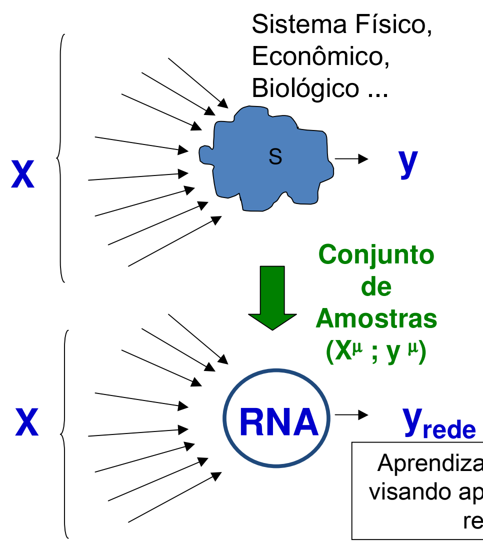
Retornando `as nossas motivaçoes inicinciais no calculo de adaptaçoes de pesos ...

Conjunto de treino em arquiteturas supervisionadas (ex. clássico: MLP com Error Back Propagation)



A computação desejada da rede pode ser definida simplesmente através de amostras / exemplos do comportamento requerido

$$Eqm = \frac{1}{M}\sum_{\mu=1}^{M}(y_{rede}(\vec{X}^{\mu},\vec{W}) - y^{\mu})^2$$

$$\Delta\vec{W} = -\eta.\vec{\nabla}Eqm$$

Aprendizado: Espaço de pesos W é explorado visando aproximar ao máximo a computação da rede da computação desejada

Transformemos nossos resultados de fórmulas de gradiente de erro quadrático de exemplar (Eq$^\mu$) e de gradiente de erro quadrático médio (Eqm) em um algoritmo / pseudo-código macroscópico:

- O Grad(Eq$^\mu$) associado a apenas um exemplo específico (X^μ;y^μ) é obtido como desenvolvido nas aulas anteriores do módulo, devemos calcular tantas derivadas parciais ∂ Eq$^\mu$ / ∂ w_xx quantos pesos w_xx existam na rede.
- Fazemos esses cálculos através, por exemplo, das técnicas de retropropagação e dos passos direto e reverso. Assim calculamos todas as componentes ∂ Eq$^\mu$ / ∂ w_xx e compomos o gradiente de Eq$^\mu$, referente a um µ específico (cada µ define um para entrada / saída de treino).
- Cumulativamente, somamos esses M gradientes de exemplar, Grad (Eq$^\mu$), para ao final da varredura de µ, chegarmos ao vetor médio Grad (Eqm)= Σ Grad (Eq$^\mu$)/M

*Usemos o cálculo do gradiente de erro quadrático médio (*Eqm*) para melhorar o valor do Eqm e o modelo neural:*

– Loop: para o cálculo de Grad (Eqm), usamos um Loop de varredura dos M exemplares de entrada / saída disponíveis para o treinamento da rede
  - Para chegarmos a Grad (Eqm), calculamos M vetores gradiente, um para cada (X^μ;y^μ). Devemos obter M vetores Grad (Eq^μ), um para cada exemplar entrada / saída (X^μ;y^μ) , varrendo pois todos os M exemplares disponíveis para treino da rede neural.
  - O Grad(Eq^μ) associado a apenas um exemplo específico (X^μ;y^μ) é obtido como desenvolvido nas aulas anteriores do módulo, devemos calcular tantas derivadas parciais ∂ Eq^μ / ∂ w_xx quantos pesos w_xx existam na rede.
  - Fazemos esses cálculos através, por exemplo, das técnicas de retropropagação e dos passos direto e reverso. Assim calculamos todas as componentes ∂ Eq^μ / ∂ w_xx e compomos o gradiente de Eq^μ, referente a um μ específico (cada μ define um para entrada / saída de treino).
  - Cumulativamente, somamos esses M gradientes de exemplar, Grad (Eq^μ), para ao final da varredura de μ, chegarmos ao vetor médio Grad (Eqm)= Σ Grad (Eq^μ)/M

– Ao sairmos do Loop em μ acima (ao finalizarmos a varredura de todos os μ) chegamos ao vetor médio, Grad(Eqm), e estamos prontos para darmos um pequeno passo vetorial ΔW no espaço de pesos, com a direção e magnitude dados por –η.Grad (Eqm).

– Assim adaptamos os pesos w's: Wnovo=Wanterior –η.Grad (Eqm)

– Com essa mudança, devemos melhorar o Eqm, pois o reduzimos, incrementalmente.

Lembremos de onde partimos: W inicial que queremos refinar

- O vetor Winicial é escolhido randomicamente: não há regiões de W melhores, no caso geral
- Tipicamente essa escolha randômica leva a um Eqm alto e portanto insatisfatório. Para melhorar o Eqm, queremos calcular Grad (Eqm) com a finalidade de adaptar W, "caminhando" contrariamente ao gradiente do erro: Wnovo = Wcorrente –η.Grad (Eqm)

– Loop: para o cálculo de Grad (Eqm), usamos um Loop de varredura dos M exemplares de entrada / saída disponíveis para o treinamento da rede:
  - Para chegarmos a Grad (Eqm), calculamos M vetores gradiente, um para cada (X^μ;y^μ). Devemos obter M vetores Grad (Eq^μ), um para cada exemplar entrada / saída (X^μ;y^μ) , varrendo pois todos os M exemplares disponíveis para treino da rede neural.
  - O Grad(Eq^μ) associado a apenas um exemplo específico (X^μ;y^μ) é obtido como desenvolvido nas aulas anteriores do módulo, devemos calcular tantas derivadas parciais ∂ Eq^μ / ∂ w_xx quantos pesos w_xx existam na rede.
  - Fazemos esses cálculos através, por exemplo, das técnicas de retropropagação e dos passos direto e reverso. Assim calculamos todas as componentes ∂ Eq^μ / ∂ w_xx e compomos o gradiente de Eq^μ, referente a um μ específico (cada μ define um para entrada / saída de treino).
  - Cumulativamente, somamos esses M gradientes de exemplar, Grad (Eq^μ), para ao final da varredura de μ, chegarmos ao vetor médio Grad (Eqm)= Σ Grad (Eq^μ)/M

– Ao sairmos do Loop em μ acima (ao finalizarmos a varredura de todos os μ) chegamos ao vetor médio, Grad(Eqm), e estamos prontos para darmos um pequeno passo vetorial ΔW no espaço de pesos, com a direção e magnitude dados por –η.Grad (Eqm).

– Assim adaptamos os pesos w's: Wnovo=Wanterior –η.Grad (Eqm)

– Com essa mudança, devemos melhorar o Eqm, pois o reduzimos, incrementalmente.

Lembremos que o processo é de refinamentos sucessivos de Eqm

- O vetor Winicial é escolhido randomicamente: não há regiões de W melhores, no caso geral
- Tipicamente essa escolha randômica leva a um Eqm alto e portanto insatisfatório. Para melhorar o Eqm, queremos calcular Grad (Eqm) com a finalidade de adaptar W, "caminhando" contrariamente ao gradiente do erro: Wnovo = Wcorrente –η.Grad (Eqm)
- Loop Externo: Loop do Gradiente Descendente:
  - Loop Interno: varredura dos M exemplares de treino. Para o cálculo de Grad (Eqm), usamos um Loop de varredura dos M exemplares de entrada / saída disponíveis para o treinamento da rede:
    - Para chegarmos a Grad (Eqm), calculamos M vetores gradiente, um para cada ($X^\mu;y^\mu$). Devemos obter M vetores Grad (Eq^μ), um para cada exemplar entrada / saída ($X^\mu;y^\mu$) , varrendo pois todos os M exemplares disponíveis para treino da rede neural.
    - O Grad(Eq^μ) associado a apenas um exemplo específico ($X^\mu;y^\mu$) é obtido como desenvolvido nas aulas anteriores do módulo, devemos calcular tantas derivadas parciais $\partial Eq^\mu / \partial w\_xx$ quantos pesos w_xx existam na rede.
    - Fazemos esses cálculos através, por exemplo, das técnicas de retropropagação e dos passos direto e reverso. Assim calculamos todas as componentes $\partial Eq^\mu / \partial w\_xx$ e compomos o gradiente de Eq^μ, referente a um μ específico (cada μ define um para entrada / saída de treino).
    - Cumulativamente, somamos esses M gradientes de exemplar, Grad (Eq^μ), para ao final da varredura de μ, chegarmos ao vetor médio Grad (Eqm)= Σ Grad (Eq^μ)/M
  - Ao sairmos do Loop em μ acima (ao finalizarmos a varredura de todos os μ) chegamos ao vetor médio, Grad(Eqm), e estamos prontos para darmos um pequeno passo vetorial ΔW no espaço de pesos, com a direção e magnitude dados por –η.Grad (Eqm).
  - Assim adaptamos os pesos w's: Wnovo=Wanterior –η.Grad (Eqm)
  - Com essa mudança, devemos melhorar o Eqm, pois o reduzimos, incrementalmente.
  - Como o gradiente descendente opera por aproximações sucessivas, temos um Loop Externo: várias adaptações de W, Wnovo = Wcorrente –η.Grad (Eqm), são necessárias em sequência.

Adicionemos critério de encerramento dos refinamentos sucessivos

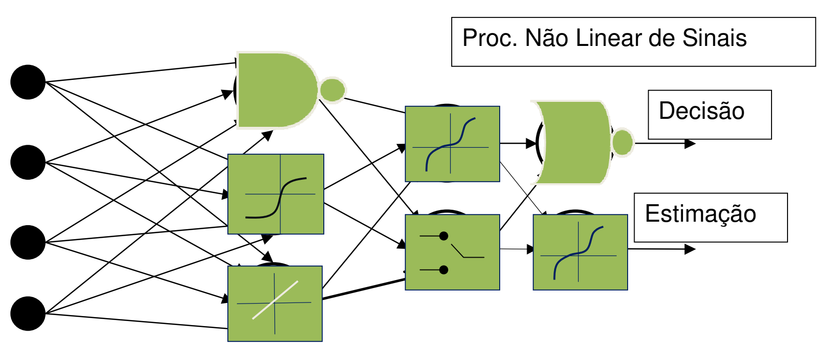
- Chutamos um vetor W inicial, escolhido randomicamente
- Tipicamente essa escolha randômica leva a um Eqm alto e portanto insatisfatório. Para melhorar o Eqm, queremos calcular Grad (Eqm) com a finalidade de adaptar W, "caminhando" contrariamente ao gradiente do erro: Wnovo = Wcorrente –η.Grad (Eqm)
- Loop Externo: Loop do Gradiente Descendente:
  - Loop Interno: varredura dos M exemplares de treino. Para o cálculo de Grad (Eqm), usamos um Loop de varredura dos M exemplares de entrada / saída disponíveis para o treinamento da rede:
    - Para chegarmos a Grad (Eqm), calculamos M vetores gradiente, um para cada ($X^\mu;y^\mu$). Devemos obter M vetores Grad (Eq^μ), um para cada exemplar entrada / saída ($X^\mu;y^\mu$) , varrendo pois todos os M exemplares disponíveis para treino da rede neural.
    - O Grad(Eq^μ) associado a apenas um exemplo específico ($X^\mu;y^\mu$) é obtido como desenvolvido nas aulas anteriores do módulo, devemos calcular tantas derivadas parciais $\partial Eq^\mu / \partial w\_xx$ quantos pesos w_xx existam na rede.
    - Fazemos esses cálculos através, por exemplo, das técnicas de retropropagação e dos passos direto e reverso. Assim calculamos todas as componentes $\partial Eq^\mu / \partial w\_xx$ e compomos o gradiente de Eq^μ, referente a um μ específico (cada μ define um para entrada / saída de treino).
    - Cumulativamente, somamos esses M gradientes de exemplar, Grad (Eq^μ), para ao final da varredura de μ, chegarmos ao vetor médio Grad (Eqm)= Σ Grad (Eq^μ)/M
  - Ao sairmos do Loop em μ acima (ao finalizarmos a varredura de todos os μ) chegamos ao vetor médio, Grad(Eqm), e estamos prontos para darmos um pequeno passo vetorial ΔW no espaço de pesos, com a direção e magnitude dados por –η.Grad (Eqm).
  - Assim adaptamos os pesos w's: Wnovo=Wanterior –η.Grad (Eqm)
  - Com essa mudança, devemos melhorar o Eqm, pois o reduzimos, incrementalmente.
  - Como o gradiente descendente opera por aproximações sucessivas, temos um Loop Externo: várias adaptações de W, Wnovo = Wcorrente –η.Grad (Eqm), são necessárias em sequência.
  - Critério de parada: seguimos no loop do Gradiente descendente até Eqm ser zero, baixo, ou estável (sem melhoras incrementais observadas), ou até estourarmos um número limíte de adaptações.

Concluindo ... Processo de aproximações sucessivas ao Eqm mínimo:

- Chutamos um vetor W inicial, escolhido randomicamente
- Tipicamente essa escolha randômica leva a um Eqm alto e portanto insatisfatório. Para melhorar o Eqm, queremos calcular Grad (Eqm) com a finalidade de adaptar W, "caminhando" contrariamente ao gradiente do erro: Wnovo = Wcorrente $-\eta$.Grad (Eqm)
- Loop Externo: Loop do Gradiente Descendente
  - Loop Interno: varredura dos M exemplares de treino. Para o cálculo de Grad (Eqm), usamos um Loop de varredura dos M exemplares de entrada / saída disponíveis para o treinamento da rede:
    - Para chegarmos a Grad (Eqm), calculamos M vetores gradiente, um para cada (X^{μ};y^{μ}). Devemos obter M vetores Grad (Eq^{μ}), um para cada exemplar entrada / saída (X^{μ};y^{μ}) , varrendo pois todos os M exemplares disponíveis para treino da rede neural.
    - O Grad(Eq^{μ}) associado a apenas um exemplo específico (X^{μ};y^{μ}) é obtido como desenvolvido nas aulas anteriores do módulo, devemos calcular tantas derivadas parciais ∂ Eq^{μ} / ∂ w_xx quantos pesos w_xx existam na rede.
    - Fazemos esses cálculos através, por exemplo, das técnicas de retropropagação e dos passos direto e reverso. Assim calculamos todas as componentes ∂ Eq^{μ} / ∂ w_xx e compomos o gradiente de Eq^{μ}, referente a um μ específico (cada μ define um para entrada / saída de treino).
    - Cumulativamente, somamos esses M gradientes de exemplar, Grad (Eq^{μ}), para ao final da varredura de μ, chegarmos ao vetor médio Grad (Eqm)= Σ Grad (Eq^{μ})/M
  - Ao sairmos do Loop em μ acima (ao finalizarmos a varredura de todos os μ) chegamos ao vetor médio, Grad(Eqm), e estamos prontos para darmos um pequeno passo vetorial ΔW no espaço de pesos, com a direção e magnitude dados por $-\eta$.Grad (Eqm).
  - Assim adaptamos os pesos w's: Wnovo=Wanterior $-\eta$.Grad (Eqm)
  - Com essa mudança, devemos melhorar o Eqm, pois o reduzimos, incrementalmente.
  - Como o gradiente descendente opera por aproximações sucessivas, temos um Loop Externo: várias adaptações de W, Wnovo = Wcorrente $-\eta$.Grad (Eqm), são necessárias em sequência.
  - Critério de parada: seguimos no loop do Gradiente descendente até Eqm ser zero, baixo, ou estável (sem melhoras incrementais observadas), ou até estourarmos um número limíte de adaptações.
- Ao sairmos do Loop Externo, temos "bons w´s" (que levam a erros Eqm baixos) e podemos usar o modelo neural para estimar novos valores de y, com conhecimento apenas dos x's.

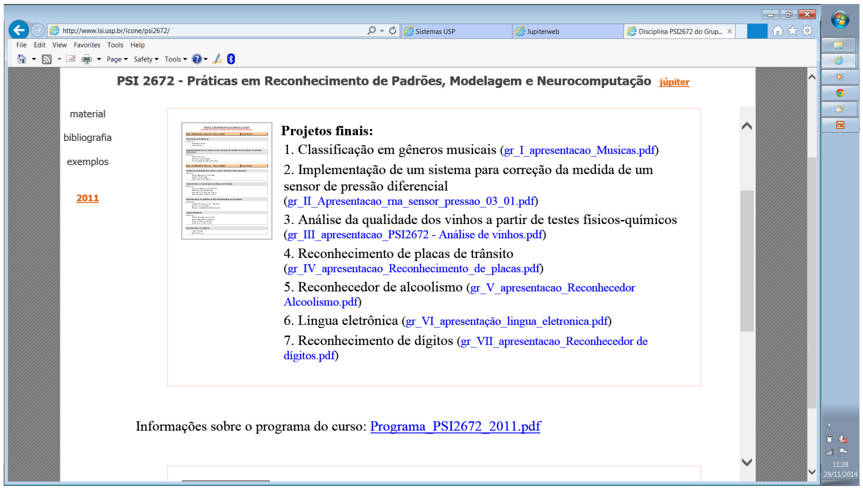
O Multi Layer Perceptron (MLP)

- Múltiplas entradas / Múltiplas saídas / Múltiplas camadas
- Variáveis (internas e externas) analógicas ou digitais
- Relações lineares ou não lineares entre elas



Kolmogorov, Cybenko (~1990)

Alguns exemplos de projetos concebidos e realizados por alunos da disciplina PSI-2672



Estamos perto do fim de PSI 2533!

- *Lembrete importante: no semestre que vem temos a optativa PSI 2672 – Práticas em Reconhecimento de Padrões, Modelagem e Neurocomputação, que é uma oportunidade interessante de aprofundar alguns dos conhecimentos vistos em PSI 2533 e de usá-los em projetos práticos concretos concebidos e implementados pelos alunos.*

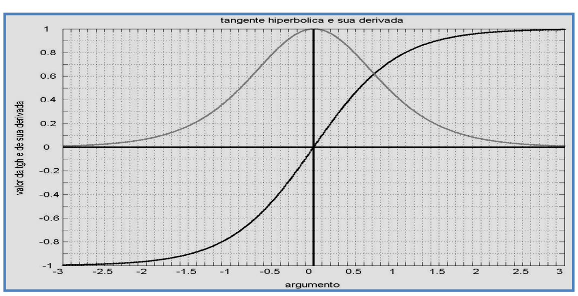
Matricule-se nessa disciplina já ☺ !

+ *Exercícios de treino sugeridos*

- Deduções de equações ... Treino para a prova
- Exercícios numéricos sobre as equações: para treinar para a prova ... (inclue o uso dos gráficos da tgh e da sua derivada)
- Exercícios com o MBP para ver o uso da teoria em casos práticos ....

Tangente Hiperbólica e sua Derivada

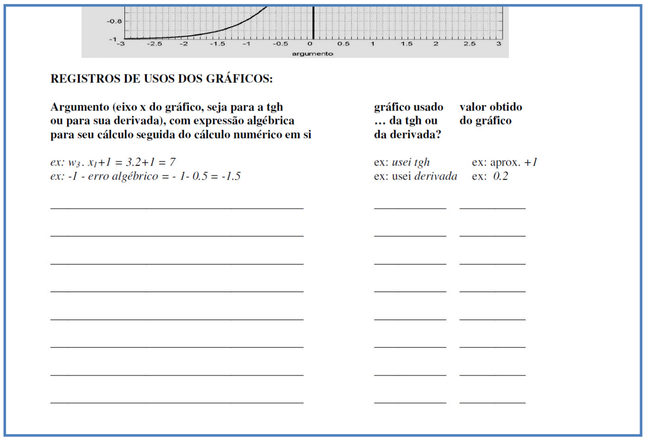
- Estes gráficos serão fornecidos e usados em prova na questão de exercício numérico



Prof. Emilio Del Moral – PSI2533 – EPUSP - direitos reservados consulte-me antes de qualquer uso externo a 2533

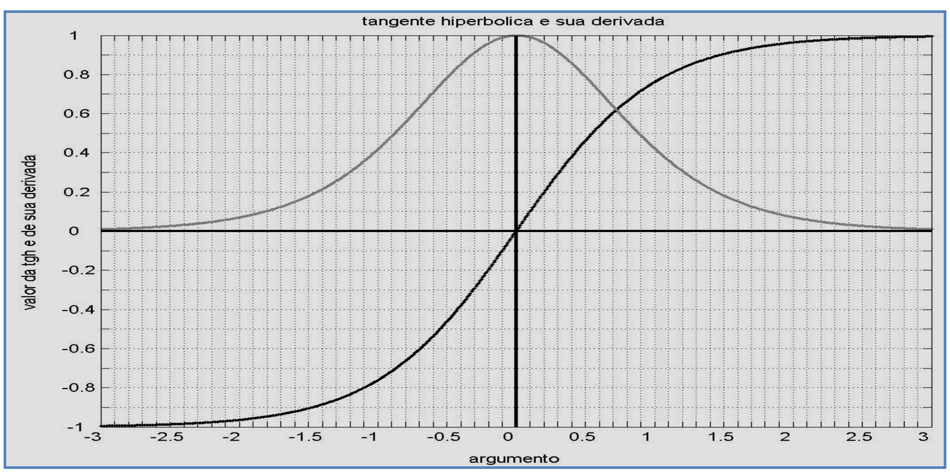
Tabela de registro de cálculos

- Esta tabela (ou similar) será fornecida e usada em prova na questão de exercício numérico



REGISTROS DE USOS DOS GRÁFICOS:

Argumento (eixo x do gráfico, seja para a tgh ou para sua derivada), com expressão algébrica para seu cálculo seguida do cálculo numérico em si	**gráfico usado … da tgh ou da derivada?**	**valor obtido do gráfico**
ex: $w_3 . x_1+1 = 3.2+1 = 7$	ex: *usei tgh*	ex: aprox. *+1*
ex: -1 - erro algébrico = - 1- 0.5 = -1.5	ex: usei *derivada*	ex: *0.2*



+ Exercícios de treino sugeridos

<u>Exercícios de treino para PSI 2533-2011 – Prof. Emilio:</u>

1) Para uma rede MLP com 2 entradas, 2 nós na primeira camada, 2 nós na segunda e 1 nó na terceira (de saída), com entrada adicional de polarização (bias / viés / "1") nos diversos nós, como usual, deduza as fórmulas de todos os retro-propagadores pertinentes. Sugestão: desenhe (e apresente) o grafo de cálculos da rede em sua operação direta e identifique as variáveis envolvidas nesse grafo, pois isto lhe facilitará chegar na solução correta.

2) Para essa mesma MLP, sorteie (por exemplo usando uma função "random" da sua linguagem preferida) valores de entradas x na faixa [-1 a +1], valores de y na faixa [-1 a +1] e valores de pesos na faixa [-1/2 a +1/2] (este fator ½ levará a valores mais interessantes para este exercício), prepare uma tabela adequada com tais sorteios e calcule todos os valores correspondentes de Vs e Ys em cada camada, bem como y_rede e erro_rede.

2b) Refaça os cálculos dos Vs e dos Ys agora sem usar a calculadora: faça somas e produtos de forma aproximada e use os gráficos da tangente hiperbólica e de sua derivada conforme necessário, praticando o procedimento que usaremos em parte da prova (procedimento sem calculadora).

3) Usando as fórmulas do item 1 e os resultados do item 2, calcule para alguns dos valores de erros de rede, os valores dos erros retro-propagados a cada nó.

3b) Refaça os cálculos agora sem usar a calculadora: faça somas e produtos de forma aproximada e use os gráficos da tangente hiperbólica e de sua derivada conforme necessário, exatamente como faremos na prova (sem calculadora).

4) Usando a regra delta de nó simples, use os resultados do ex 3 para calcular a adaptação deltaW de cada nó.

4b) Refaça os cálculos agora sem usar a calculadora: faça somas e produtos de forma aproximada e use os gráficos da tangente hiperbólica e de sua derivada conforme necessário, exatamente como faremos na prova (sem calculadora).

5) Deduza as fórmulas de retro-propagação dos erros de 2ª camada (com 2 nós) para os erros da 1ª camada (com 2 nós), a partir das expressões de retro-propagação de cada um dos 4 nós envolvidos (obtidas no item 1).

6) Uso do software MBP: Agora com uma rede de apenas duas camadas, com três nós com tangente hiperbólica na primeira camada e um nó linear segunda camada (= camada de saída) use o MBP para aproximar com esse modelo neural a função y=x1*[(x2)^2], com as devidas normalizações para que as faixas de entrada e saída estejam limitadas a [-1 a +1]. <u>Capture os gráficos de evolução do erro durante o aprendizado e comente</u>.

6b) Depois, aumentando o número de nós na primeira camada, avalie como isso melhora o valor final do erro quadrático da rede. <u>Capture os gráficos de evolução do erro durante o aprendizado e comente</u>.

6c) Defina outra função alvo de sua escolha, para ser aproximada pela rede e repita o Ex 6 e Ex 6b). <u>Capture os gráficos de evolução do erro durante o aprendizado e comente</u>.