Serie 6
Approssimazione di Funzioni e Dati

1 Interpolazione polinomiale (di Lagrange)

I polinomi vengono rappresentati in Matlab® come degli array. In particolare, un generico polinomio di grado n

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

corrisponde ad un array (riga) di $n+1$ elementi

$$\texttt{p = [}a_n, a_{n-1}, \ldots, a_1, a_0\texttt{]}.$$

Supponendo di avere le $n+1$ coppie $\{x_i, y_i\}$, $i = 0, \ldots, n$, con x_i distinti, il polinomio interpolatore di Lagrange Π_n associato a queste coppie può essere calcolato tramite il comando `polyfit`. In particolare, il comando

```
p = polyfit(x,y,n)
```

restituisce i coefficienti di Π_n in `p`, dove `x=[`$x_0, \ldots, x_n$`]` e `y=[`$y_0, \ldots, y_n$`]`. Una volta calcolato `p`, il polinomio Π_n può essere valutato in `z` tramite il comando

```
pz = polyval(p,z).
```

Il parametro di input `z` può essere uno scalare, o in generale una matrice. In quest'ultimo caso, la valutazione viene eseguita elemento per elemento.

Ad esempio, la valutazione del polinomio $p(x) = x^2 - 1$ nei punti 1 e 2, potrà essere eseguita in Matlab® con il comando `polyval([1 0 -1],[1 2])`, che restituirà come output il vettore `[0 3]`.

Esercizio 1

Si consideri la funzione:

$$f(x) = x\,\sin(x).$$

1. Si disegni il grafico della funzione nell'intervallo $[-2, 6]$.
2. Si costruiscano i polinomi interpolanti di Lagrange $\Pi_n f$ di grado $n = 2$, 4, 6 relativi ad una distribuzione di nodi equispaziati e rappresentarli graficamente.
3. Si rappresenti graficamente l'errore $\varepsilon_n(x) = |f(x) - \Pi_n(x)|$ e si calcoli la sua norma infinito per $n = 2$, 4, 6:

$$||\varepsilon_n(x)||_\infty = \max_{x\in[-2,6]} |f(x) - \Pi_n f(x)|.$$

(*Suggerimento:* Si utilizzino opportunamente le funzioni Matlab® `polyfit` e `polyval`.)

Esercizio 2

Si consideri la funzione:

$$f(x) = \sin\left(\frac{1}{1+x^2}\right),$$

nell'intervallo $I = [-2\pi, 2\pi]$.

1. Si calcolino i polinomi di Lagrange $\Pi_n f(x)$ di grado $n = 2,\ 4,\ 8,\ 10$ relativi ad una distribuzione di nodi equispaziati nell'intervallo I. Si confrontino i grafici dei $\Pi_n f(x)$ con quello della funzione $f(x)$.

2. Calcolare per $n = 2,\ 4,\ 8,\ 10$ la norma infinito dell'errore $\varepsilon_n(x)$ (definito come nell'esercizio 1) e rappresentarla su un grafico in funzione del grado n. Che fenomeno si osserva?

3. Ripetere i punti 1) e 2) utilizzando la distribuzione di nodi di Chebyshev, definiti per un generico intervallo $[a, b]$ come segue:

   $$x_k := \frac{b-a}{2}\,t_k + \frac{a+b}{2}, \quad \text{dove} \quad t_k := -\cos(\pi k/n), \quad \text{per } k = 0, \ldots, n.$$

   Si commentino i risultati ottenuti.

Esercizio 3

Si consideri la funzione $f(x) = -x^3 + 3x^2 - 2$ definita sull'intervallo $I = [0, 2]$.

- Si scriva l'espressione analitica delle funzioni di base lagrangiane per la terna di punti $x_0 = 0,\ x_1 = \frac{1}{2}$ e $x_2 = 2$ e si costruisca il polinomio interpolatore di Lagrange associato.
- Si ripeta il punto 1) prendendo $x_1 = 1$. Cosa si osserva in questo caso?
- Quali sono i coefficienti del polinomio interpolatore di Lagrange di grado 3, che interpola $f(x)$ nei nodi $x_0 = 0,\ x_1 = e^{-\sqrt{2}},\ x_2 = 3^{-\sqrt{0.5}}$ e $x_3 = 2$?

2 Approssimazione polinomiale nel senso dei minimi quadrati

Nel caso in cui si voglia approssimare nel senso dei minimi quadrati un insieme di coppie di dati $\{(x_i, y_i)\}$, $i = 0, \ldots, n$, con x_i nodi distinti, i comandi Matlab®da utilizzare sono ancora `polyfit` e `polyval`. Infatti, dato un numero $m < n$, il comando

```
p = polyfit(x,y,m)
```

restituisce il polinomio approssimante di grado m nel senso dei minimi quadrati, associato ai punti assegnati. Il funzionamento di `polyval` è invece del tutto analogo al caso mostrato nel Laboratorio 12 per l'interpolazione polinomiale.

3 Interpolante lineare a tratti e spline naturale cubica

In questi ultimi due casi, la funzione approssimante *non* è un polinomio, ma un polinomio *a tratti*. Al contrario dei polinomi, le due fasi di interpolazione e valutazione, che prima erano distinte, ora risultano accorpate in un unico comando.[1] Nel caso dell'interpolazione lineare a tratti, il comando

```
pz = interp1(x,y,z)
```

genera il polinomio lineare a tratti $\Pi_1^H(x)$ interpolante nelle coppie corrispondenti ai vettori `x=[`$x_0, \ldots, x_n$`]` e `y=[`$y_0, \ldots, y_n$`]`, e lo valuta nelle ascisse contenute nel vettore `z`, fornendo il risultato della valutazione in `pz`.

Analogamente, la valutazione della spline naturale cubica interpolante viene fatta tramite il comando

```
pz = cubicspline(x,y,z),
```

dove `cubicspline.m` è una function fornita che sfrutta i comandi `csape` e `fnval` di Matlab®, non inclusi nel programma di questo corso. Alternativamente, può essere utilizzato il comando Matlab®`spline`, che però genera la spline interpolante utilizzando le condizioni di chiusura "not-a-knot" (si consulti l'`help` di Matlab®per maggiori informazioni).

Esercizio 4

Sono state svolte delle prove a trazione su una nuova lega per determinare la relazione tra lo *sforzo* σ (forza per unità di superficie, $[1000 \times \mathrm{kg_F}/\mathrm{cm}^2]$) e la *deformazione* ε (allungamento per unità di lunghezza, [cm/cm]). I risultati delle prove sono riportati nella seguente tabella:

σ	0.1800	0.3000	0.5000	0.6000	0.7200	0.7500	0.8000	0.9000	1.0000
ε	0.0005	0.0010	0.0013	0.0015	0.0020	0.0045	0.0060	0.0070	0.0085

A partire da questi dati si vuole stimare la deformazione ε della lega in corrispondenza dei valori dei sforzo per cui non si ha a disposizione un dato sperimentale. A tal fine si utilizzano opportune tecniche di interpolazione.

Gli interpolanti da utilizzare sono i seguenti:

- interpolazione polinomiale di Lagrange (`polyfit` e `polyval`);
- interpolazione polinomiale composita lineare (`interp1`);
- spline cubica naturale interpolante (`cubicspline`);
- spline cubica interpolante con condizioni di chiusura "not-a-knot" (`spline`);

si considerino inoltre le seguenti:

- approssimazioni nel senso dei minimi quadrati di grado 1, 2 e 4 (`polyfit` e `polyval`).

[1] In realtà è possibile distinguerle, poichè in Matlab®è possibile memorizzare anche polinomi a tratti nella cosiddetta `ppform`. Si veda, ad esempio, l'help del comando `spline`.

In particolare si richiede di:

1. rappresentare graficamente le singole funzioni interpolanti ed approssimanti a confronto con i dati sperimentali;
2. confrontare in un unico grafico i dati sperimentali con tutte le interpolanti e approssimanti (per l'approssimante ai minimi quadrati si consideri solo il polinomio di grado 4);
3. valutare, per ogni interpolante ed approssimante, la deformazione ε in corrispondenza di $\sigma = 400$ $\mathrm{kg_F/cm^2}$ e $\sigma = 650$ $\mathrm{kg_F/cm^2}$; si commentino i risultati ottenuti.

Esercizio 5

Si consideri la funzione:

$$f(x) = e^{-x^2}\sin(x), \qquad \text{con } x \in [a,b].$$

Si prendano gli estremi $a = -2$ e $b = 3$.

1. Si calcoli il polinomio interpolante composito lineare $\Pi_1^H f(x)$ su $n = 3$ sottointervalli di uguale ampiezza $H = (b-a)/n$, si utilizzi la funzione `interp1`, e se ne disegni il grafico insieme a quello della funzione $f(x)$.
2. Si calcoli l'errore in norma infinito
$$\epsilon_H = \max_{x\in[a,b]} \left| f(x) - \Pi_1^H f(x) \right|.$$
3. Si calcoli ora il polinomio interpolante composito lineare $\Pi_1^H f$ su $n = 4, 8, 16, 32, 64, 128$ sottointervalli di uguale ampiezza. Si valuti l'errore in norma infinito ϵ_H in ciascun caso e se ne visualizzi l'andamento in funzione di H su un grafico in scala logaritmica su entrambi gli assi. Verificare graficamente che ci sia accordo con la stima teorica dell'errore:
$$\epsilon_H \leq \frac{H^2}{8} \max_{x\in[a,b]} \left| f''(x) \right|.$$

Esercizio 6

Sono state effettuate prove di caduta libera di un grave. Tale grave viene lasciato cadere dalla quota $y_0 = 10$ m. La caduta è misurata con uno strumento di scarsa precisione che fornisce l'altezza dal suolo ogni $\Delta t = 0.1$ s. Il tempo totale della simulazione è pari a $T = 1$ s. I dati ottenuti sperimentalmente per l'altezza sono:

$$y_S = [\ 10.0 \ \ 9.89 \ \ 9.75 \ \ 9.66 \ \ 9.10 \ \ 8.95 \ \ 8.10 \ \ 7.49 \ \ 6.80 \ \ 6.13 \ \ 5.05\ ]$$

1. Si rappresentino sullo stesso grafico i dati sperimentali e la legge di moto ideale (in linea continua) data da:
$$y = y_0 - \frac{1}{2}\, g\, t^2, \qquad g = 9.81 \text{ m/s}^2.$$

2. A partire dai dati sperimentali disponibili si determinino:
   - il polinomio interpolante di Lagrange;
   - l'interpolante composita lineare;
   - l'approssimante polinomiale di grado 2, nel senso dei minimi quadrati.

   Si confrontino graficamente le approssimazioni, (sull'intervallo $[0, 1]$), con l'andamento previsto dalla legge ideale. Si faccia inoltre un confronto grafico della distribuzione dell'errore commesso da ogni interpolante e approssimazione e si valuti anche l'errore in norma infinito.

3. Si stimi l'altezza raggiunta dal grave al tempo $t = 1.05$ s sulla base dei dati sperimentali: in particolare si confrontino le stime ottenute dall'interpolazione polinomiale di Lagrange e dal polinomio approssimante nel senso dei minimi quadrati (di grado 2) con il risultato fornito dalle legge ideale. Si commenti il risultato ottenuto.

Esercizio 7

Si consideri un generico strumento di misurazione digitale che campioni un segnale espresso dalla funzione $g(x) = 10x^2$ per N valori di x nell'intervallo $I = [0, 1]$. Le misure rilevate dallo strumento sono affette da rumore casuale e pertanto si possono rappresentare mediante una funzione $f(x) = g(x) + \varepsilon(x)$, dove $|\varepsilon(x)| \leq 1$.

Tale rumore può essere espresso in Matlab dall'espressione `2*rand(size(x))-1`, che restituisce un vettore di valori pseudo-casuali nell'intervallo $[-1, 1]$, e dunque consideriamo le seguenti definizioni per g e f:

```
g = @(x) 10 * x.^2;                   % Segnale fisico
f = @(x) g(x) + 2*rand(size(x))-1;   % Rilevazione dello strumento
```

Si noti che la funzione `f` restituisce un vettore diverso ogni volta che viene valutata (anche se `x` contiene gli stessi valori).

1. Usando i comandi Matlab `polyfit` e `polyval`, calcolare il polinomio $\Pi_9 f(x)$ di grado $n = 9$ interpolante $f(x)$ in $n+1$ nodi equispaziati su I. Usando gli stessi nodi, si calcoli il polinomio $\widetilde{f}_2(x)$ di grado $m = 2$ che approssima $f(x)$ nel senso dei minimi quadrati. Si traccino, dunque, in un'unica figura i grafici di $f, g, \Pi_9 f, \widetilde{f}_2$. Quale polinomio approssima meglio il segnale originale?

2. Usare i polinomi $\Pi_9 f(x)$ e $\widetilde{f}_2$ per estrapolare il valore di $g(x)$ in $x = 2$. Si discutano i risultati ottenuti.

3. A causa della presenza di rumore, misurazioni ripetute forniscono tipicamente segnali $f(x)$ diversi. Questa caratteristica è già inclusa nell'implementazione Matlab considerata; infatti, la funzione `rand` restituisce valori diversi ad ogni chiamata. Pertanto, è possibile analizzare la stabilità delle approssimazioni polinomiali $\Pi_9 f$ e $\widetilde{f}_2$ valutando le variazioni dei risultati rispetto alle variazioni delle coppie $\{(x_i, f(x_i)\}_{i=0}^{9}$ fornite come input.

   Cosa si può osservare, ripetendo i punti precedenti? Si discutano i risultati ottenuti.

4 Minimi quadrati non lineari e reti neurali

Consideriamo ora approssimazioni di funzioni costruite mediante il metodo dei minimi quadrati non lineari e reti neurali.

Esercizio 8

Il campo di velocità di un getto turbolento libero mediato in tempo (Reynolds-averaged), essendo caratterizzato da una componente radiale trascurabile, può essere descritto attraverso la sola componente assiale come

$$u(a,r) = \frac{U\,d}{a}\; e^{-A\,r^2/a^2}, \tag{1}$$

con a e r, rispettivamente, la distanza assiale e radiale dal punto di uscita del getto (Figura 1). Il parametro U rappresenta la velocità caratteristica del getto, tipicamente corrispondente alla velocità di uscita da un ugello di diametro d, mentre A è un parametro adimensionale che ne determina la decrescita in direzione radiale.

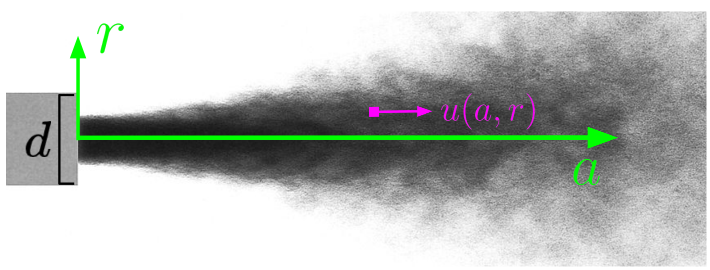

Figura 1: Esempio di un getto turbolento libero assialsimmetrico in uscita da un ugello di diametro d; a e r rappresentano, rispettivamente, la distanza assiale e radiale del getto dal punto di uscita.

1. Scrivere una funzione Matlab® `gauss_newton.m` che, prese in ingresso le coppie di dati nei vettori `x` e `y`, minimizza la funzione obiettivo

$$\frac{1}{2}\sum_{i=1}^{n} r^2(x_i, y_i; \mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n} (y_i - \tilde{f}(x_i; \mathbf{w}))^2,$$

   tramite il metodo di Gauss-Newton. In particolare, r è la funzione residuo dipendente da un vettore di parametri $\mathbf{w}$ da determinare, mentre $\tilde{f}(x;\mathbf{w})$, dove $\tilde{f}(\cdot;\mathbf{w}) : \mathbb{R} \to \mathbb{R}$ rappresenta la funzione approssimante da costruire e dipendente dai parametri $\mathbf{w}$. Si consideri la seguente intestazione

```
[w_vect, it] = gauss_newton(x, y, r_fun, J_fun, w0, toll, maxit)
```

   dove `r_fun` e `J_fun` sono *function handles* rispettivamente per la funzione residuo r e il suo Jacobiano, `w0` è la guess iniziale, `maxit` il numero massimo di iterazioni e `toll` un'opportuna tolleranza sul criterio d'arresto basato sul gradiente.

2. Si vuole stimare la componente assiale della velocità di un getto turbolento in corrispondenza della posizione assiale $a = 0.8$ tramite la funzione

$$\tilde{u}(a, r) = \frac{w_1\, d}{a}\, e^{-w_2\, r^2/a^2},$$

dove $d = 1$, mentre la velocità caratteristica $U = w_1$ e il parametro adimensionale $A = w_2$ rappresentano i parametri incogniti da determinarsi. Dobbiamo dunque determinare la funzione

$$\tilde{f}(x; \mathbf{w}) = \frac{w_1\, d}{a}\, e^{-w_2\, x^2/a^2},$$

essendo $x = r$, la distanza radiale del getto dal punto di uscita. Data la dipendenza non lineare rispetto a w_2, si fornisca una stima dei parametri $\mathbf{w} = (w_1,\, w_2)^T$ risolvendo il problema nel senso dei minimi quadrati non lineari mediante la funzione `gauss_newton.m` implementata. Si generino i dati su una griglia $\{r_1, ..., r_{50}\} = \{x_1, ..., x_{50}\}$ di $n = 50$ punti equispaziati nell'intervallo $[-0.5, 0.5]$ usando Eq. (1) con $U = U_{true} = 1$ e $A = A_{true} = 20$ per determinare $y_i = u(a, x_i)$ per $i = 1, \ldots, n$. Osserviamo come w_1 e w_2 rappresentino rispettivamente le approssimazioni di U e A. Considerando $\mathbf{w}^{(0)} = (2,\, 2)^T$ come guess iniziale e tolleranza $tol = 10^{-6}$, si riporti il numero di iterazioni richiesto dal metodo per arrivare a convergenza.

3. Si ripeta la stima dei parametri $\mathbf{w} = (w_1,\, w_2)^T$ tramite il metodo di Gauss-Newton considerando dati affetti da rumore gaussiano $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ con $\sigma = 3 \cdot 10^{-2}$, commentando il risultato ottenuto. *Suggerimento*: è possibile generare numeri casuali gaussiani con la funzione Matlab® `randn`.

Esercizio 9

Una generica funzione $\mathbf{f} : I \subset \mathbb{R}^d \to \mathbb{R}^m$, con I compatto, può essere approssimata mediante una *shallow neural network* $\tilde{\mathbf{f}}$, ovvero una rete neurale caratterizzata da un singolo hidden layer

$$\hat{\mathbf{y}} = \tilde{\mathbf{f}}(\mathbf{x}) := W_2\, \underbrace{\rho(\overbrace{W_1\, \mathbf{x} + \mathbf{b}_1}^{\mathbf{z}})}_{\mathbf{h}} + \mathbf{b}_2. \tag{2}$$

Considerando una collezione (*batch*) di $n \geq 1$ dati di input raggruppati nella matrice $X = \left[\mathbf{x}_1 | \cdots | \mathbf{x}_n\right] \in \mathbb{R}^{d \times n}$ è possibile riscrivere il *forward pass* di Eq. (2) in notazione matriciale come

$$\hat{Y} = \tilde{f}(X) := W_2\, \underbrace{\rho(\overbrace{W_1\, X + \mathbf{b}_1}^{\mathbf{Z}})}_{\mathbf{H}} + \mathbf{b}_2 \tag{3}$$

con $\mathbf{H}$, $\mathbf{Z} \in \mathbb{R}^{h \times n}$ matrici di *hidden states* e *pre-attivazioni*. Così facendo, si possono ottenere le rispettive valutazioni della rete neurale $\hat{Y} = \left[\hat{\mathbf{y}}_1 | \cdots | \hat{\mathbf{y}}_n\right] \in \mathbb{R}^{m \times n}$.

1. Completare l'implementazione del *forward pass* in formulazione matriciale fornita nel file `nn_forward.m` attraverso la funzione

```
[y_hat, h, z] = nn_forward(x,params,activation_fn)
```

dove `x` rappresenta una collezione di dati di input, `params` è una lista contenente i parametri della rete $\{W_1, \mathbf{b}_1, W_2, \mathbf{b}_2\}$, e `activation_fn` è una stringa che specifica la tipologia di funzione di attivazione da utilizzare (e.g. `'Sigmoid'`).

Suggerimento: Si utilizzi la funzione `act_fn.m` fornita per l'applicazione della funzione di attivazione ρ.

2.a Allenare una rete con dimensione dello stato hidden $h = 20$ e funzione di attivazione `'Sigmoid'` per l'approssimazione della funzione $f(x) = x\sin(4\pi x) + x^2$ nell'intervallo $[0, 1]$, considerando un sampling di $N = 100$ punti da una distribuzione uniforme $\{x_i\}_{i=0}^{N} \sim \mathcal{U}[0, 1]$, seguendo il template fornito in `es9_2.m`. Per aggiornare la lista di parametri `params` durante l'allenamento, è possibile considerare l'algoritmo di discesa del gradiente stocastico (SGD), implementato nella funzione Matlab®`sgd_step.m` da usarsi con la seguente intestazione

```
[params] = sgd_step(params, grads, lr)
```

dove `grads` indica una lista contentente i gradienti della loss rispetto ai parametri della rete $\{\nabla_{W_1}\mathcal{L}, \nabla_{\mathbf{b}_1}\mathcal{L}, \nabla_{W_2}\mathcal{L}, \nabla_{\mathbf{b}_2}\mathcal{L}\}$, da calcolare mediante la funzione fornita

```
[loss, grads] = nn_backward(y, y_hat, x, h, z, params, activation_fn)
```

che implementa il backward pass e restituisce anche il valore della funzione costo `loss` in norma di Frobenius, ovvero $\|\texttt{y} - \texttt{y\_hat}\|_F^2$.

Con `lr` si indica il parametro di *learning rate*, in particolare si adotti la seguente learning rate che decresce ad ogni iterazione

$$\eta_i = 10^{-2}\,(0.9)^{i/\texttt{maxit}} \tag{4}$$

dove i e `maxit`$=5 \cdot 10^4$ rappresentano, rispettivamente, l'iterazione corrente dell'allenamento e il numero massimo di iterazioni. Inizializzare i parametri della rete attraverso un sampling da una distribuzione uniforme $\mathcal{U}[-\sqrt{k}, \sqrt{k}]$, con k la dimensione di input del rispettivo layer.

Suggerimento: utilizzare la funzione `sample_minibatch.m` per estrarre un sottoinsieme di dati (*minibatch*) ad ogni iterazione dell'allenamento, con dimensione di minibatch pari a 32.

2.b Ripetere l'allenamento della rete neurale utilizzando l'algoritmo di ottimizzazione *Adam*, implementato nella funzione Matlab®`adam_step.m` da usarsi con la seguente intestazione

```
[params, m, v] = adam_step(params, grads, m, v, lr, beta1, beta2, it)
```

dove `m` e `v` indicano le liste dei momenti primi e secondi, `beta1` e `beta2` sono i rispettivi parametri di decay, e `it` rappresenta l'iterazione corrente dell'allenamento. In particolare, si consideri $\beta_1 = \beta_2 = 0.8$.

Confrontare i risultati ottenuti mediante SGD e Adam sulla base degli andamenti della funzione costo durante l'allenamento. Confrontare, inoltre, gli errori di approssimazione ottenuti su una griglia di $N = 500$ nodi equispaziati nell'intervallo $[0, 1]$.

3. Si consideri il problema di approssimare il campo di velocità assiale di un getto turbolento assialsimmetrico in Eq. (1) nel dominio $(a, r) \in [0.5, 1] \times [-0.5, 0.5]$ mediante una rete neurale, avendo delle misurazioni ottenute attraverso sensori disposti su una griglia non-equispaziata lungo entrambe le dimensioni. La griglia è composta da 30 sensori per lato, le cui misurazioni sono affette da rumore gaussiano $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, con $\sigma = 3 \cdot 10^{-2}$.

   Seguendo il template del punto precedente, si alleni una rete costituita da un hidden state di dimensione $h = 20$, e funzione di attivazione `'Tanh'`, adottando i seguenti hyper-parametri: `maxit`$=2 \cdot 10^5$, dimensione mini-batch 32, parametri di decay dei momenti per l'algoritmo Adam $\beta_1 = \beta_2 = 0.8$, e learning rate `lr`$= 5 \cdot 10^{-4}$. Si visualizzi l'approssimazione e l'errore assoluto nel dominio richiesto.