COL333/671: Introduction to AI

Semester I, 2021

Learning - I

Rohan Paul

Outline

- Last Class
  - Probabilistic Reasoning over Time
- This Class
  - Learning: classification, Naïve Bayes parameter learning, MLE/MAP
- Reference Material
  - Please follow the notes as the primary reference on this topic. Supplementary reading on topics covered in class from AIMA Ch 20 sections 20.1 – 20.2.4.

Acknowledgement

These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Doina Precup, Dorsa Sadigh, Percy Liang, Mausam, Parag, Emma Brunskill, Alexander Amini, Dan Klein, Anca Dragan, Nicholas Roy and others.

Machine Learning Models

- Models are useful for making optimal decisions
- Data is necessary for learning a good model
- Example: building a model for the classification

- Task: given inputs x, predict labels (classes) y
- Examples:
  - Spam detection (input: document,
    classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Fraud detection (input: account activity, classes: fraud / no fraud)

Classification Task

- Input: an email
- Output: spam or not spam

- Features or attributes that can be useful for classification
  - Words: “FREE!”
  - Text Patterns: ALL CAPS
  - Others: whether the sender is in the contact list or not
  - …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

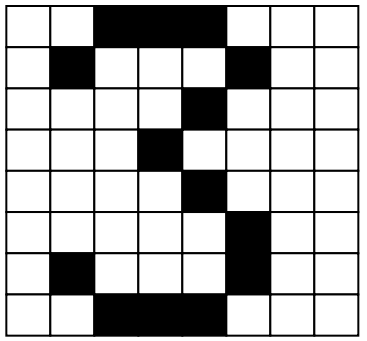
TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Classification Task

- Input: images / pixel grids
- Output: a digit 0-9



- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images


0


1


2


1

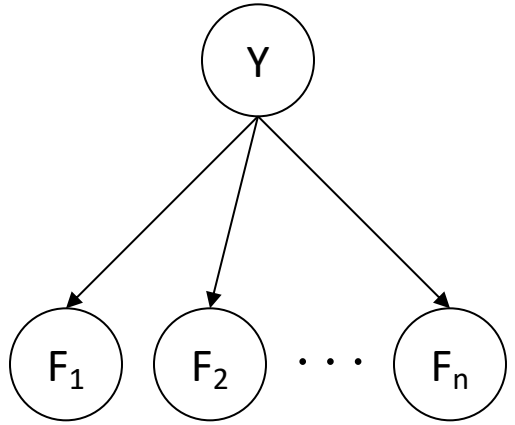

Not clear

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …

Naïve Bayes Model for Classification

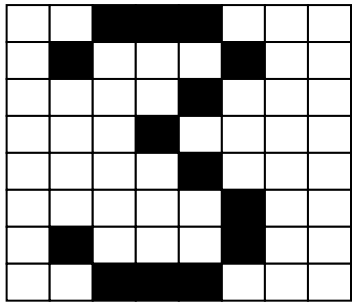
- Naïve Bayes: Assume all features are independent effects of the label

- Simple digit recognition:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.



$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

$$P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$$

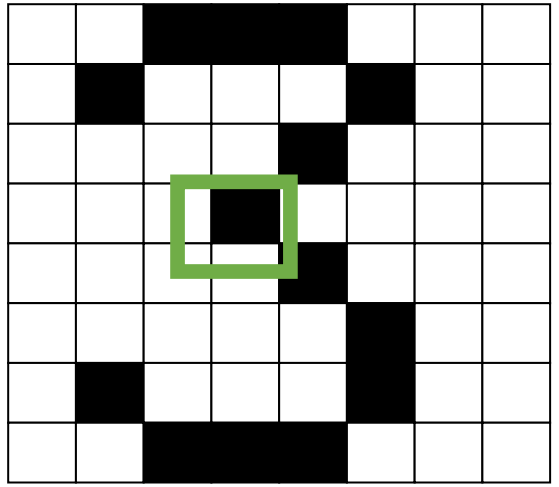


Using Naïve Bayes

- Need the estimates of local conditional probability tables.
  - P(Y), the prior over labels
  - $P(F_i|Y)$ for each feature (evidence variable)
  - These probabilities are collectively called the ***parameters*** of the model and denoted by θ
    - Till now, the table values were provided.
    - Now, use data to acquire these values.

Estimating Conditional Probability Tables

- P(Y) – how frequent is the class-type for digit 3?
  - If you take a sample of images of numbers how frequent is this number
- $P(F_i|Y)$ – for digit 3 what fraction of the time the cell is on?
  - Conditioned on the class type how frequent is the feature
- Use **relative frequencies** from the data to estimate these values.



Maximum Likelihood Estimation (MLE)

- Write the expression for the likelihood of the data given the probabilistic model as characterized by the parameters.
  - Take the log likelihood.
- Optimize the likelihood given the parameters.
  - Write down the derivative of the log likelihood with respect to each parameter.
  - Find the parameters such that the derivatives are zero.

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$

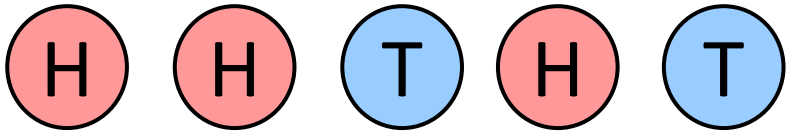
$$= \arg\max_{\theta} \prod_{i} P_{\theta}(X_i)$$

$$P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

MLE Example

- Estimating the parameters of a biased coin.
- Setup
  - P(Heads) = θ, P(Tails) = 1-θ
  - Flips are independent and identically distributed according to an unknown distribution.
  - Observe a sequence D of α_H Heads and α_T Tails
  - Learning task: parameter θ.
  - Hypothesis space: Binomial distributions

Sequence of coin toss observations



$$D=\{x_i \mid i=1 \ldots n\},\ P(D \mid \theta) = \Pi_i P(x_i \mid \theta)$$

MLE Steps

- Formulate the objective Function
- MLE of θ that maximizes the probability of D.
- Optimize the objective function.

$$P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

$$\begin{aligned}\widehat{\theta} &= \arg\max_{\theta} P(\mathcal{D} \mid \theta) \\ \widehat{\theta} &= \arg\max_{\theta} \ln P(\mathcal{D} \mid \theta) \\ &= \arg\max_{\theta} \ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}\end{aligned}$$

$$\begin{aligned}\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) &= \frac{d}{d\theta}\left[\ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}\right] \\ &= \frac{d}{d\theta}\left[\alpha_H \ln\theta + \alpha_T \ln(1-\theta)\right] \\ &= \alpha_H \frac{d}{d\theta}\ln\theta + \alpha_T \frac{d}{d\theta}\ln(1-\theta) \\ &= 0\end{aligned}$$

$$\widehat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T}$$

Problem: values not seen in the training data

$P(\text{features}, C = 2)$

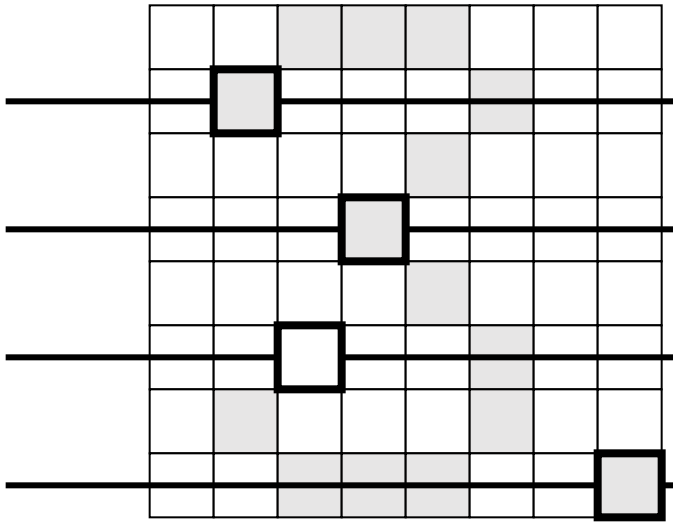
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

If one feature was not seen in the training data, the likelihood goes to zero. If we did not see this feature in the training data, does not mean we will not see this in training. Essentially overfitting to the training data set.

Laplace Smoothing

- Pretend that every outcome occurs **once more** than it is actually observed.
- If certain counts are **not** seen in training does not mean that they have zero probability of occurring in future.
- Another version of Laplace smoothing
  - instead of 1, add **k times**
  - k is an adjustable parameter.
- Essentially, encodes a **prior** (pseudo-counts).

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]} = \frac{c(x)+1}{N+|X|}$$

$$P_{LAP,k}(x) = \frac{c(x)+k}{N+k|X|}$$

Maximum a posteriori (MAP) Estimates

- **Maximum likelihood estimate (MLE)**
  - Estimates the parameters that maximizes the data likelihood.
  - Relative counts give MLE estimates

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_{i} P_{\theta}(X_i)$$

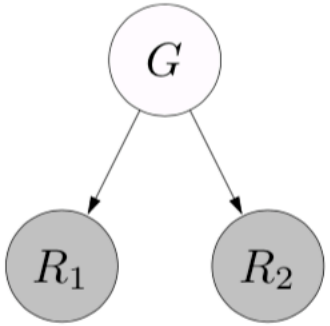
- **Maximum a posteriori estimate (MAP)**
  - Bayesian parameter estimation
  - Encodes a prior over the parameters (not all parameters are equal prior values).
  - Combines the prior and the likelihood while estimating the parameters.

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

Some variables may be unobserved

Consider a problem of inferring the genre of a movie (Comedy or Drama) from the ratings given by two film reviewers R1 and R2.

Setting where only the ratings are observed.



What if we **don't observe** some of the variables?

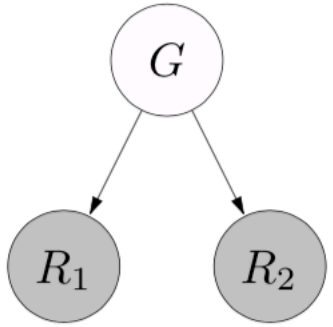
$$\mathcal{D}_{\text{train}} = \{(?, 4, 5), (?, 4, 4), (?, 5, 3), (?, 1, 2), (?, 5, 4)\}$$

Slide adapted from Dorsa Sadigh and Percy Liang

Maximum Marginal Likelihood

Variables: H is hidden, $E = e$ is observed

Example:



$$H = G \quad E = (R_1, R_2) \quad e = (1, 2)$$
$$\theta = (p_G, p_R)$$

Marginalize over the latent variables in the likelihood

Maximum marginal likelihood objective:

$$\max_\theta \prod_{e \in \mathcal{D}_{\text{train}}} \mathbb{P}(E = e; \theta)$$

$$= \max_\theta \prod_{e \in \mathcal{D}_{\text{train}}} \sum_h \mathbb{P}(H = h, E = e; \theta)$$

Slide adapted from Dorsa Sadigh and Percy Liang

Expectation Maximization

A form a unsupervised learning. Alternate the Expectation (E) and the Maximization (M) steps till convergence.

Initialize θ

E-step:

- Compute $q(h) = \mathbb{P}(H = h \mid E = e; \theta)$ for each h (use any probabilistic inference algorithm)
- Create weighted points: (h, e) with weight $q(h)$

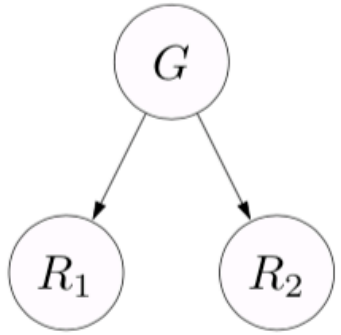
M-step:

- Compute maximum likelihood (just count and normalize) to get θ

Repeat until convergence.

Slide adapted from Dorsa Sadigh and Percy Liang

Expectation Maximization



$$\mathcal{D}_{\text{train}} = \{(?, 2, 2), (?, 1, 2)\}$$

θ:

g	$p_G(g)$
c	0.5
d	0.5

g	r	$p_R(r \mid g)$
c	1	0.4
c	2	0.6
d	1	0.6
d	2	0.4

E-step →

(r_1, r_2)	g	$\mathbb{P}(G = g, R_1 = r_1, R_2 = r_2)$	$q(g)$
(2, 2)	c	$0.5 \cdot 0.6 \cdot 0.6 = 0.18$	$\frac{0.18}{0.18+0.08} = 0.69$
(2, 2)	d	$0.5 \cdot 0.4 \cdot 0.4 = 0.08$	$\frac{0.08}{0.18+0.08} = 0.31$
(1, 2)	c	$0.5 \cdot 0.4 \cdot 0.6 = 0.12$	$\frac{0.12}{0.12+0.12} = 0.5$
(1, 2)	d	$0.5 \cdot 0.6 \cdot 0.4 = 0.12$	$\frac{0.12}{0.12+0.12} = 0.5$

M-step →

θ:

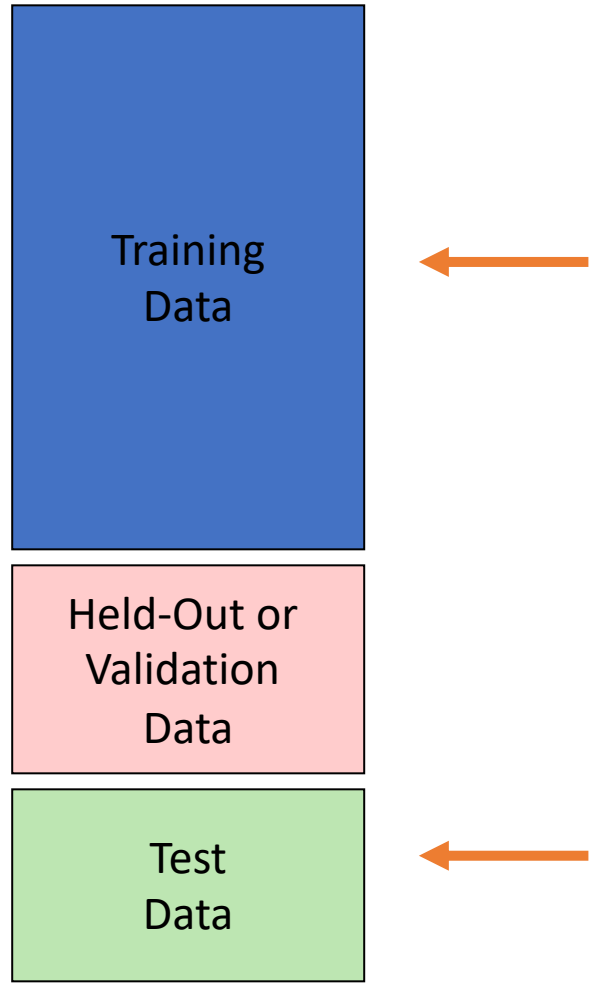
g	count	$p_G(g)$
c	0.69 + 0.5	0.59
d	0.31 + 0.5	0.41

g	r	count	$p_R(r \mid g)$
c	1	0.5	0.21
c	2	0.5 + 0.69 + 0.69	0.79
d	1	0.5	0.31
d	2	0.5 + 0.31 + 0.31	0.69

Slide adapted from Dorsa Sadigh and Percy Liang

Training and Generalization Errors

- **Goal**
  - Doing learning to estimate parameters of a model.
  - Optimizing a loss function that measures the good ness of a model.
- **Data set consists of labeled instances**
  - Digit images and digits
  - Emails with labels such as spam or no spam
- **Need for Generalization**
  - A machine learning algorithm must perform well on new, previously unseen inputs
  - Note just those on which the model was trained.
  - Essentially, it should not memorize the data.
- **Training and Test sets**
  - The learner should never look at the test data.
  - Training set -> training error
  - Test set -> “generalization” error
- **Central Challenge for a Learning Algorithm**
  - Make the training error small
  - Make the gap between the training and the test error small

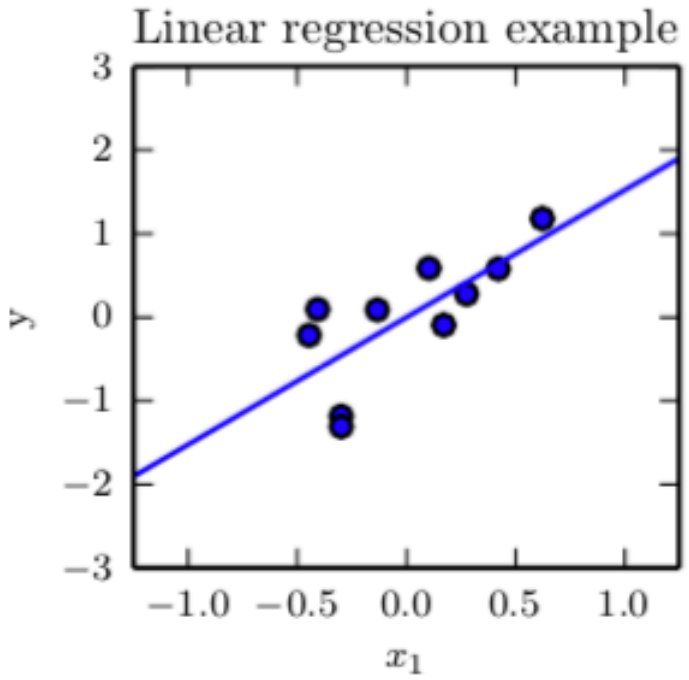


Linear Regression Task

Linear regression (no bias) $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$

Linear regression (with bias) $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$

Error term/Loss term $\mathrm{MSE}_{\mathrm{test}} = \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\mathrm{test})} - \boldsymbol{y}^{(\mathrm{test})}||_2^2,$



Learning: Optimizing the loss will estimate the model parameters w and b.
Online: Given the trained model, we can predict a value.

Examples:
- Predicting pollution levels from visibility
- Predicting reactivity of a molecule from structural data.

Material from
https://www.deeplearningbook.org/
Chapter 5

Linear Regression Example

Learning/Training:
Optimize the error w.r.t. the model parameters, w

$$\nabla_{\boldsymbol{w}} \text{MSE}_{\text{train}} = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\text{train})} - \boldsymbol{y}^{(\text{train})}||_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} ||\boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})}||_2^2 = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right)^\top \left(\boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right) = 0 \tag{5.9}$$

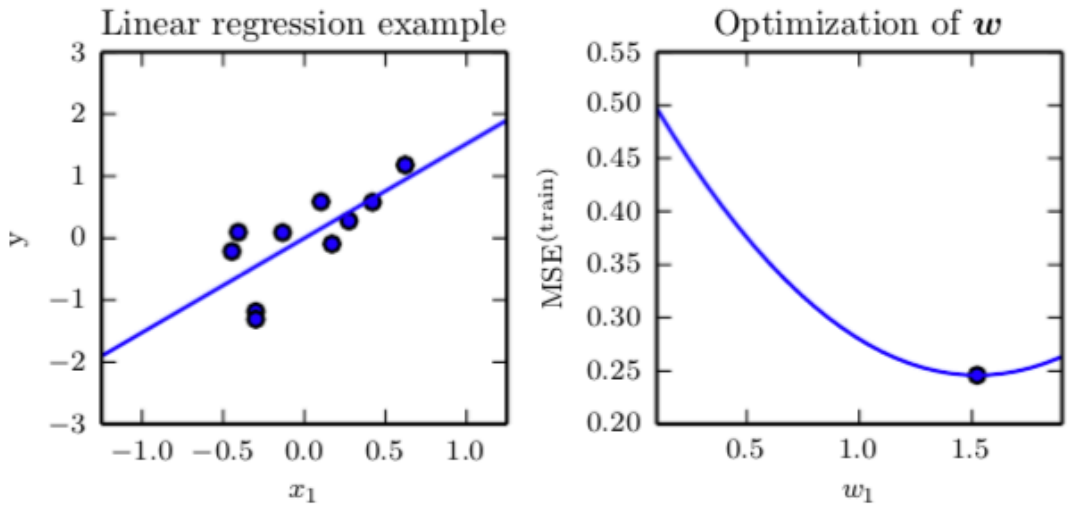
$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - 2\boldsymbol{w}^\top \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} + \boldsymbol{y}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} \right) = 0 \tag{5.10}$$

$$\Rightarrow 2\boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - 2\boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} = 0 \tag{5.11}$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \right)^{-1} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})} \tag{5.12}$$

Inference:
Use the trained model i.e. w, to perform predictions

Optimal w and the implied linear model



Function/model space **Parameter space**

Material from
https://www.deeplearningbook.org/
Chapter 5

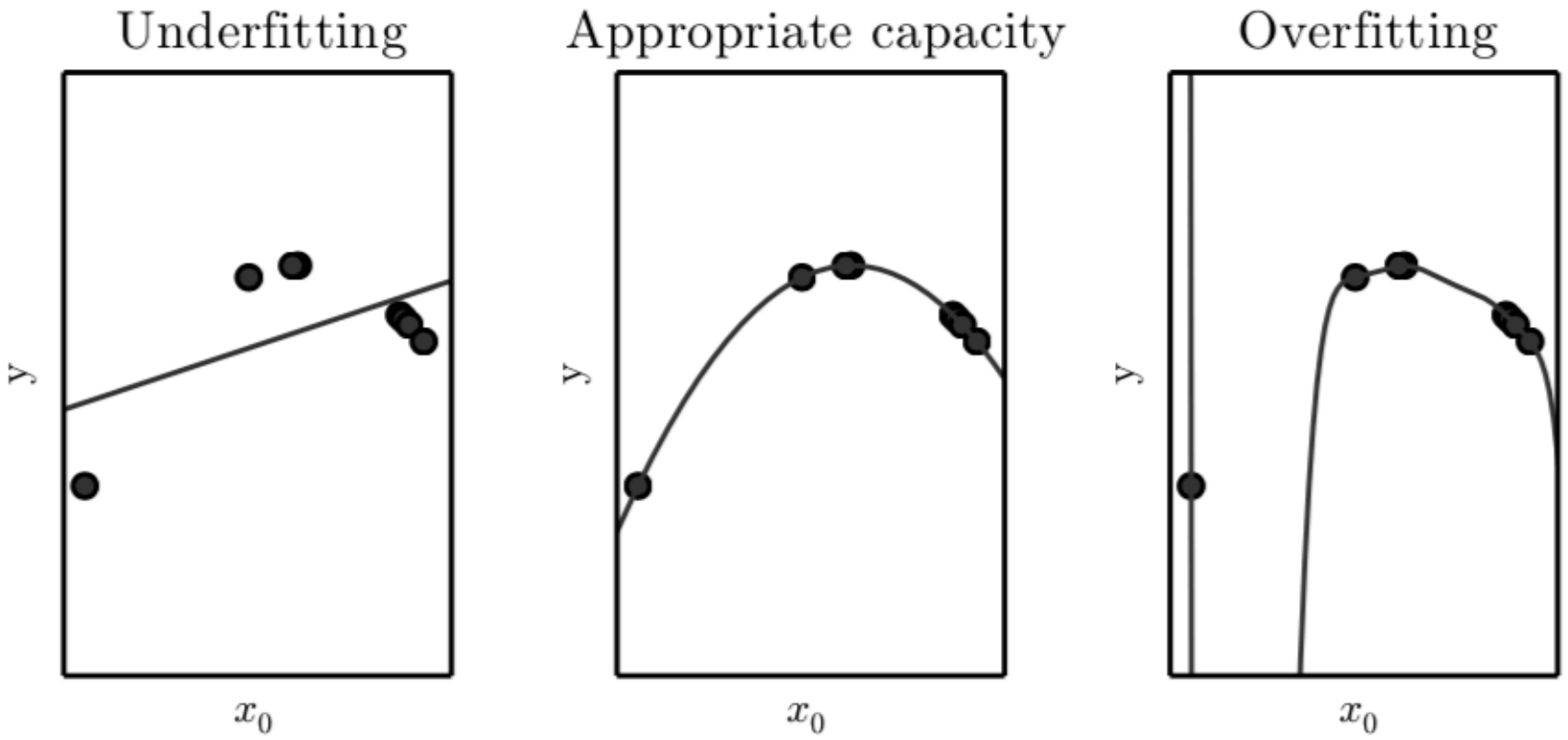
Model Fitting to Data

Overfitting and underfitting with polynomial functions

We seek a reasonable model that is neither underfitting nor over fitting.

That means, we prefer certain types of models.

How to "regularize" or solution, incorporate certain preferences?



Material from
https://www.deeplearningbook.org/
Chapter 5

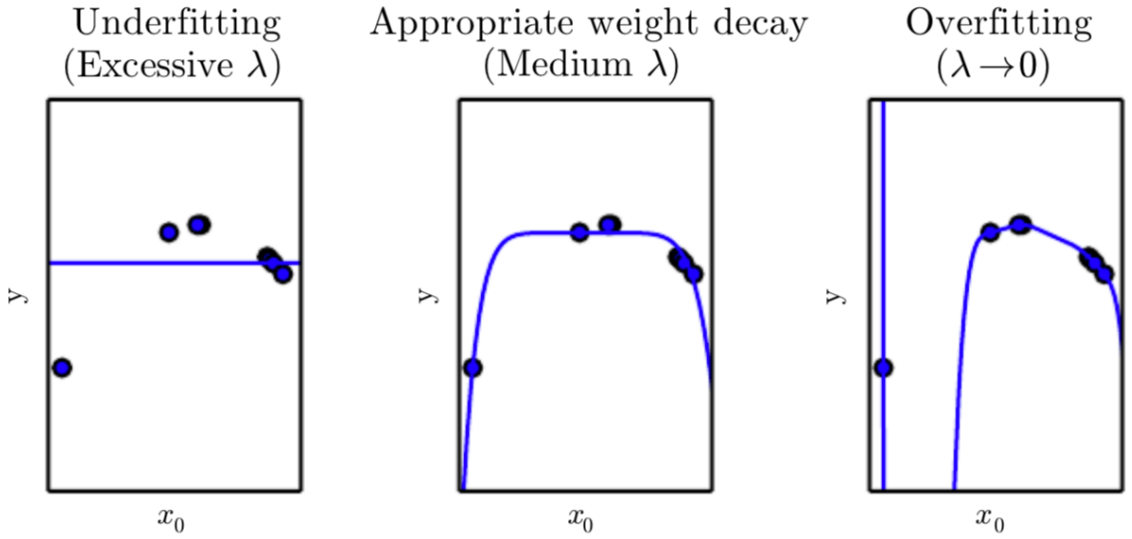
Regularization

- Adding a **preference** for one solution in its hypothesis space to another.
  - Incorporate that preference in the objective function we are optimization.

- **Weight term**
  - Adding a term to the loss function that prefers smaller squared sum of weights. A prior over the parameters.
  - Penalize a very complex model to explain the date.

- **Lambda parameter**
  - Selected ahead of time that controls the strength of our preference for smaller weights.
  - It is a “hyper”-parameter that needs tuning, expressing our trade off.

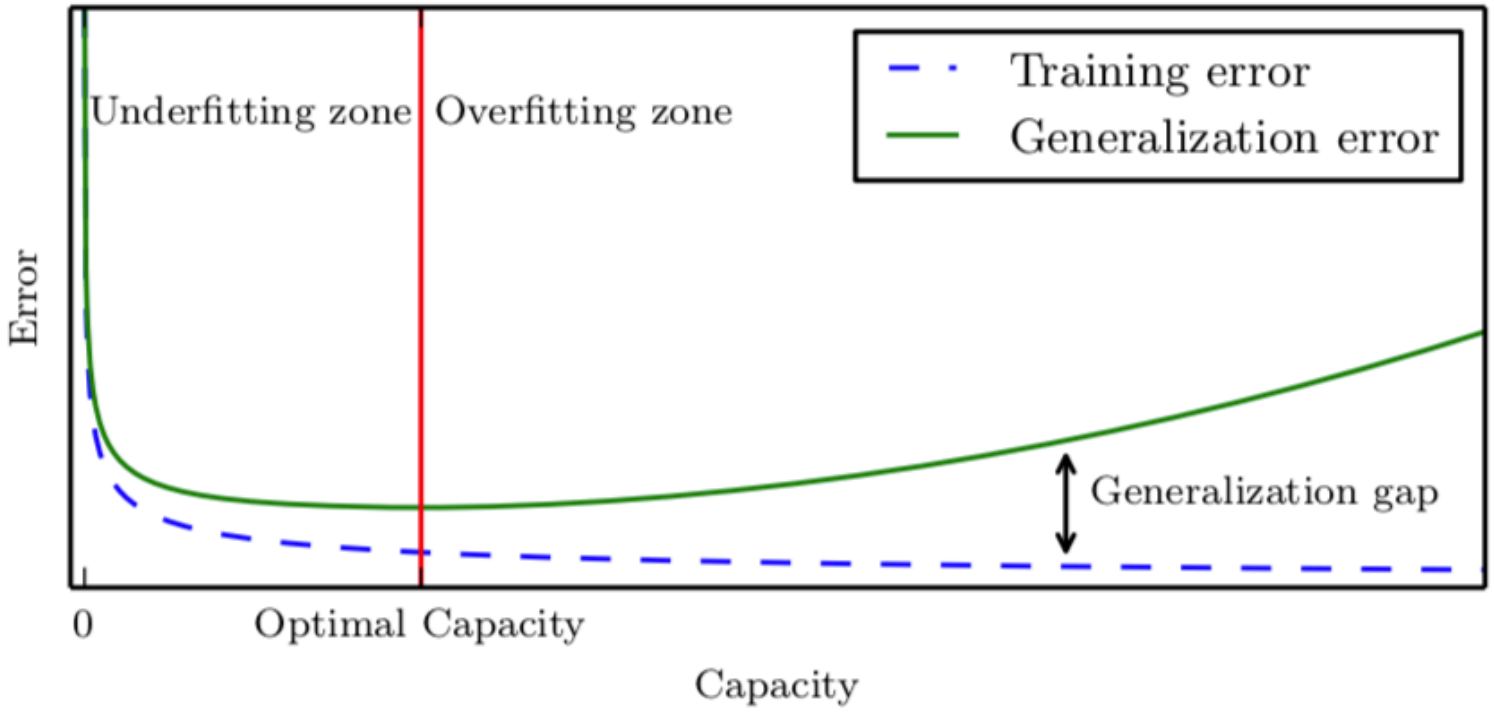
$$J(\boldsymbol{w}) = \text{MSE}_{\text{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$



Material from
https://www.deeplearningbook.org/
Chapter 5

Relation between model capacity and error

- **Underfitting regime**
  - Training error and generalization error are both high.
- **Increase capacity**
  - Training error decreases
  - Gap between training and generalization error increases.
- **Overfitting**
  - The size of this gap outweighs the decrease in training error,
  - capacity is too large, above the optimal capacity.



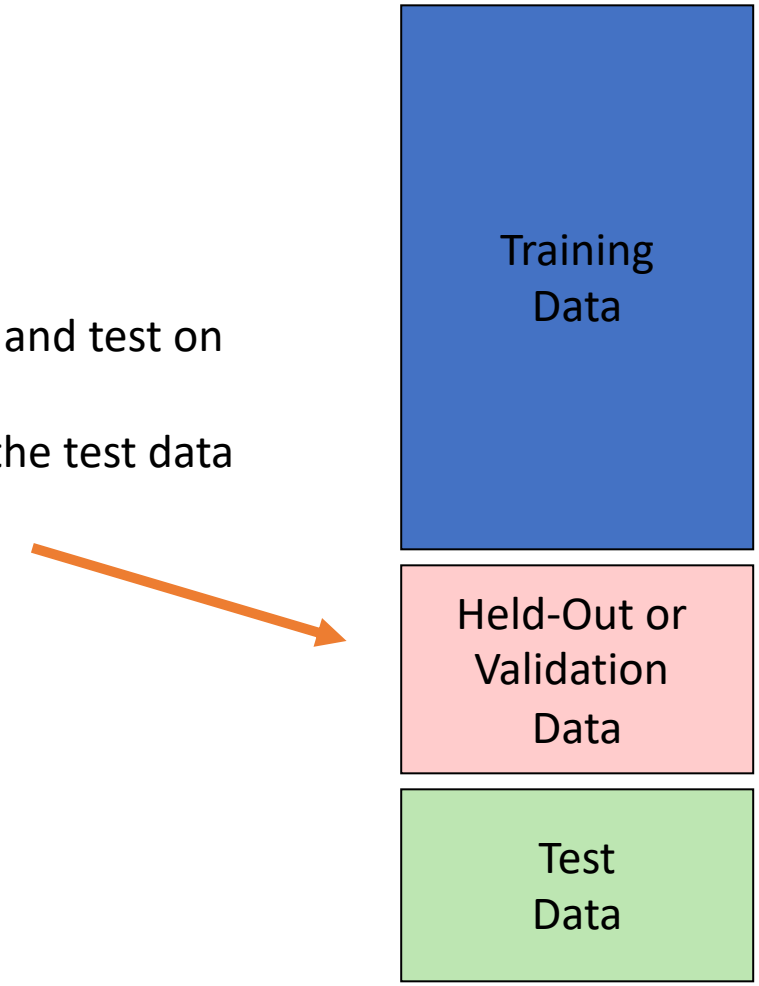
We can train a model only on the training set. The test set is not available during training.
How do we know the generalization error when we cannot use the test set?

Material from
https://www.deeplearningbook.org/
Chapter 5

Hyper-parameters and Validation Sets

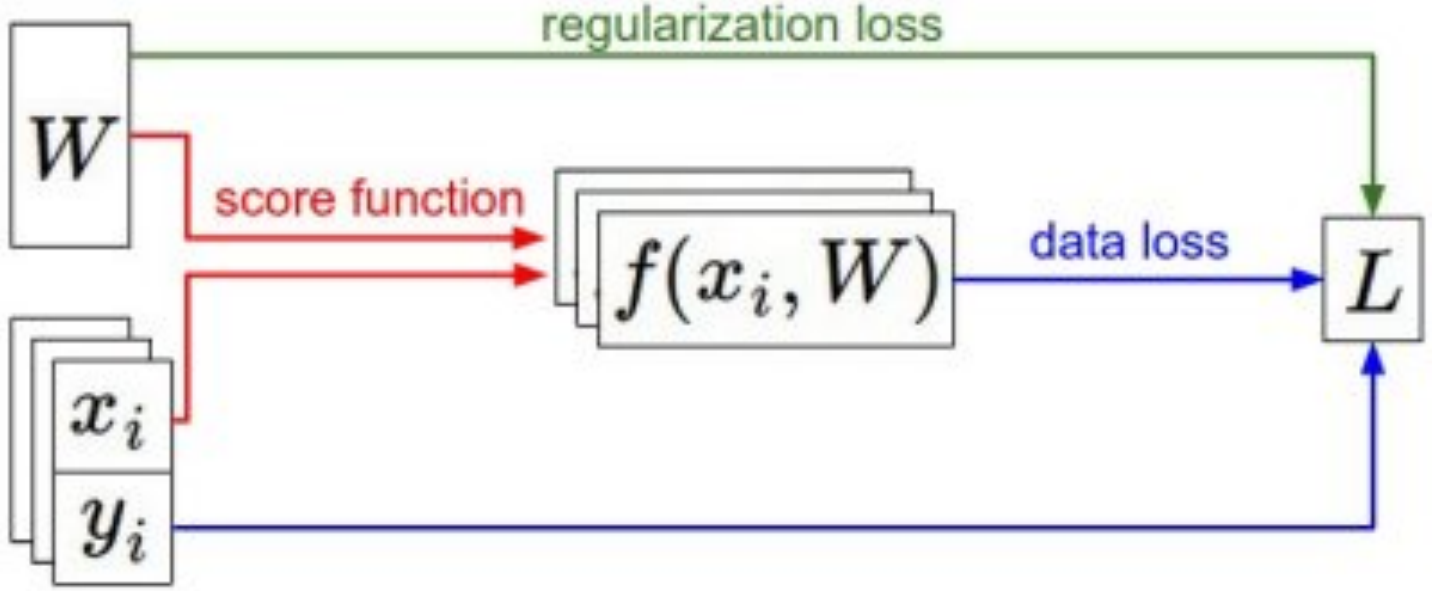
- Parameters: P(X), P(Y|X)
- Hyper-parameters: k, lambda etc.

- Selecting hyper-parameters
  - For each value of the hyperparameters, train and test on the held-out data or the validation data set.
  - Choose the best value and do a final test on the test data



Optimizing the loss

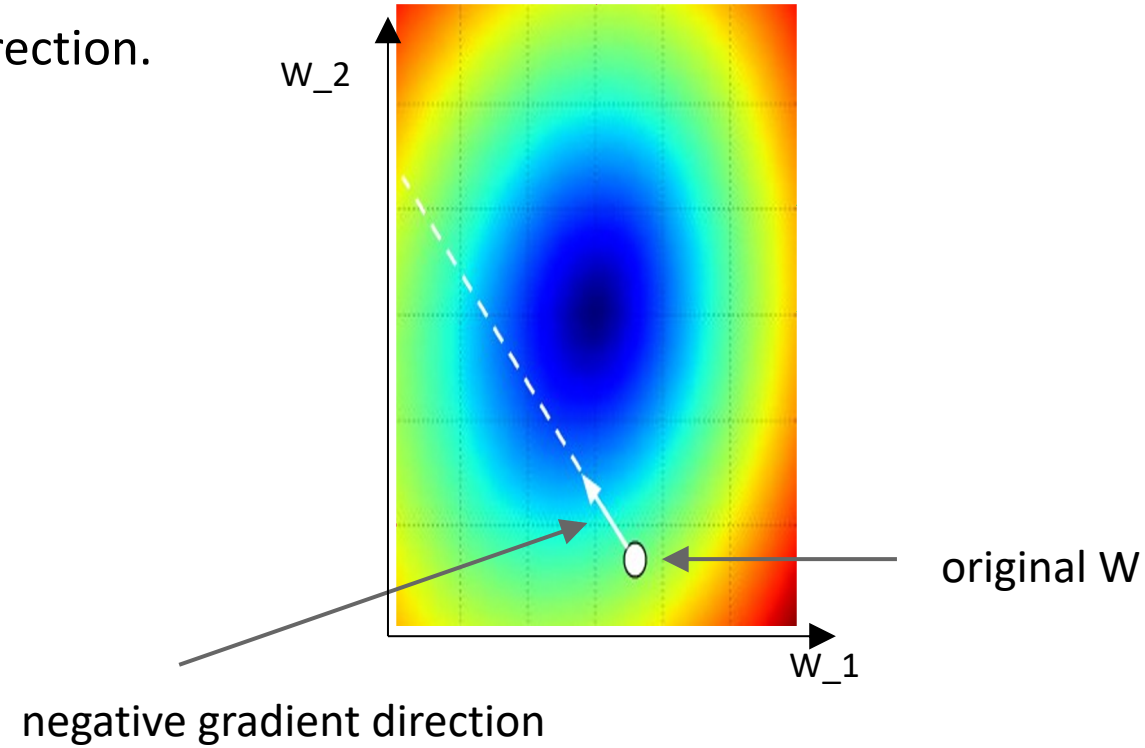
- The loss term is composed of the predictions under the weights and the regularization term.
- The goal is to optimize the loss function given the parameters



Gradient Descent

- Compute the gradient and take the step in that direction.
- Gradient computation
  - Analytic
  - Numerical

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

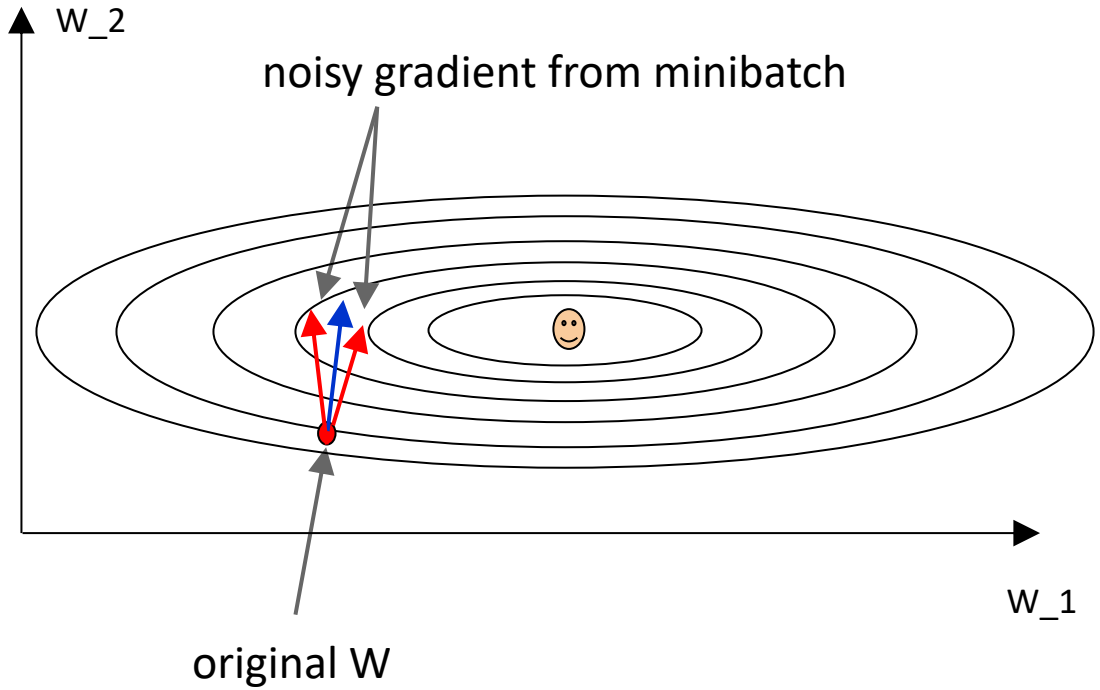


Mini-batch Gradient Descent

Problem: Large data sets. Difficult to compute the full loss function over the entire training set in order to perform only a single parameter update

Solution: Only use a small portion of the training set to compute the gradient.

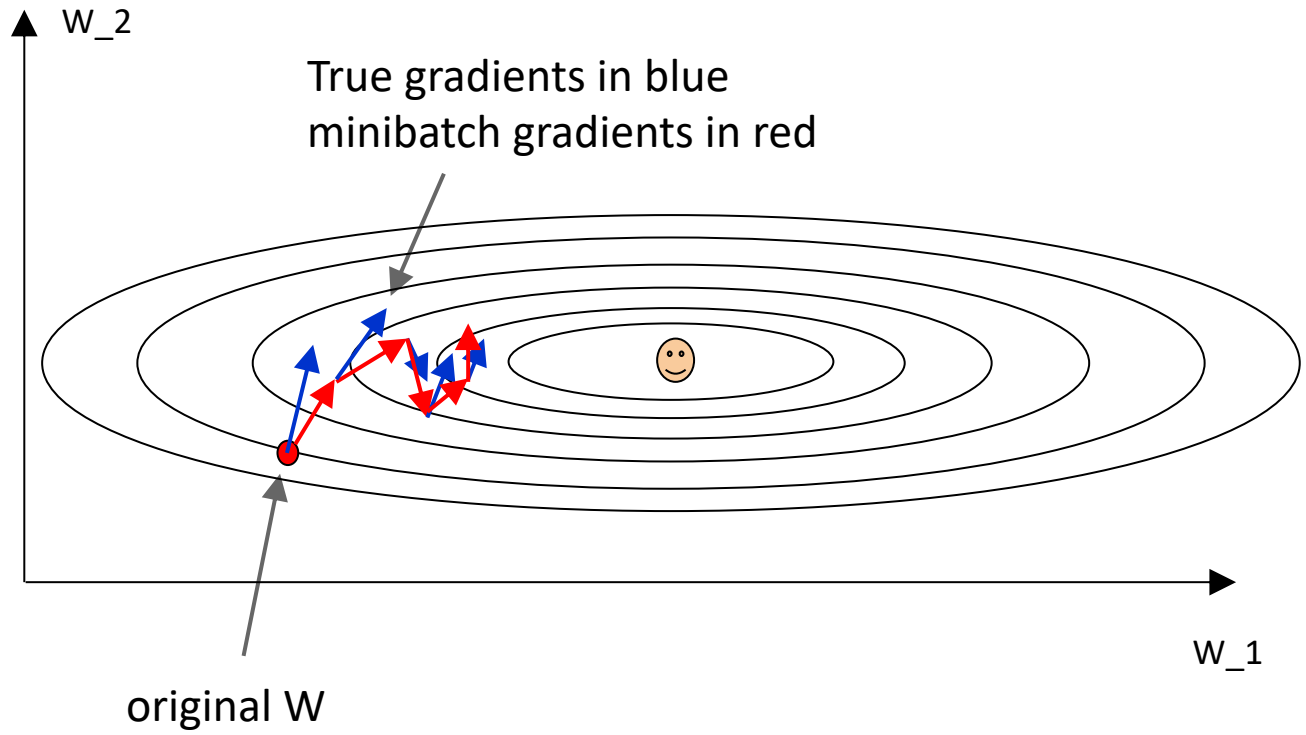
```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```



Stochastic Gradient Descent

Setting the mini-batch to contain only a single example.

Gradients are noisy but still make good progress on average.



Gradient vs Mini-batch Gradient Descent

Cost function decomposes as a sum over training examples of per-example loss function

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y \mid \boldsymbol{x}; \boldsymbol{\theta})$$

Gradient for an additive cost function

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

Mini-batch m', where m' is kept constant while m is increasing.

Update the parameters with the estimated gradient from the mini-batch.

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

Theta denotes the parameters. Same as w used in the previous slides.