Deep Learning & Applied AI

Going nonlinear, overfitting, and regularization

Emanuele Rodolà
rodola@di.uniroma1.it

2nd semester a.y. 2020/2021 · March 18, 2021

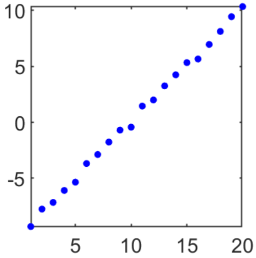
A glimpse into neural networks

In deep learning, we deal with highly parametrized models called deep neural networks:

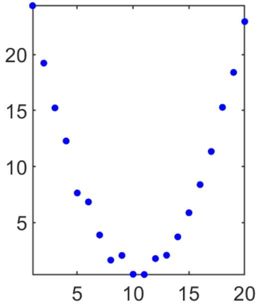
$$f_{\boldsymbol{\Theta}}(\mathbf{x}) = \mathbf{y}$$

Parametrized models

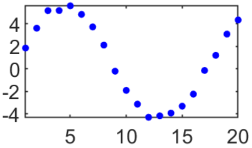
The parameters describe the behavior of the network, and must be solved for.



$$y = ax + b$$



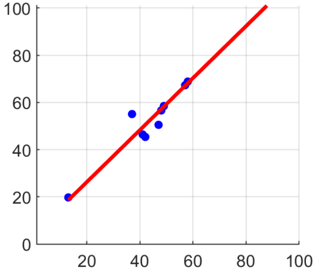
$$y = ax^2 + bx + c$$



$$y = a\sin(x) + bx + c$$

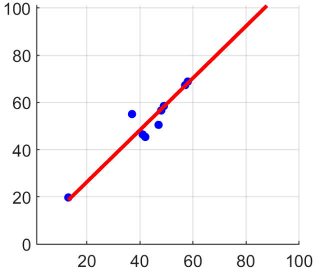
From a technical standpoint, our task is to determine the parameters Θ.

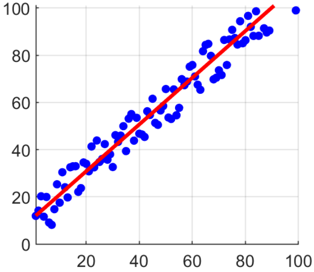
Data distribution



Assumption: linear model

Data distribution

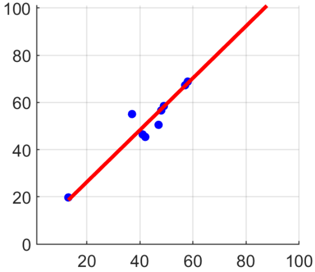


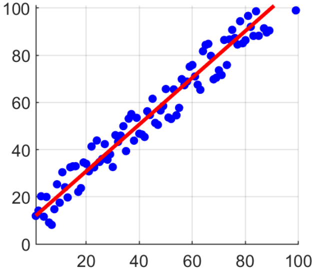


Assumption: linear model

More data allows us to improve our prediction

Data distribution



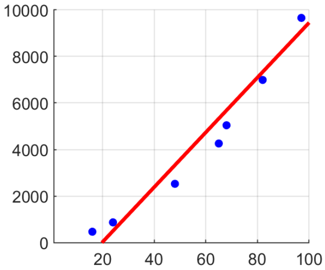


Assumption: linear model

More data allows us to improve our prediction

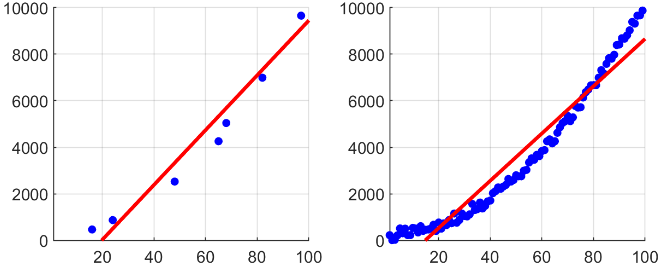
What if the assumption (i.e. linear prior here) is **wrong**?

Data distribution



Assumption: linear model

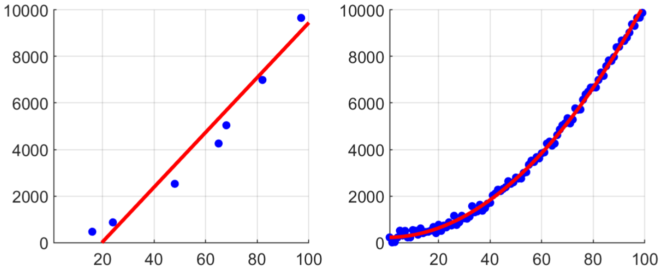
Data distribution



Assumption: linear model

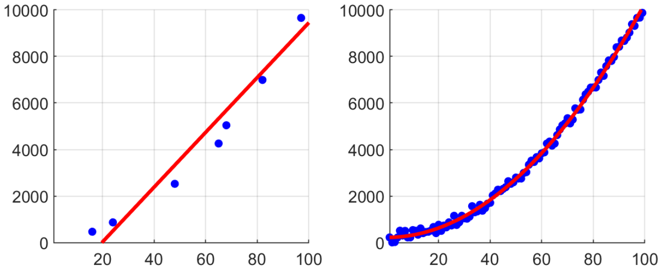
More data confutes our assumptions

Data distribution



Assumption: quadratic model

Data distribution

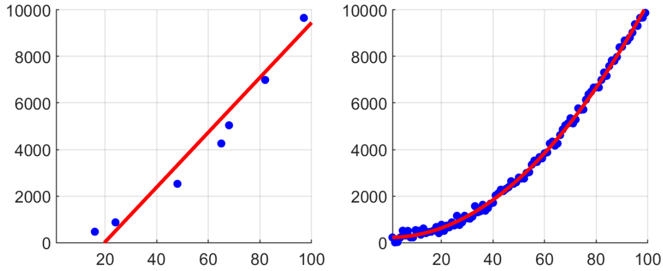


Assumption: quadratic model

Key questions:

- How to select the correct distribution?

Data distribution

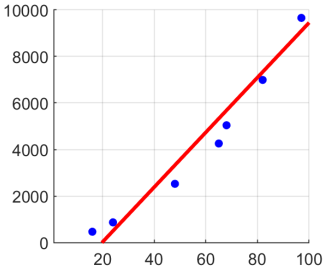


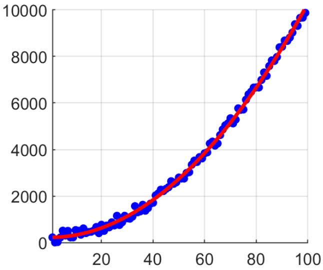
Assumption: quadratic model

Key questions:

- How to select the correct distribution?
- How much data do we need?

Data distribution





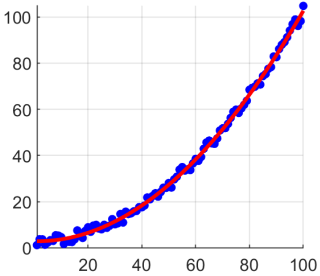
Assumption: quadratic model

Key questions:

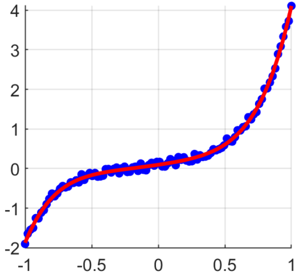
- How to select the correct distribution?
- How much data do we need?
- What if the correct distribution does not admit a simple expression?

Polynomial regression

After the linear model, the simplest thing is a polynomial model.



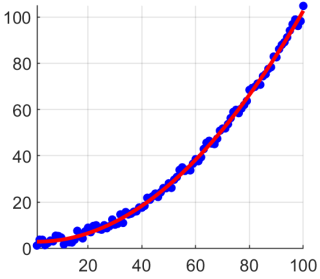
2nd-order



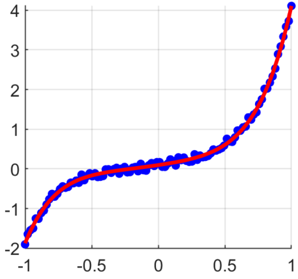
5th-order

Polynomial regression

After the linear model, the simplest thing is a polynomial model.



2nd-order

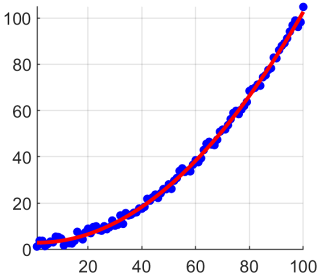


5th-order

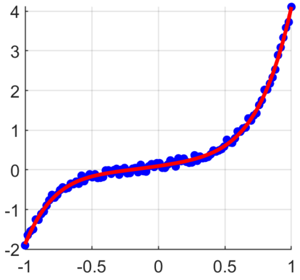
The number of parameters grows with the order.

Polynomial regression

After the linear model, the simplest thing is a polynomial model.



2nd-order



5th-order

The number of parameters grows with the order.

More data are needed to make an informed decision on the order.

Polynomial regression

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = a_3 x_i^3 + a_2 x_i^2 + a_1 x_i + b \quad \text{for all data points } i = 1, \ldots, n$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

In matrix notation:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1^k & x_1^{k-1} & \cdots & x_1 & 1 \\ x_2^k & x_2^{k-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^k & x_n^{k-1} & \cdots & x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a_k \\ a_{k-1} \\ \vdots \\ a_1 \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

Linear regression with polynomial features

Remark: Despite the name, polynomial regression is still linear in the parameters. It is polynomial with respect to the data.

$$y_i = b + \sum_{j=1}^{k} a_j x_i^j \quad \text{for all data points } i = 1, \ldots, n$$

In matrix notation:

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1^k & x_1^{k-1} & \cdots & x_1 & 1 \\ x_2^k & x_2^{k-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^k & x_n^{k-1} & \cdots & x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a_k \\ a_{k-1} \\ \vdots \\ a_1 \\ b \end{pmatrix}}_{\boldsymbol{\theta}}$$

The same exact least-squares solution as with linear regression applies, with the requirement that $k < n$.

Polynomial fitting

An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

Polynomial fitting

An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

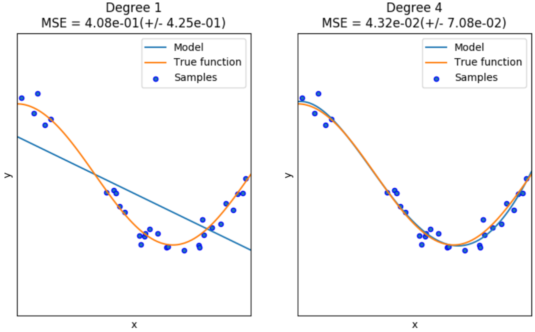
Thus, we can try to fit a polynomial in many cases.

Polynomial fitting

An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

Thus, we can try to fit a polynomial in many cases.

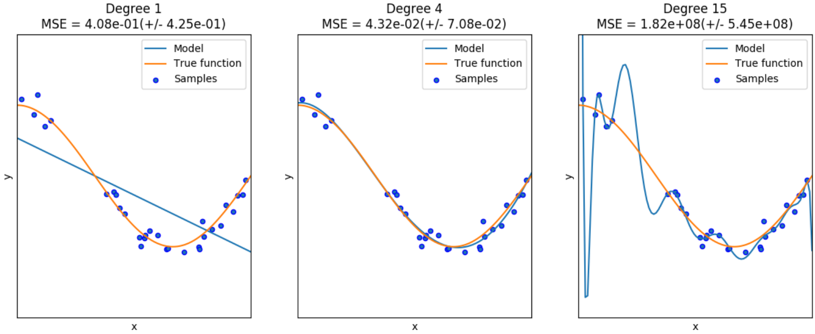


Polynomial fitting

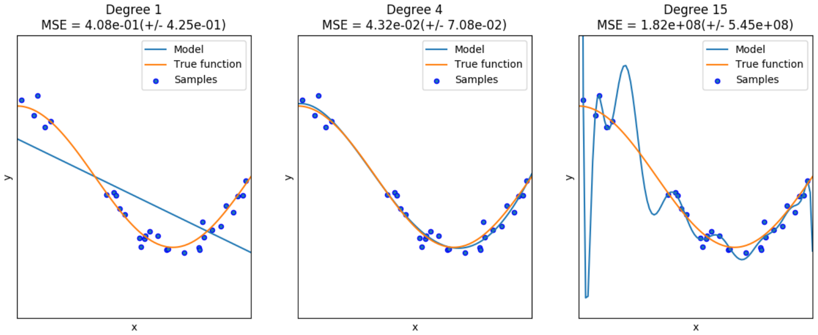
An application of the Stone-Weierstrass theorem tells us:

If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x.

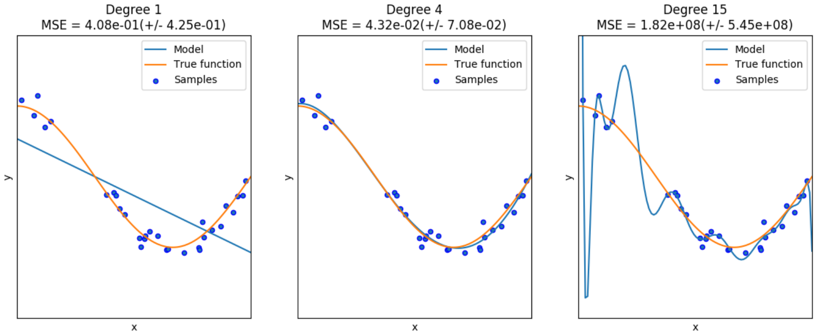
Thus, we can try to fit a polynomial in many cases.



Underfitting vs. Overfitting

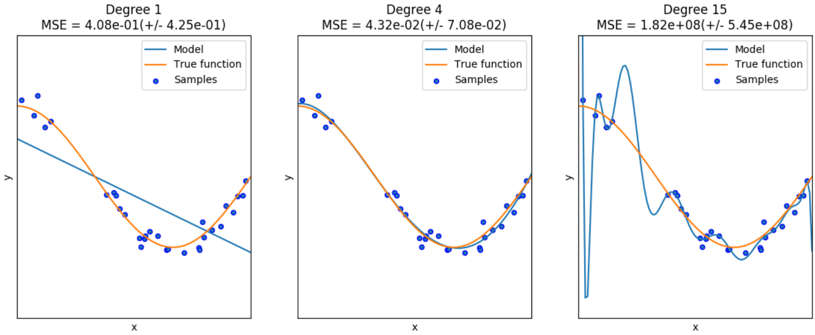


Underfitting vs. Overfitting



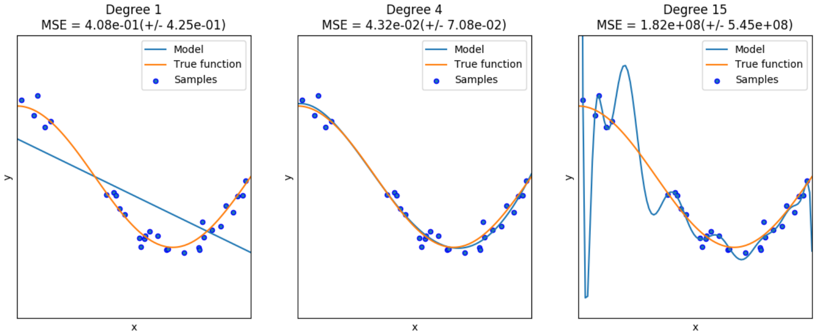
- Underfitting: Not sufficiently fitting the data (large MSE).

Underfitting vs. Overfitting



- Underfitting: Not sufficiently fitting the data (large MSE).
- Overfitting: We are "learning the noise" (small MSE).

Underfitting vs. Overfitting



- Underfitting: Not sufficiently fitting the data (large MSE).
- Overfitting: We are "learning the noise" (small MSE).

Adding complexity can lead to overfitting and thus worse generalization.

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting
3. Small MSE on the training $\Rightarrow$
   Apply the model parameters to a validation set.
   (the MSE is computed on different example data)

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting
3. Small MSE on the training $\Rightarrow$
   Apply the model parameters to a validation set.
   (the MSE is computed on different example data)
4. Large MSE on the validation $\Rightarrow$ overfitting

Underfitting vs. Overfitting

This trade-off is always present, and still an open problem.

Different mechanisms defend us from under- and overfitting.

Detection is relatively easier:

1. Estimate the model parameters on a training set.
   (the MSE is minimized on example data)
2. Large MSE on the training $\Rightarrow$ underfitting
3. Small MSE on the training $\Rightarrow$
   Apply the model parameters to a validation set.
   (the MSE is computed on different example data)
4. Large MSE on the validation $\Rightarrow$ overfitting $\Rightarrow$ bad generalization

Underfitting:
large training error, large validation error

Underfitting vs. Overfitting

Underfitting:
large training error, large validation error

Overfitting:
(very) small training error, large validation error

n-fold cross-validation

We do not want to overfit on the validation set either!

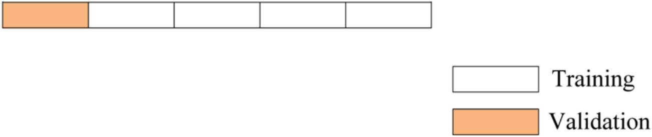
n-fold cross-validation

We do not want to overfit on the validation set either!

- Split the training set into n subsets (folds)

n-fold cross-validation

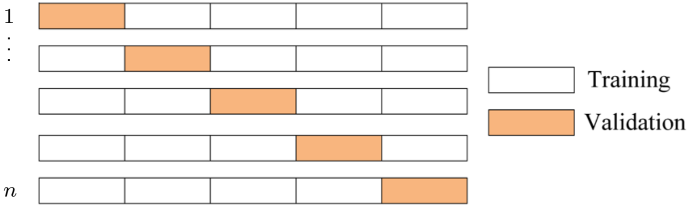
We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1

n-fold cross-validation

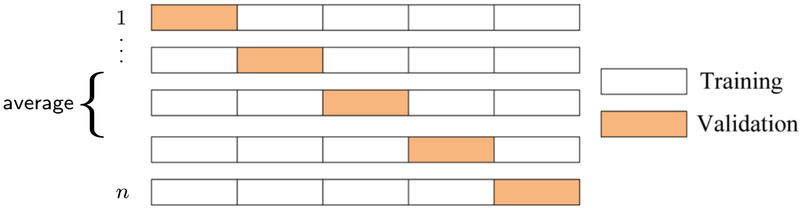
We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1
- Do this n times

n-fold cross-validation

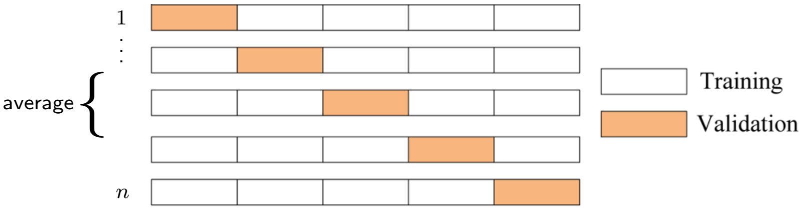
We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1
- Do this n times
- Average the score over the n folds

n-fold cross-validation

We do not want to overfit on the validation set either!



- Split the training set into n subsets (folds)
- Train on $n-1$ subsets and validate on the remaining 1
- Do this n times
- Average the score over the n folds

Example: For polynomial regression, do the above many times with different degrees, choose the run with the smallest average MSE.

Not done yet

"If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x."

So is polynomial regression all we need?

Not done yet

"If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x."

So is polynomial regression all we need?

- Different loss than MSE
- Regularization
- Additional priors
- Intermediate features
- Flexibility
- Regression (predict a value) vs. classification (predict a category)

Not done yet

"If f is continuous on the interval $[a, b]$, then for every $\epsilon > 0$ there exists a polynomial p such that $|f(x) - p(x)| < \epsilon$ for all x."

So is polynomial regression all we need?

- Different loss than MSE
- Regularization
- Additional priors
- Intermediate features
- Flexibility
- Regression (predict a value) vs. classification (predict a category)

From now on, we embrace the idea that many natural phenomena of interest are nonlinear.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting and thus control the complexity of our learning model:

$$\min_{\mathbf{\Theta}} \underbrace{\ell_{\mathbf{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting and thus control the complexity of our learning model:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting and thus control the complexity of our learning model:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

$$\min_{\boldsymbol{\Theta}} \ell_{\boldsymbol{\Theta}} + \lambda \|\Theta\|_p$$

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting and thus control the complexity of our learning model:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

$$\min_{\boldsymbol{\Theta}} \ell_{\boldsymbol{\Theta}} + \lambda \|\Theta\|_p$$

Controlling parameter growth is generally known as shrinkage.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting and thus control the complexity of our learning model:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{convex}} + \lambda \underbrace{\|\Theta\|_p}_{\text{convex}}$$

Controlling parameter growth is generally known as shrinkage.

Regularization penalties

Sometimes our prior knowledge can be expressed in terms of an energy.

For example, avoid large parameters to counteract overfitting and thus control the complexity of our learning model:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}}}_{\text{data term}} + \underbrace{\lambda}_{\text{trade-off}} \cdot \underbrace{\|\Theta\|_F^2}_{\text{regularizer}}$$

Adding a quadratic penalty to the loss is also known as weight decay, ridge, or Tikhonov regularization.

More in general:

$$\min_{\boldsymbol{\Theta}} \underbrace{\ell_{\boldsymbol{\Theta}} + \lambda\|\Theta\|_p}_{\text{convex}}$$

Controlling parameter growth is generally known as shrinkage.

Regularization

- Impose a prior on the expected behavior
- Control the model complexity
- Reduce the need for lots of data

Regularization

- Impose a prior on the expected behavior
- Control the model complexity
- Reduce the need for lots of data

But regularizers are not always written as penalties.

Regularization

- Impose a prior on the expected behavior
- Control the model complexity
- Reduce the need for lots of data

But regularizers are not always written as penalties.

Any modification that is intended to reduce the generalization error but not the training error.

Goodfellow et al., "Deep Learning", MIT Press 2016

Regularization

- Impose a prior on the expected behavior
- Control the model complexity
- Reduce the need for lots of data

But regularizers are not always written as penalties.

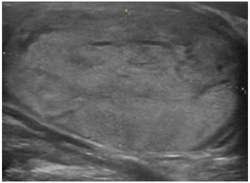
Any modification that is intended to reduce the generalization error but not the training error.

Other forms include the choice of a representation, early stopping, dropout, etc. (we'll see them in the future lectures)

Goodfellow et al., "Deep Learning", MIT Press 2016

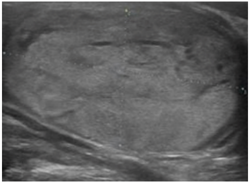
Classification

What if we want to predict a category instead of a value?

$f_\Theta($  $) = \{0, 1\}$

Classification

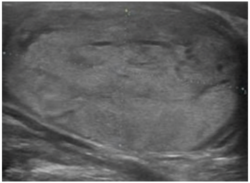
What if we want to predict a category instead of a value?

$$f_{\Theta}($$

$$) = \{0, 1\}$$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

Classification

What if we want to predict a category instead of a value?

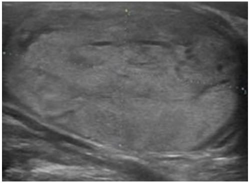
$f_{\Theta}($  $) = \{0, 1\}$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

$\Rightarrow$ The solution is not necessarily an optimum anymore.

Classification

What if we want to predict a category instead of a value?

$$f_\Theta($$  $$) = \{0, 1\}$$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

$\Rightarrow$ The solution is not necessarily an optimum anymore.

Instead: Modify the loss to minimize over categorical values directly.

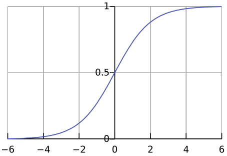
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



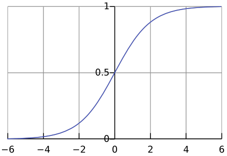
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

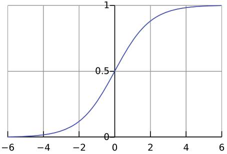
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2 \qquad \text{non-convex}$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

Logistic regression

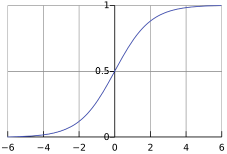
New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

$$c(x_i, y_i) = \begin{cases} -\ln(\sigma(ax_i + b)) & y_i = 1 \\ -\ln(1 - \sigma(ax_i + b)) & y_i = 0 \end{cases} \qquad \text{convex}$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

Logistic regression

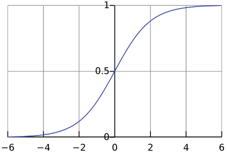
New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

$$c(x_i, y_i) = -y_i \ln(\sigma(ax_i + b)) - (1 - y_i) \ln(1 - \sigma(ax_i + b)) \quad \text{convex}$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

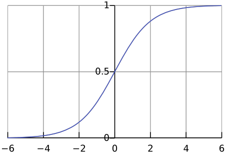
Logistic regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = -\sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

Here, σ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



σ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

More advanced models

We are now moving toward more advanced models to express more complex structures in the data.

More advanced models

We are now moving toward more advanced models to express more complex structures in the data.

- We cannot go too complex! Avoid overfitting.

  Enough to capture the relevant structures while leaving irrelevant variability aside.

More advanced models

We are now moving toward more advanced models to express more complex structures in the data.

- We cannot go too complex! Avoid overfitting.

  Enough to capture the relevant structures while leaving irrelevant variability aside.
- No closed-form solutions, not even convex losses in general.

More advanced models

We are now moving toward more advanced models to express more complex structures in the data.

- We cannot go too complex! Avoid overfitting.

  Enough to capture the relevant structures while leaving irrelevant variability aside.
- No closed-form solutions, not even convex losses in general.
- We need strong priors. More examples? More regularization?

  Data $\gg$ # params forces the model to learn general features

More advanced models

We are now moving toward more advanced models to express more complex structures in the data.

- We cannot go too complex! Avoid overfitting.

  Enough to capture the relevant structures while leaving irrelevant variability aside.
- No closed-form solutions, not even convex losses in general.
- We need strong priors. More examples? More regularization?

  Data $\gg$ # params forces the model to learn general features

With neural networks, the number of parameters will be very high, so the risk of overfitting is always behind the corner.

Suggested reading

On polynomial regression vs. neural nets:
https://arxiv.org/pdf/1806.06850

Proof that the logistic loss is convex:
https://math.stackexchange.com/questions/1582452/logistic-regression-prove-that-the-cost-function-is-convex