IN5550: Neural Methods in Natural Language Processing
Sub-lecture 2.3
Training as optimization

Andrey Kutuzov

University of Oslo

31 January 2023

Contents

Training as optimization

- The goal of the training is to find the optimal values of parameters in θ.

Training as optimization

- The goal of the training is to find the optimal values of parameters in θ.
- Formally, it means to minimize the loss $\mathcal{L}(\theta)$ on training or development dataset.

Training as optimization

- The goal of the training is to find the optimal values of parameters in θ.
- Formally, it means to minimize the loss $\mathcal{L}(\theta)$ on training or development dataset.
- Conceptually, loss is a measure of how 'far away' the model predictions $\hat{\boldsymbol{y}}$ are from gold labels $\boldsymbol{y}$.

Training as optimization

- The goal of the training is to find the optimal values of parameters in θ.
- Formally, it means to minimize the loss $\mathcal{L}(\theta)$ on training or development dataset.
- Conceptually, loss is a measure of how 'far away' the model predictions $\hat{\boldsymbol{y}}$ are from gold labels $\boldsymbol{y}$.
- It can be any function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ returning a scalar value:
  - for example, $\mathcal{L} = (\boldsymbol{y} - \hat{\boldsymbol{y}})^2$ (square error)

Training as optimization

- The goal of the training is to find the optimal values of parameters in θ.
- Formally, it means to minimize the loss $\mathcal{L}(\theta)$ on training or development dataset.
- Conceptually, loss is a measure of how 'far away' the model predictions $\hat{\boldsymbol{y}}$ are from gold labels $\boldsymbol{y}$.
- It can be any function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ returning a scalar value:
  - for example, $\mathcal{L} = (\boldsymbol{y} - \hat{\boldsymbol{y}})^2$ (square error)
- It is averaged over all training instances and gives us estimation of the model 'fitness'.

Training as optimization

- The goal of the training is to find the optimal values of parameters in θ.
- Formally, it means to minimize the loss $\mathcal{L}(\theta)$ on training or development dataset.
- Conceptually, loss is a measure of how 'far away' the model predictions $\hat{\boldsymbol{y}}$ are from gold labels $\boldsymbol{y}$.
- It can be any function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ returning a scalar value:
  - for example, $\mathcal{L} = (\boldsymbol{y} - \hat{\boldsymbol{y}})^2$ (square error)
- It is averaged over all training instances and gives us estimation of the model 'fitness'.
- $\hat{\theta}$ is the best set of parameters:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) \tag{1}$$

Training as optimization

What is the intuition behind a loss function?

Training as optimization

What is the intuition behind a loss function?

- Let's take a simple example

Training as optimization

What is the intuition behind a loss function?

- Let's take a simple example
- Predicting the price of a home from its size

Training as optimization

What is the intuition behind a loss function?

- Let's take a simple example
- Predicting the price of a home from its size
- What kind of problem is this?

What is the intuition behind a loss function?

- Let's take a simple example
- Predicting the price of a home from its size
- What kind of problem is this? Regression.

Training as optimization

What is the intuition behind a loss function?

- Let's take a simple example
- Predicting the price of a home from its size
- What kind of problem is this? Regression.
- WHITEBOARD EXAMPLE

Predicting the price of a home from its size

Training as optimization

What are the main classes of NLP tasks?

Training as optimization

What are the main classes of NLP tasks?

- regression

Training as optimization

What are the main classes of NLP tasks?

- regression
- classification

Training as optimization

What are the main classes of NLP tasks?

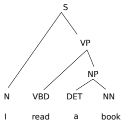
- regression
- classification
- ranking

Training as optimization

What are the main classes of NLP tasks?

- regression
- classification
- ranking
- structured prediction

Training as optimization

Given x,	predict y	
A sentence That's great!!!!!	**How positive is it** (from 1-10) 9.5	**regression** (scalar)
A sentence That's great!!!!!	**Pos. or Neg.?** Positive	**binary classification** (two choices)
A tweet That was fun. Vale la pena!	**Which langs.?** English Spanish	**multi-class classification** (many choices)
A question What is the most expensive spice in the world?	**A ranked list of searches** 1) Saffron-is-expensive 2) Truffles-are-crazy	**ranking** (ranked list of available choices)
A sentence I read a book	**Its syntactic parse**  	**structured prediction** (millions of choices)

Training as optimization

How do you choose a loss function?

Training as optimization

How do you choose a loss function?

1. It depends on your task...

Training as optimization

How do you choose a loss function?

1. It depends on your task...
   - regression

Training as optimization

How do you choose a loss function?

1. It depends on your task...
    - regression - mean absolute error, mean squared error, ...

Training as optimization

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification

Training as optimization

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification - hinge-loss, cross-entropy, ...

Training as optimization

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification - hinge-loss, cross-entropy, ...
   - ranking

Training as optimization

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification - hinge-loss, cross-entropy, ...
   - ranking - ranking loss, triplet loss

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification - hinge-loss, cross-entropy, ...
   - ranking - ranking loss, triplet loss
2. A mix of theoretical and practical desires often determine your final choice
   - For classification, we often use some variant of...

Training as optimization

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification - hinge-loss, cross-entropy, ...
   - ranking - ranking loss, triplet loss
2. A mix of theoretical and practical desires often determine your final choice
   - For classification, we often use some variant of...
   - ... hinge-loss (max margin)

How do you choose a loss function?

1. It depends on your task...
   - regression - mean absolute error, mean squared error, ...
   - classification - hinge-loss, cross-entropy, ...
   - ranking - ranking loss, triplet loss
2. A mix of theoretical and practical desires often determine your final choice
   - For classification, we often use some variant of...
   - ... hinge-loss (max margin)
   - ... cross-entropy loss

Training as optimization

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$

Training as optimization

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$

Training as optimization

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
4. Binary cross-entropy (logistic loss):
   $L(\hat{y}, y) = -y \ log\hat{y} - (1 - y)log(1 - \hat{y})$

Training as optimization

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
4. Binary cross-entropy (logistic loss):
   $L(\hat{y}, y) = -y \ log\hat{y} - (1 - y)log(1 - \hat{y})$
5. Categorical cross-entropy (negative log-likelihood):
   $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_i \boldsymbol{y}_{[i]} log(\hat{\boldsymbol{y}}_{[i]})$

Training as optimization

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
4. Binary cross-entropy (logistic loss):
   $L(\hat{y}, y) = -y \ log \hat{y} - (1 - y) log(1 - \hat{y})$
5. Categorical cross-entropy (negative log-likelihood):
   $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_i \boldsymbol{y}_{[i]} log(\hat{\boldsymbol{y}}_{[i]})$
6. Ranking losses, etc, etc...

Training as optimization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.

Training as optimization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

Training as optimization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

  Why do you think this is? Pause the video and think.

Training as optimization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

  Why do you think this is? Pause the video and think.
- We can live with some errors on the training data, if it gives more generalization power.

Training as optimization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

    Why do you think this is? Pause the video and think.
- We can live with some errors on the training data, if it gives more generalization power.
- For that, we minimize both the loss and the regularization term $R(\theta)$:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) + \lambda R(\theta) \tag{2}$$

- The hyperparameter λ is regularization weight (how important is it).

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

  Why do you think this is? Pause the video and think.
- We can live with some errors on the training data, if it gives more generalization power.
- For that, we minimize both the loss and the regularization term $R(\theta)$:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) + \lambda R(\theta) \tag{2}$$

- The hyperparameter λ is regularization weight (how important is it).
- Common regularization terms:
  1. L_2 norm (*Gaussian prior* or *weight decay*);
  2. L_1 norm (*sparse prior* or *lasso*)

Training as optimization

Now we can measure model performance. How can we change our parameters θ to improve?

1. We could just randomly change some of the parameters and see if the result is better (hill climbing algorithm)...

Training as optimization

1. We could just randomly change some of the parameters and see if the result is better (hill climbing algorithm)...
2. or we could be smarter about it (gradient-based methods).

Training as optimization

Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.

Training as optimization

Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.
- Commonly solved using gradient methods:
    1. compute the loss,

Training as optimization

Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.
- Commonly solved using gradient methods:
  1. compute the loss,
  2. compute gradient of θ parameters with respect to the loss,

Training as optimization

Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.
- Commonly solved using gradient methods:
    1. compute the loss,
    2. compute gradient of θ parameters with respect to the loss,
    3. *(gradient here is the collection of partial derivatives, one for each parameter of θ)*

Training as optimization

Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.
- Commonly solved using gradient methods:
  1. compute the loss,
  2. compute gradient of θ parameters with respect to the loss,
  3. *(gradient here is the collection of partial derivatives, one for each parameter of θ)*
  4. move the parameters in the opposite direction (to decrease the loss),

Training as optimization

Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.
- Commonly solved using gradient methods:
  1. compute the loss,
  2. compute gradient of θ parameters with respect to the loss,
  3. *(gradient here is the collection of partial derivatives, one for each parameter of θ)*
  4. move the parameters in the opposite direction (to decrease the loss),
  5. repeat until the optimum is found (the derivative is 0) or until the pre-defined number of iterations (epochs) is achieved.

Training as optimization

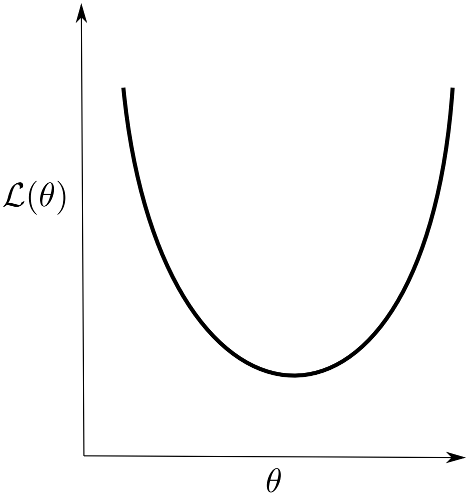
Optimizing with gradient

- $\hat{\theta} = \arg\min_{\theta}(\mathcal{L}(\theta) + \lambda R(\theta))$ is an optimization problem.
- Commonly solved using gradient methods:
  1. compute the loss,
  2. compute gradient of θ parameters with respect to the loss,
  3. *(gradient here is the collection of partial derivatives, one for each parameter of θ)*
  4. move the parameters in the opposite direction (to decrease the loss),
  5. repeat until the optimum is found (the derivative is 0) or until the pre-defined number of iterations (epochs) is achieved.

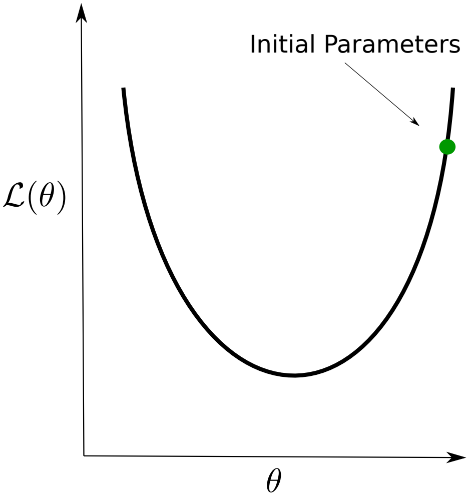
Convexity

- Convex functions: a single optimum point.
- Non-convex functions: multiple optimum points.

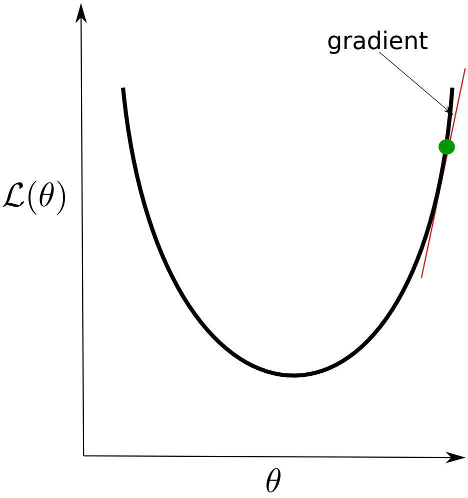
Training as optimization



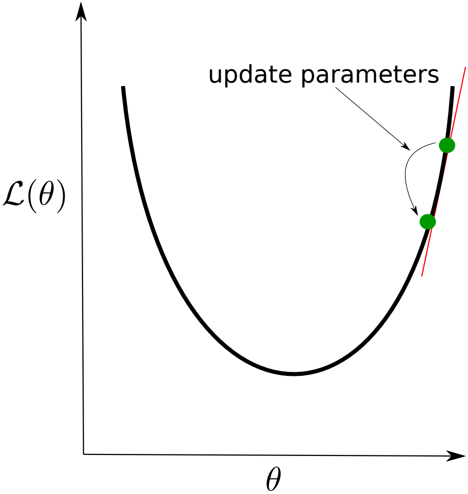
Training as optimization



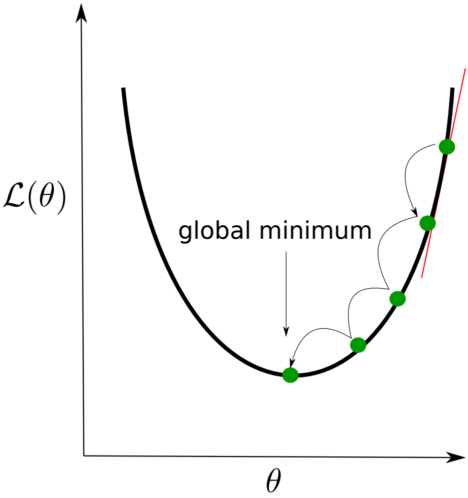
Training as optimization



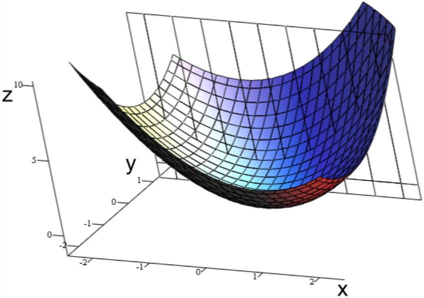
Training as optimization



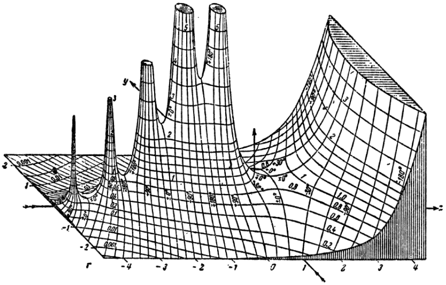
Training as optimization



Error surfaces of convex and not-convex functions:



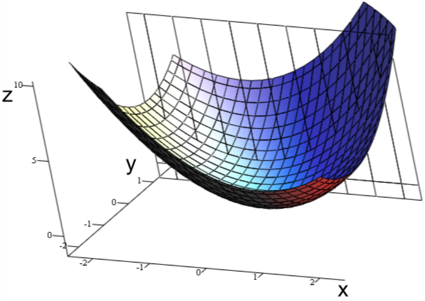
Convex function



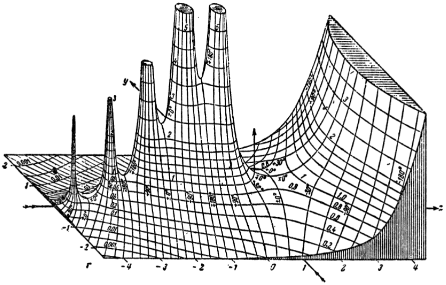
Non-convex function

Training as optimization

Error surfaces of convex and not-convex functions:



Convex function

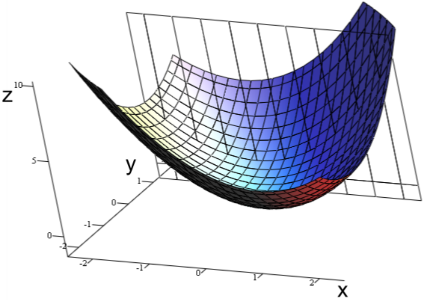


Non-convex function

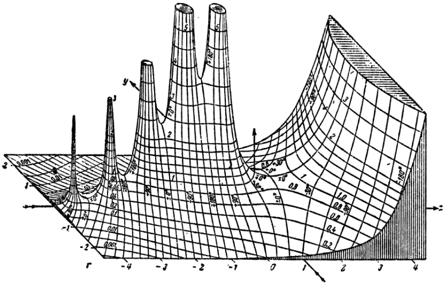
- Convex functions can be easily minimized with gradient methods, reaching the global optimum.
- With non-convex functions, optimization can end up in a local optimum.

Training as optimization

Error surfaces of convex and not-convex functions:



Convex function



Non-convex function

- Convex functions can be easily minimized with gradient methods, reaching the global optimum.
- With non-convex functions, optimization can end up in a local optimum.
- Linear and log-linear models as a rule have convex error functions.

Training as optimization

Stochastic gradient descent (SGD)

Training as optimization

Stochastic gradient descent (SGD)

- SGD samples one instance from the training set and computes the error of the gradient on it,

Training as optimization

Stochastic gradient descent (SGD)

- SGD samples one instance from the training set and computes the error of the gradient on it,
- then θ is updated in the opposite direction,

Training as optimization

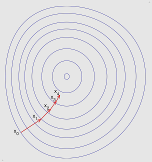
Stochastic gradient descent (SGD)

- SGD samples one instance from the training set and computes the error of the gradient on it,
- then θ is updated in the opposite direction,
- the update is scaled by the learning rate n (can be decaying over the training time),

Training as optimization

Stochastic gradient descent (SGD)

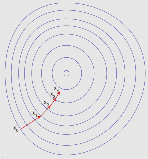
- SGD samples one instance from the training set and computes the error of the gradient on it,
- then θ is updated in the opposite direction,
- the update is scaled by the learning rate n (can be decaying over the training time),
- repeat until convergence.



Training as optimization

Stochastic gradient descent (SGD)

- SGD samples one instance from the training set and computes the error of the gradient on it,
- then θ is updated in the opposite direction,
- the update is scaled by the learning rate n (can be decaying over the training time),
- repeat until convergence.



Instead of one instance, batches can be used (more stable and computationally efficient).

Training as optimization

Other gradient-based optimizers:

- Momentum
- AdaGrad
- RMSProp
- Adam
- AdamW
- etc...

All implemented in the libraries we are going to use: *PyTorch*, *Scikit-Learn*, *TensorFlow*, *Keras*, etc.

Training as optimization

Other gradient-based optimizers:

- Momentum
- AdaGrad
- RMSProp
- Adam
- AdamW
- etc...

All implemented in the libraries we are going to use: *PyTorch*, *Scikit-Learn*, *TensorFlow*, *Keras*, etc.

So far so good. But what are the limitations of linear models? See the next sub-lecture 2.4!