Machine Learning Tricks

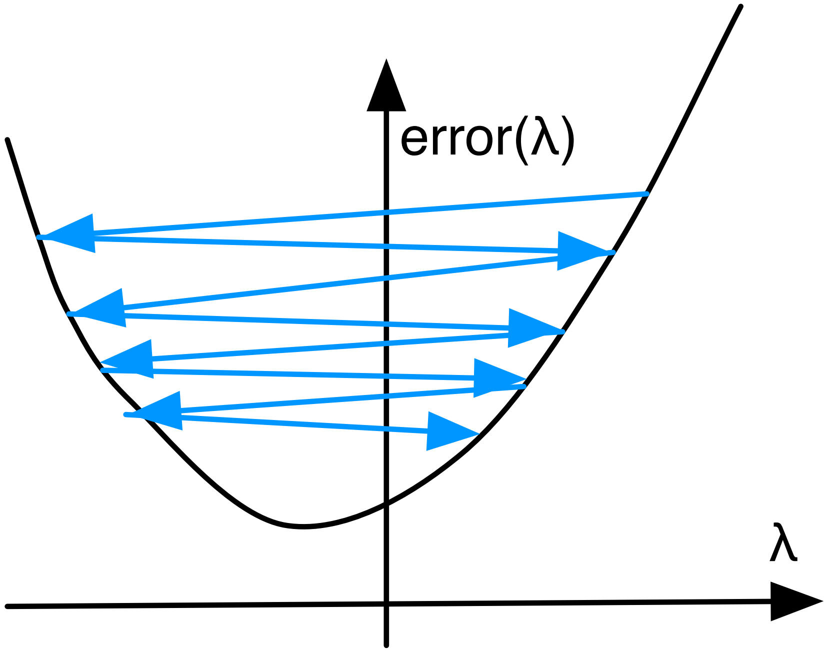
Philipp Koehn

Machine Learning

- Myth of machine learning
  - given: real world examples
  - automatically build model
  - make predictions
- Promise of deep learning
  - do not worry about specific properties of problem
  - deep learning automatically discovers the feature
- Reality: bag of tricks

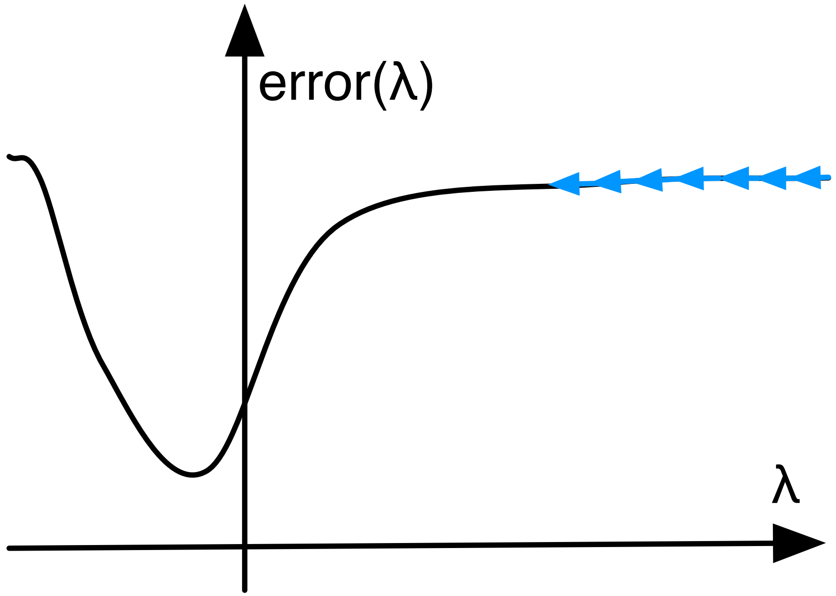
failures in machine learning

Failures in Machine Learning



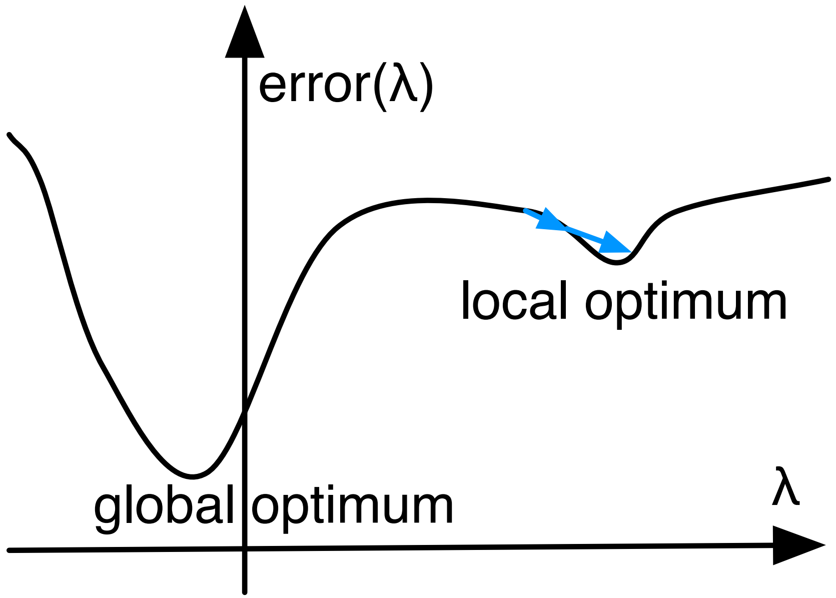
Too high learning rate may lead to too drastic parameter updates→ overshooting the optimum

Failures in Machine Learning



Bad initialization may require many updates to escape a **plateau**

Failures in Machine Learning



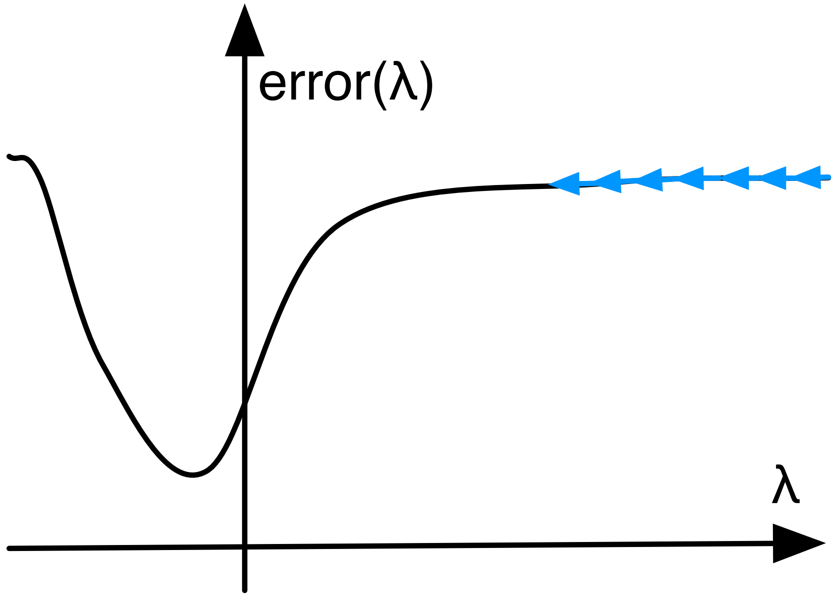
Local optima trap training

Learning Rate

- Gradient computation gives direction of change
- Scaled by learning rate
- Weight updates
- Simplest form: fixed value
- Annealing
  - start with larger value (big changes at beginning)
  - reduce over time (minor adjustments to refine model)

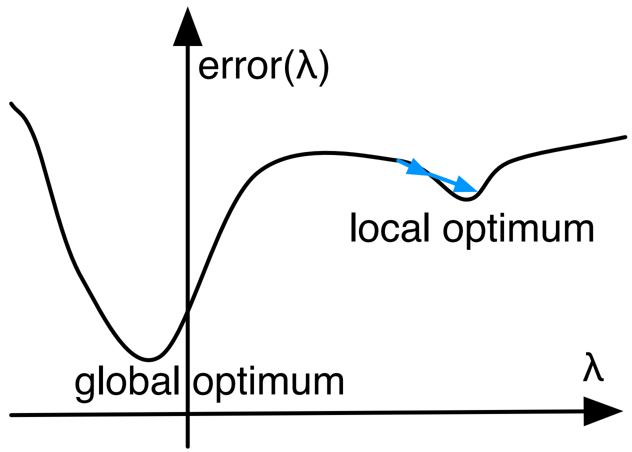
Initialization of Weights

- Initialize weights to random values
- But: range of possible values matters



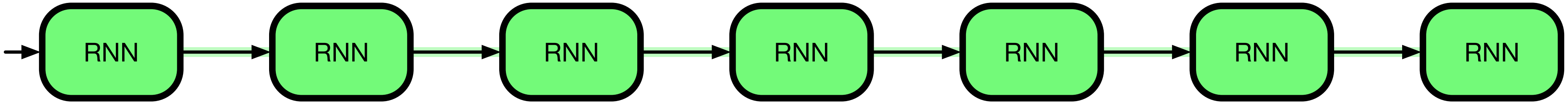
Local Optima

- Cartoon depiction



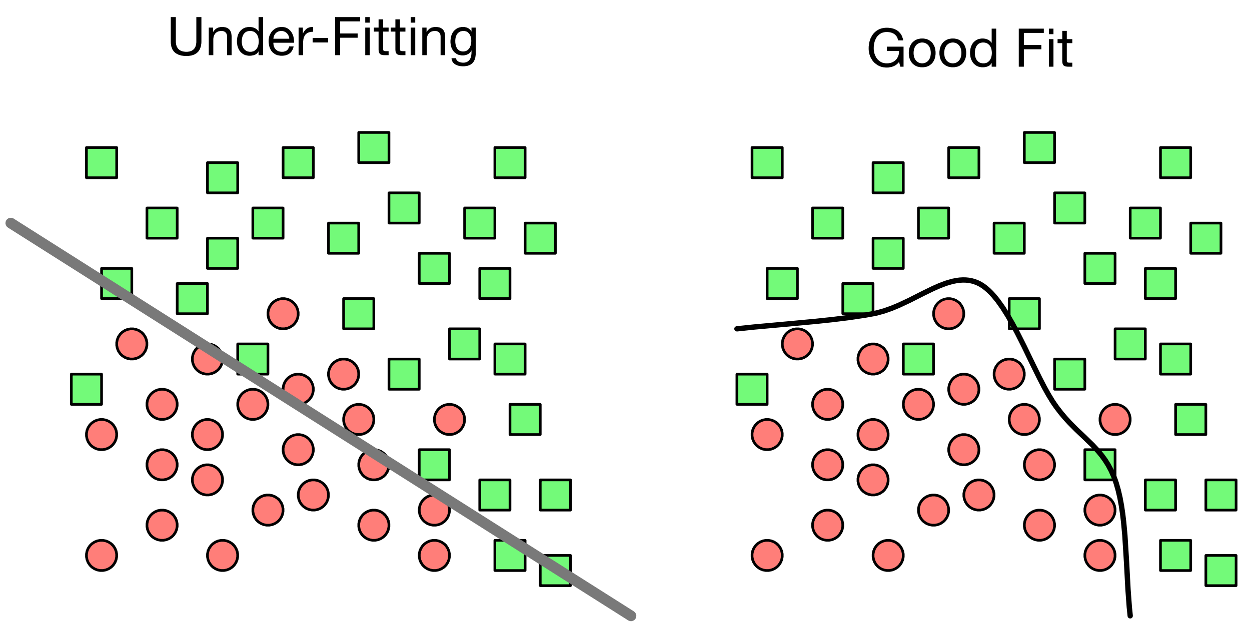
- Reality
  - highly dimensional space
  - complex interaction between individual parameter changes
  - ”bumpy”

Vanishing and Exploding Gradients



- Repeated multiplication with same values
- If gradients are too low $\rightarrow 0$
- If gradients are too big $\rightarrow \infty$

Overfitting and Underfitting



- Complexity of the problem has too match the capacity of the model
- Capacity $\simeq$ number of trainable parameters

ensuring randomness

Ensuring Randomness

- Typical theoretical assumption

independent and identically distributed

training examples

- Approximate this ideal
  - avoid undue structure in the training data
  - avoid undue structure in initial weight setting
- ML approach: Maximum entropy training
  - Fit properties of training data
  - Otherwise, model should be as random as possible (i.e., has maximum entropy)

Shuffling the Training Data

- Typical training data in machine translation
  - different types of corpora
    - European Parliament Proceedings
    - collection of movie subtitles
  - temporal structure in each corpus
  - similar sentences next too each other (e.g., same story / debate)

- Online updating: last examples matter more

- Convergence criterion: no improvement recently

  $\rightarrow$ stretch of hard examples following easy examples: prematurely stopped

$\Rightarrow$ randomly shuffle the training data

(maybe each epoch)

Weight Initialization

- Initialize weights to random values
- Values are chosen from a uniform distribution
- Ideal weights lead to node values in transition area for activation function

adjusting the learning rate

Adjusting the Learning Rate

- Gradient descent training: weight update follows the gradient downhill
- Actual gradients have fairly large values, scale with a learning rate (low number, e.g., $\mu = 0.001$)
- Change the learning rate over time
  - starting with larger updates
  - refining weights with smaller updates
  - adjust for other reasons
- Learning rate schedule

Batched Gradient Updates

- Accumulate all weight updates for all the training example $\rightarrow$ update (converges slowly)

- Process each training example $\rightarrow$ update (stochastic gradient descent) (quicker convergence, but last training disproportionately higher impact)

- Process data in batches
  - compute all their gradients for individual word predictions errors
  - use sum over each batch to update parameters

  $\rightarrow$ better parallelization on GPUs

- Process data on multiple compute cores
  - batch processing may take different amount of time
  - asynchronous training: apply updates when they arrive
  - mismatch between original weights and updates may not matter much

avoiding local optima

Avoiding Local Optima

- One of hardest problem for designing neural network architectures and optimization methods

- Ensure that model converges to at least to a set of parameter values that give results close to this optimum on unseen test data.

- There is no real solution to this problem.

- It requires experimentation and analysis that is more craft than science.

- Still, this section presents a number of methods that generally help avoiding getting stuck in local optima.

Overfitting and Underfitting

- Neural machine translation models
  - 100s of millions of parameters
  - 100s of millions of training examples (individual word predictions)

- No hard rules for relationship between these two numbers

- Too many parameters and too few training examples $\rightarrow$ overfitting

- Too few parameters and many training examples $\rightarrow$ underfitting

Regularization

- Motivation: prefer as few parameters as possible

- Strategy: set un-needed paramters a value of 0

- Method
  - adjust training objective
  - add cost for any non-zero parameter
  - typically done with L2 norm

- Practical impact
  - derivative of L2 norm is value of parameter
  - if not signal from training: reduce value of parameter
  - alsp called weight decay

- Not common in deep learning, but other methods understood as regularization

Curriculum Learning

- Human learning
  - learn simple concepts first
  - learn more complex material later
- Early epochs: only easy training examples
  - only short sentences
  - create artificial data by extracting smaller segments
    (similar to phrase pair extraction in statistical machine translation)
  - Later epochs: all training data
- Not easy to callibrate

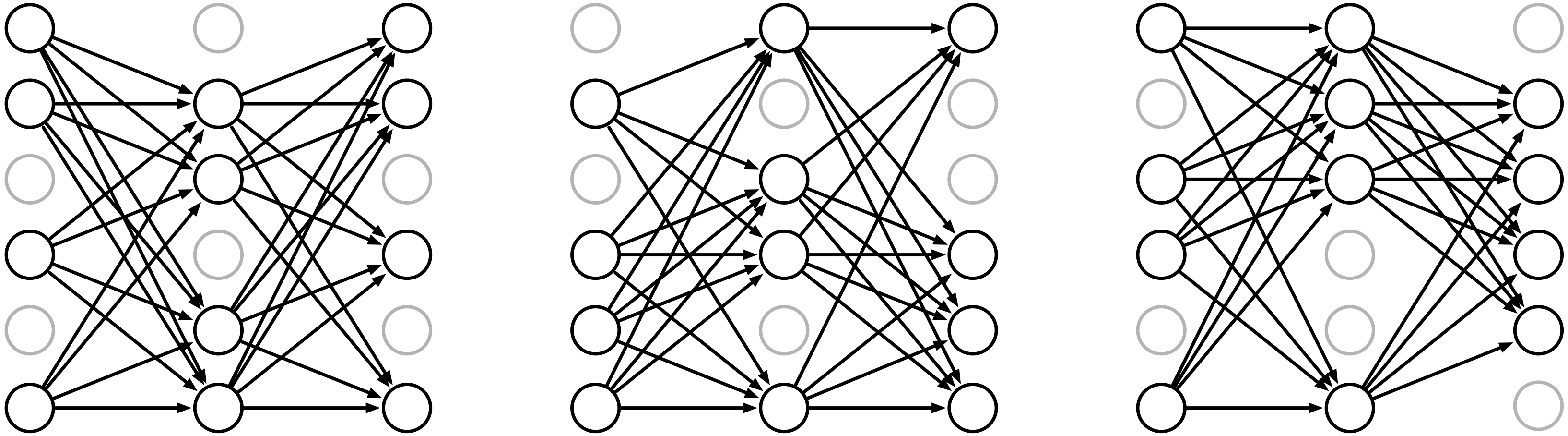
Dropout

- Training may get stuck in local optima
  - some properties of task have been learned
  - discovery of other properties would take it too far out of its comfort zone.

- Machine translation example
  - model learned the language model aspects
  - but cannot figure out role of input sentence

- Drop out: for each batch, eliminate some nodes

Dropout



- Dropout
  - For each batch, different random set of nodes is removed
  - Their values are set to 0 and their weights are not updated
  - 10%, 20% or even 50% of all the nodes
- Why does this work?
  - robustness: redundant nodes play similar nodes
  - ensemble learning: different subnetworks are different models

generative adversarial training

Sequence-Level Training

- Traditional training
  - predict one word at a time
  - compare against correct word
  - proceed training with correct word
- Sequence-level training
  - predict entire sequence
  - measure translation with sentence-level metric (e.g., BLEU)
- May use n-best translations, beam search, etc.

Generative Adversarial Networks (GAN)

- Game between two players
  - generator proposes a translation
  - discriminator distinguishes between generator's translation and human translation
  - generator tries to fool discriminator

- Training example: input sentence x and output sentence y

- Generator
  - traditional neural machine translation model
  - generates full sentence translations t for each input sentence

- Discriminator
  - is trained to classify (x, y) as correct example
  - is trained to classify (x, t) as generated example

Generative Adversarial Networks (GAN)

1. First train generator to some maturity

2. Train discriminator on generator predictions and human reference translations

3. Train jointly

   – generator with additional objective to fool discriminator
   – discriminator to do well on detecting generator's output as such

- In practice, this is hard to callibrate correctly