependent. The vector fields $V_1, \dots, V_d$ are specified by the remaining parameters in θ, which will be non-trainable and constant in t.

The role of randomness and few trainable parameters

- Recall that $V(x,\theta)$ was initially specifed by $V(x,\theta_t) = \ell_t(x) - x$ with $\ell_t(x) = (\sigma(\langle A_{t,1}, x\rangle + b_{t,1}), \ldots, \sigma(\langle A_{t,m}, x\rangle + b_{t,m})) = \sigma(A_t x + b_t)$.
- For each layer t, a $m \times m$ matrix A_t and a vector $b_t \in \mathbb{R}^m$ has to be trained.
- Our results imply training of very few parameters: for instance we can specify

$$V(x,\theta_t) = \sum_{i=1}^{7} u_t^i \sigma_i(C_i x + d_i)$$

where C_i is a random matrix, d_i a random vector and σ_i polynomials with random coefficients. Only u_t^i are subject to training to achieve what we call generic expressiveness.

Universal approximation

One form of expressiveness of neural networks is the universal approximation property.

Universal approximation (meta)-theorem

Any continuous (say) function $f : [0,1]^m \to \mathbb{R}$ can be uniformly approximated to arbitrary accuracy by a neural network of sufficient depth and/or width.

- This is a very important part of the theory of deep and shallow learning.
- Prominent contributions include Cybenko ('89), Hornik ('91), Barron ('93), Shaham et al. ('16), Bölcskei et al. ('16), etc.

Universal interpolation

- Another form is what we shall call universal interpolation.
- The system

$$dX_t^x = u_t^1 V_1(X_t^x) + \cdots u_t^d V_d(X_t) dt \qquad (*)$$

turns out to be expressive in the following sense if $V_1, \ldots, V_d$ are chosen appropriately.

Universal interpolation

- Another form is what we shall call universal interpolation.
- The system

$$dX_t^x = u_t^1 V_1(X_t^x) + \cdots u_t^d V_d(X_t)dt \qquad (*)$$

turns out to be expressive in the following sense if $V_1, \dots, V_d$ are chosen appropriately.

Definition

The control system (*), specified by $V_1, \dots, V_d$, is called a universal N-point interpolator on $\Omega \subseteq \mathbb{R}^m$ if, for any training set $\{(x_i, y_i) \in \Omega \times \Omega \colon i = 1, \dots, N\}$, there exist controls $u_t^1, \dots, u_t^d$ that achieve the exact matching

$$X_1^{x_i} = y_i \quad \forall i = 1, \dots, N.$$

Here it is required that the training inputs and outputs are both pairwise distinct.

- Perfect interpolation is not necessarily a desirable training goal, but here it serves as a measure of expressiveness. The readout R is here the identity.

1-point controllability

Toy training data set $N = 1$

- Can we control any input $x \in \mathbb{R}^m$ to any output $y \in \mathbb{R}^m$?

1-point controllability

Toy training data set $N = 1$

- Can we control any input $x \in \mathbb{R}^m$ to any output $y \in \mathbb{R}^m$?
- The answer is given by the classical Chow - Rashevskii theorem.

1-point controllability

Toy training data set $N = 1$

- Can we control any input $x \in \mathbb{R}^m$ to any output $y \in \mathbb{R}^m$?
- The answer is given by the classical Chow - Rashevskii theorem.

Notation

- Lie brackets of vector fields V and W:

$$[V, W](x) = DW(x)V(x) - DV(x)W(x)$$

  Example: For linear vector fields $V(x) = Ax$, $W(x) = Bx$, this is $[V, W](x) = (AB - BA)x$.

- Lie algebra of all vector fields generated by $V_1, \ldots, V_d$:

$$\mathrm{Lie}(V_1, \ldots, V_d) = \mathrm{span}\{V_1, \ldots, V_d \text{ and their iterated Lie brackets}\}$$

- Evaluation of Lie algebra at $x \in \mathbb{R}^m$

$$\mathrm{Lie}(V_1, \ldots, V_d)|_x = \{W(x) : W \in \mathrm{Lie}(V_1, \ldots, V_d)\} \subseteq \mathbb{R}^m$$

Chow-Rashevskii for 1-point controllability

Theorem (Chow - Rashevsky)

If the Hörmander condition

$$Lie(V_1, \dots, V_d)|_x = \mathbb{R}^m$$

holds at every point $x \in \mathbb{R}^m$, then controllability holds: for every input/output pair (x, y) there exist smooth scalar controls $u_t^1, \dots, u_t^d$ that achieve $X_1^x = y$, where X_t is the solution of (*).

Chow-Rashevskii for 1-point controllability

Theorem (Chow - Rashevsky)

If the Hörmander condition

$$Lie(V_1, \dots, V_d)|_x = \mathbb{R}^m$$

holds at every point $x \in \mathbb{R}^m$, then controllability holds: for every input/output pair (x, y) there exist smooth scalar controls $u_t^1, \dots, u_t^d$ that achieve $X_1^x = y$, where X_t is the solution of (*).

Why Lie brackets?

- Consider linear vector fields $V(x) = Ax$ and $W(x) = Bx$.
- Flowing along V for a time t gives $x \mapsto e^{tA}x$.
- Alternating between W, V, $-W$, and $-V$:

$$e^{-tA}e^{-tB}e^{tA}e^{tB}x = x + t^2(AB - BA)x + O(t^3)$$

This produces motion in the direction $[V, W](x) = (AB - BA)x$.

Universal N-point interpolation

Training data set of size N

- Can we simultanously control inputs $\bar{x} = (x_1, \dots, x_N) \in (\mathbb{R}^m)^N$ to outputs $\bar{y} = (y_1, \dots, y_N) \in (\mathbb{R}^m)^N$ using a common set of vector fields and controls?
- If yes, how many and which vector fields do we need?

- Consider the "stacked" system

$$\frac{d}{dt} \underbrace{\begin{pmatrix} X_t^{x_1} \\ \vdots \\ X_t^{x_N} \end{pmatrix}}_{\bar{X}_t^{\bar{x}}} = u_t^1 \underbrace{\begin{pmatrix} V_1(X_t^{x_1}) \\ \vdots \\ V_1(X_t^{x_N}) \end{pmatrix}}_{V_1^{\oplus N}(\bar{X}_t^{\bar{x}})} + \dots + u_t^d \underbrace{\begin{pmatrix} V_d(X_t^{x_1}) \\ \vdots \\ V_d(X_t^{x_N}) \end{pmatrix}}_{V_d^{\oplus N}(\bar{X}_t^{\bar{x}})}.$$

 with initial values in the space of pairwise distinct N-tuples: $\overline{\Omega} = \Omega^N \setminus \Delta$ with $\Delta = \{(x_1, \dots, x_N) \in \Omega^N : x_i = x_j \text{ for some } i \neq j\}$.
- By the Chow–Rashevskii theorem, controllability holds true provided that the N-point Hörmander condition, $\mathrm{Lie}(V_1^{\oplus N}(\bar{x}), \dots, \dots V_d^{\oplus N}(\bar{x})) = (\mathbb{R}^m)^N$ holds at every $\bar{x} = (x_1, \dots, x_N) \in \overline{\Omega}$.

First result

Theorem

Fix $m \geq 2$ and a bounded open connected subset $\Omega \subseteq \mathbb{R}^m$. There exist $d = 5$ smooth bounded vector fields $V_1, \ldots, V_5$ on $\mathbb{R}^m$ such that

$$dX_t^x = u_t^1 V_1(X_t^x) + \cdots u_t^d V_d(X_t)dt \qquad (*)$$

is a universal N-point interpolator in Ω, for every N.

Remarks

- $m = 1$ is not covered (on the real line inputs cannot be directed to outputs if they are differently ordered)
- Note that $d = 5$ is independent of both N and m, and the same vector fields (but not the same controls) work for any N.
- 5 is probably not optimal.

Sketch of the proof

- Let $V_1(x) = Ax$, $V_2(x) = Bx$, where A and B are suitable traceless $m \times m$ matrices and

$$V_3(x) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad V_4(x) = \begin{pmatrix} (x^m)^2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad V_5(x) = \begin{pmatrix} x^1 x^m \\ x^2 x^m \\ \vdots \\ (x^m)^2 \end{pmatrix}.$$

  Then $\mathrm{Lie}(V_1, \ldots, V_5)$ contains all polynomial vector fields.

- The set of all polynomial vector fields on $\mathbb{R}^m$ interpolates at every $\bar{x} \in \overline{\Omega}$, i.e. for every $\bar{x} \in \overline{\Omega}$ and $\bar{y} \in (\mathbb{R}^m)^N$ there exists some polynomial vector field V s.t. $V(x_i) = y_i$ for all i. The same property thus holds for $\mathrm{Lie}(V_1, \ldots, V_5)$.

- This implies the N-point Hörmander condition for these five vector fields at every $\bar{x}$ and in turn the N-point interpolator property.

Generic expressiveness

- So far: universal interpolators can be constructed using just five vector fields.
- Our next goal is to prove that such expressive systems are generic.
- Appropriately randomly chosen nonlinear polynomial vector fields allow to generate controlled ODEs (*) that are sufficiently expressive to interpolate almost every training set.
- Instead of using the identity as final read out, we relate here the input x and output y via

$$\begin{aligned} y = \lambda(X_1^x - x), \text{ with } X_1^x \text{ solving} \\ dX_t^x = u_t^1 V_1(X_t^x) + \cdots u_t^d V_d(X_t^x) \end{aligned} \qquad (**)$$

where λ is some scalar parameter which has to be trained.

Main result - Ingredients

- Bounded open connected subset $\Omega \subset \mathbb{R}^m$, $m \geq 2$
- A polynomial vector field V of degree at most k has components of the form
$$V^j(x) = \sum_{|\alpha|=0}^{k} c_\alpha^j x^\alpha.$$
Coefficient vector: $\mathbf{c} = (c_\alpha^j : j = 1, \ldots, m, |\alpha| \leq k) \in \mathbb{R}^{D_k}$ where $D_k = m\binom{m+k}{m}$.
- $d \geq 5$ polynomial vector fields $V_1, \ldots, V_d$ of degree at most $k \geq 2$ in Ω, with coefficients $(\mathbf{c}_1, \ldots, \mathbf{c}_d)$
- For some $l \in \mathbb{N}$, some polynomial map $Q : \mathbb{R}^l \to (\mathbb{R}^{D_k})^d$, and some random vector Z in $\mathbb{R}^l$, we assume the coefficients are drawn randomly in the following way.
$$(\mathbf{c}_1, \ldots, \mathbf{c}_d) = Q(Z).$$

Main result

Theorem (C., M. Larsson and J. Teichmann)

Assume that

1. *the law of Z admits a probability density on $\mathbb{R}^l$;*
2. *for some $\widehat{z} \in \mathbb{R}^l$, the Lie algebra generated by the vector fields with coefficients $(\widehat{\mathbf{c}}_1, \ldots, \widehat{\mathbf{c}}_d) = Q(\widehat{z})$ contains all polynomial vector fields;*
3. *the training data set is generic: inputs x_i of $\{(x_i, y_i) \in \Omega \times \Omega \colon i = 1, \ldots, N\}$ are drawn from some density on $(\Omega)^N$; outputs y_i are just pairwise distinct.*

Then, with probability one, (**) forms a universal interpolator, i.e. there exist controls $u_t^1, \ldots, u_t^d$ and a constant $\lambda > 0$ such that $y_i = \lambda(X_1^{x_i} - x_i)$ for all i.

Example

- Let $l = (D_k)^5$ and Q be the identity map.
- Draw $(\mathbf{c}_1, \ldots, \mathbf{c}_d)$ from any density on $(\mathbb{R}^{D_k})^5$.
- Take $\widehat{z} = (\widehat{\mathbf{c}}_1, \ldots, \widehat{\mathbf{c}}_5)$, where $(\widehat{\mathbf{c}}_1, \ldots, \widehat{\mathbf{c}}_5)$ are the coefficients of the specific 5 polynomial vector fields from above.

Idea of the proof

- Let $\vec{V} = (V_1, \ldots, V_5)$ be polynomial vector fields parameterized by coefficients $(\mathbf{c}_1, \ldots, \mathbf{c}_5) \in (\mathbb{R}^{D_k})^5$ such that $(\mathbf{c}_1, \ldots, \mathbf{c}_5) = Q(Z)$ for $Q : \mathbb{R}^l \to (\mathbb{R}^{D_k})^5$.
- Let $(\widehat{\mathbf{c}}_1, \ldots, \widehat{\mathbf{c}}_5) = Q(\widehat{z})$ are the coefficients of the specific 5 polynomial vector fields $\widehat{V}_1, \ldots, \widehat{V}_5$ from the previous theorem.
- $\mathrm{Lie}(\widehat{V}_1, \ldots, \widehat{V}_5)$ contains a basis $E_1(\vec{\widehat{V}}, x), \ldots, E_{D_n}(\vec{\widehat{V}}, x)$ for the space of polynomial vector fields of degree at most n.
- To guarantee that $\langle E_1(\vec{V}, \cdot), \ldots E_{D_n}(\vec{V}, \cdot)\rangle$ interpolates at $(x_1, \ldots, x_N) \in \Omega^N$, the $mN \times D_n$ matrix

$$\begin{pmatrix} E_1(\vec{V}, x_1) & \cdots & E_{D_n}(\vec{V}, x_1) \\ \vdots & & \vdots \\ E_1(\vec{V}, x_N) & \cdots & E_{D_n}(\vec{V}, x_N) \end{pmatrix}$$

  has to have columns that span $(\mathbb{R}^m)^N$, i.e. the determinant of at least one $mN \times mN$ matrix has to be nonzero.

Idea of the proof

- E_j are polynomials in x, but also in Z (seen as function in Z generating V). The same holds for the squared determinant $\Gamma_n(x_1, \dots, x_N, Z)$.
- Since for n big enough the squared determinant $\Gamma_n(x_1, \dots, x_N, \widehat{z}) > 0$ for every pairwise distinct data set, we can conclude that

$$(x_1, \dots, x_N, Z) \mapsto \Gamma_n(x_1, \dots, x_N, Z)$$

it is not identically zero. Here Condition (2) is needed.

- The density condition on the data set and Z is used to avoid zeros which can exist, but which only constitute a nullset.

Idea of the proof

- E_j are polynomials in x, but also in Z (seen as function in Z generating V). The same holds for the squared determinant $\Gamma_n(x_1, \dots, x_N, Z)$.
- Since for n big enough the squared determinant $\Gamma_n(x_1, \dots, x_N, \widehat{z}) > 0$ for every pairwise distinct data set, we can conclude that

$$(x_1, \dots, x_N, Z) \mapsto \Gamma_n(x_1, \dots, x_N, Z)$$

  it is not identically zero. Here Condition (2) is needed.
- The density condition on the data set and Z is used to avoid zeros which can exist, but which only constitute a nullset.
- With probability 1, $\mathrm{Lie}(V_1, \dots, V_5)$ interpolates at $\bar{x}$.
  $\Rightarrow$ N-point Hörmander condition holds at $\bar{x}$.
- By continuity, there is an open connected neighborhood $\mathcal{U} \subset \Omega^N$ of $\bar{x}$ where the Hörmander condition holds. $\Rightarrow$ Choose $\lambda > 0$ large enough so that $\bar{x} + \lambda^{-1}\bar{y} \in \mathcal{U}$.
- The Chow - Rashevskii theorem then implies that $x_i + \lambda^{-1} y_i$ can be reached $X_1^{x_i}$ for all $i = 1, \dots, N$.

Concrete example of neural network typ

Corollary

Consider $d = 7$ vector fields of the form $V_i(x) = \sigma_i(C_i x + b_i), i = 1, \ldots, 7,$ where

- *C_i is a random matrix in $\mathbb{R}^{m \times m}$, b_i a random vector in $\mathbb{R}^m$, and*
- *$\sigma_i(\cdot)$ a polynomial nonlinearity, whose coefficients depend polynomially on some random vector Z_0.*

Assume that

1. *the random elements $Z = (Z_0, C_1, \ldots, C_7, b_1, \ldots, b_7)$ admit a joint density;*
2. *for some value $\widehat{z}_0$ of $supp(law(Z_0))$, we have $\sigma_i(r) = r$ for $i = 1, 2, 3$, and $\sigma_i(r) = r^2$ for $i = 4, 5, 6, 7$;*
3. *the training data set is generic.*

Then with probability one, (**) forms a universal interpolator in the sense of the above Theorem.

Consequences for training

- Universal interpolation is a generic property.
- In practice, the CODE (*) is replaced by a discretization, say with M steps.
- This yields a network of depth M . After randomly choosing d vector fields, the number of trainable parameters (including λ) becomes $Md + 1$.
- This tends to be much smaller than the total number of parameters needed to specify the vector fields, and can potentially simplify the training task significantly.
- The fact that most parameters are chosen randomly reinforces the view that randomness is a crucial ingredient for training.

Conclusion

- Deep feedforward neural networks can be modeled as controlled dynamical systems.
- Expressiveness can be proved in this formulation using classical results on controllability.
- Expressiveness is generic since $\mathrm{Lie}(V_1^{\oplus N}, \ldots V_d^{\oplus N})$ generically spans $(\mathbb{R}^m)^N$.
- Many parameters can be chosen randomly, which truely works in applications.
- We illustrate this with the MNIST data set by training a generic network with much less trainable parameters than in the standard implementation.

Part II

Gradient descent and backpropagation

Supervised learning task with neural networks

Supervised learning

Given training data $\{(x_i, y_i), i = 1, \dots N\}$ with $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}^d$, find a neural network g within a class of neural networks NN_Θ with a certain architecture characterized by parameters $\theta \in \Theta$, such that

$$g \in \operatorname*{argmin}_{NN_\Theta} \sum_{i=1}^{N} \mathcal{L}(g(x_i), y_i),$$

where $\mathcal{L}$ is a loss function: $C(\mathbb{R}^m, \mathbb{R}^{\widetilde{d}}) \times \mathbb{R}^d \to \mathbb{R}_+$. Note that the input dimension of the neural network is m and the output dimension $\widetilde{d}$.

Since g is determined by the parameters θ, the above optimization corresponds to searching the minimum in the parameter space Θ which is nothing else than the collection of $(A_t, b_t)_{t=1,\dots,n}$ (if we have n hidden layers) and the readout map R.

Examples

- Example MNIST classification: $x_i \in \mathbb{R}^{28\times 28}$, i.e. $m = 28 \times 28$ and $y \in \mathbb{R}$, i.e. $d = 1$. The output dimension of the neural network is $\widetilde{d} = 10$. The loss function is given by

$$\mathcal{L}(g(x), y) = \sum_{k=1}^{10} 1_{\{y=k-1\}} \log((g(x))_k).$$

- Example classical regression with L^2 loss:

$$\mathcal{L}(g(x), y) = \|g(x) - y\|^2.$$

Here the output dimension of the neural network $\widetilde{d}$ is equal to d.

But how...?

- ... to deal with a non-linear, non-convex optimization problem and with around 600 000 parameters, as it is the case for the MNIST data set?

Gradient descent: the simplest method

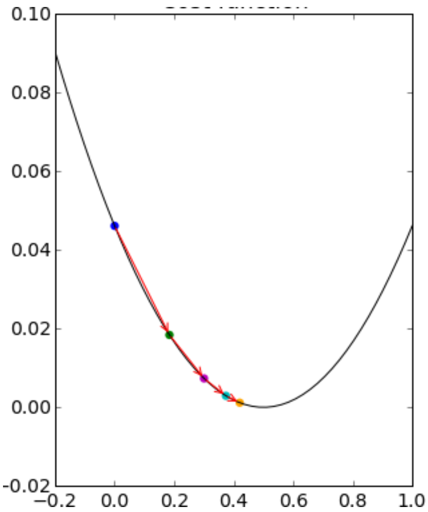
- The gradient of a function $F(\theta) : \mathbb{R}^M \to \mathbb{R}$ is given by

$$\nabla F(\theta) = (\partial_{\theta_1} F(\theta), \ldots, \partial_{\theta_M} F(\theta)).$$

- Gradient descent:
  starting with an initial guess $\theta^{(0)}$, one iteratively defines for some learning rate η_k

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla F(\theta^{(k)})$$

Gradient descent: the simplest method



Classical convergence result

Theorem

Suppose the function $F : \mathbb{R}^M \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $\|\nabla F(\theta) - \nabla F(\beta)\| \leq L\|\theta - \beta\|$ for any $\theta, \beta \in \mathbb{R}^M$. Then if we run gradient descent for k iterations with a fixed step size $\eta \leq 1/L$, it will yield a solution $F(\theta^{(k)})$ which satisfies

$$F(\theta^{(k)}) - F(\theta^*) \leq \frac{\|\theta^{(0)} - \theta^*\|^2}{2\eta k},$$

where $F(\theta^*)$ is the optimal value. Intuitively, this means that gradient descent is guaranteed to converge and that it converges with rate $O(1/k)$.

In practice, the convexity condition is often not satisfied. Moreover, the solution depends crucially on the inital value.

How to compute the gradient

- Nevertheless all optimization algorithms build on the classical idea of gradient descent usually in its enhanced form of stochastic gradient descent.
- How to compute the gradient in our case of supervised learning, where

$$F(\theta) = \sum_{i=1}^{N} \mathcal{L}(g(x_i|\theta), y_i)$$

  and θ corresponds to $(A_t, b_t)_{t=1,\ldots,n}$ (if we have n hidden layers) and the readout map R? We here indicate the dependence of the neural network on θ.
- We suppose here for simplicity that the readout map R is linear, i.e.

$$R(x) = A_{n+1}x + b$$

  where A_{n+1} has $\widetilde{d}$ rows and $b \in \mathbb{R}^{\widetilde{d}}$, so that $\theta = \{(A_t, b_t)_{t=1,\ldots,n+1}\}$.

Backpropagation

- Since
$$\nabla_\theta F(\theta) = \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(g(x_i|\theta), y_i),$$
we need to determine $\partial_{A_t,kl}\mathcal{L}(g(x|\theta), y)$ and $\partial_{b_t,k}\mathcal{L}(g(x|\theta), y)$.
- By the chain rule this is given by
$$\partial_{A_t,kl}\mathcal{L}(g(x|\theta), y) = \langle \partial_g \mathcal{L}(g(x|\theta), y), \partial_{A_t,kl} g(x|\theta)\rangle$$
$$\partial_{b_t,k}\mathcal{L}(g(x|\theta), y) = \langle \partial_g \mathcal{L}(g(x|\theta), y), \partial_{b_t,k} g(x|\theta)\rangle.$$
- Output Layer:
$$\partial_{A_{n+1},kl}\mathcal{L}(g(x|\theta), y) = (\partial_g \mathcal{L}(g(x|\theta))y)_k (\underbrace{\sigma(A_n(\cdots) + b_n)}_{x(n)})_l$$
$$\partial_{b_{n+1},k}\mathcal{L}(g(x|\theta), y) = (\partial_g \mathcal{L}(g(x|\theta))y)_k$$

Backpropagation: second last layer

- Recall $x(t+1) = \sigma(z_{t+1})$ where $z_{t+1} = A_{t+1}x(t) + b_{t+1}$ and $g = z_{n+1} = A_{n+1}x(n) + b_{n+1}$.
- To continue with the second last layer, we use the chain rule again

  Note that $\mathcal{L}(g, y) = \mathcal{L}(z_{n+1}, y) = \mathcal{L}(A_{n+1}x(n) + b_n, y)$. Hence...

$$
\begin{aligned}
\partial_{A_n,kl}\mathcal{L} &= \langle \partial_{x(n)}\mathcal{L}, \partial_{A_n,kl}x(n)\rangle = \langle A_{n+1}\partial_g\mathcal{L}, \partial_{A_n,kl}x(n)\rangle \\
&= \langle A_{n+1}\partial_g\mathcal{L}, \mathrm{diag}(\sigma'(z_n)) \qquad \underbrace{\partial_{A_n,kl}z_n}_{\text{similar as in the last layer}} \qquad \rangle
\end{aligned}
$$