REGRESSION

MACHINE LEARNING

Task

Performance

Experience

Algorithms that improve their performance
at some task with experience
– Tom Mitchell (1998)

REGRESSION

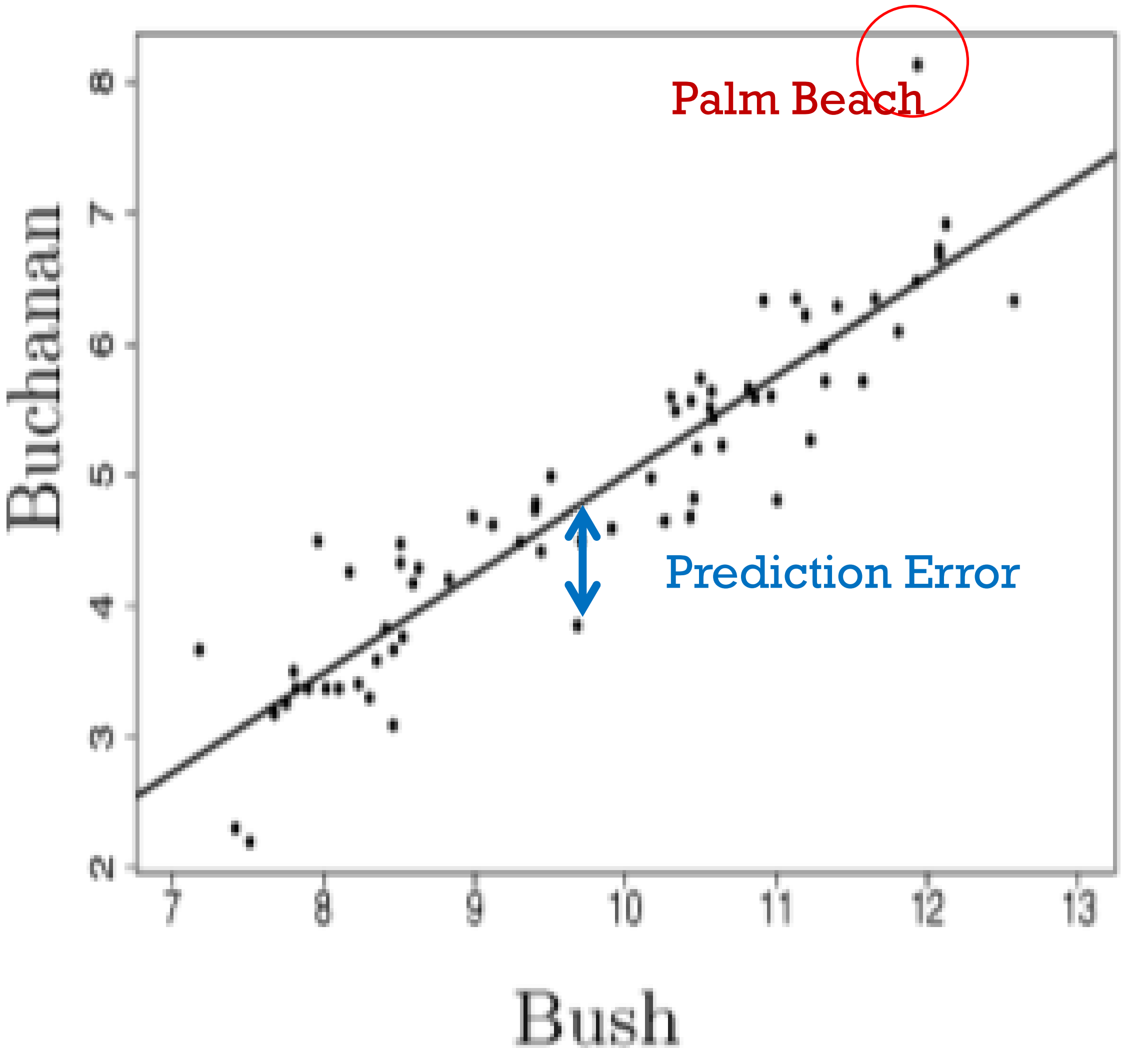
Machine Learning

> Supervised Learning

> Regression

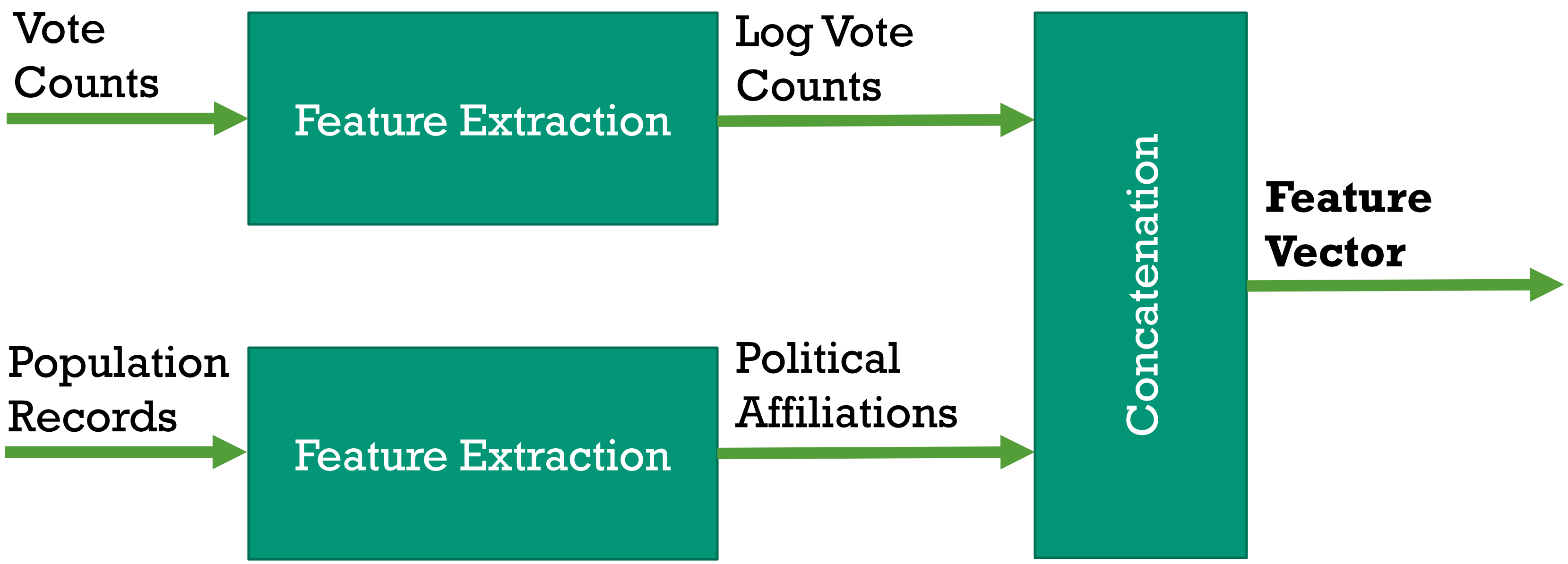
- **Task.** Find function $f: \mathbb{R}^d \to \mathbb{R}$ such that $y \approx f(x;\theta)$
- **Experience.** Training data $\left(x^{(1)}, y^{(1)}\right), \dots, \left(x^{(n)}, y^{(n)}\right)$
- **Performance.** Prediction error $y - f(x;\theta)$ on test data

WORKED EXAMPLE



METHODOLOGY

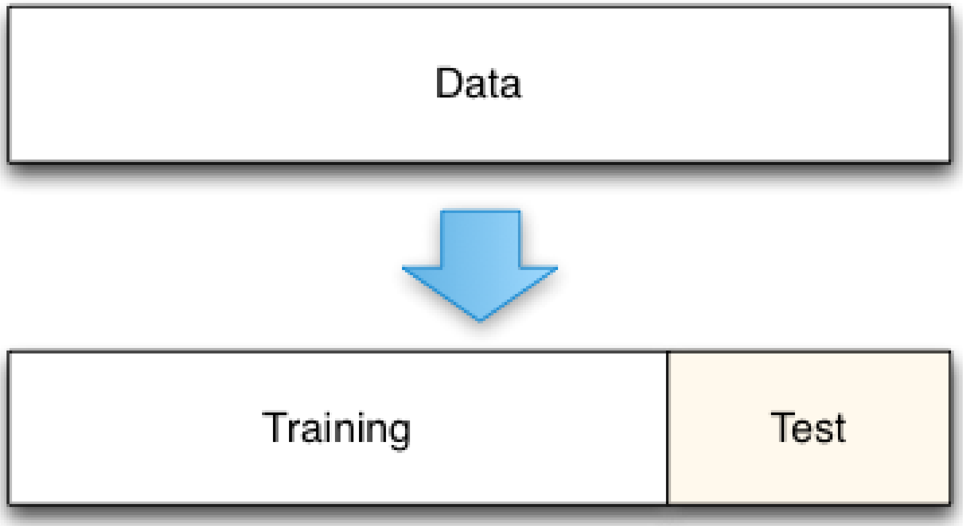
FEATURES


Vote Counts
Feature Extraction
Log Vote Counts
Population Records
Feature Extraction
Political Affiliations
Concatenation
Feature Vector

TRAINING DATA VS TEST DATA

Partition data into:

- Training data set $\mathcal{S}_n$
- Test data set $\mathcal{S}_*$



Training data

$$\mathcal{S}_n = \left\{ \left(x^{(i)}, y^{(i)}\right) \,\middle|\, i = 1, \dots, n \right\}$$

- Features/Inputs $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^\top \in \mathbb{R}^d$
- Response/Output $y^{(i)} \in \mathbb{R}$

Each f is a *predictor* or *hypothesis*

Model (or Hypothesis Class) $\mathcal{H}$

Set of *linear* functions $f: \mathbb{R}^d \to \mathbb{R}$

$$f(x; \theta, \theta_0) = \theta_d x_d + \cdots + \theta_1 x_1 + \theta_0 = \theta^\top x + \theta_0$$

Model Parameters

$$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

Sometimes, we write $\mathcal{L}(\theta; \mathcal{S}_n)$ instead of $\mathcal{L}(f; \mathcal{S}_n)$

Training Loss/Objective

$$\mathcal{L}(f; \mathcal{S}_n) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \frac{1}{2} \big(y - f(x)\big)^2$$

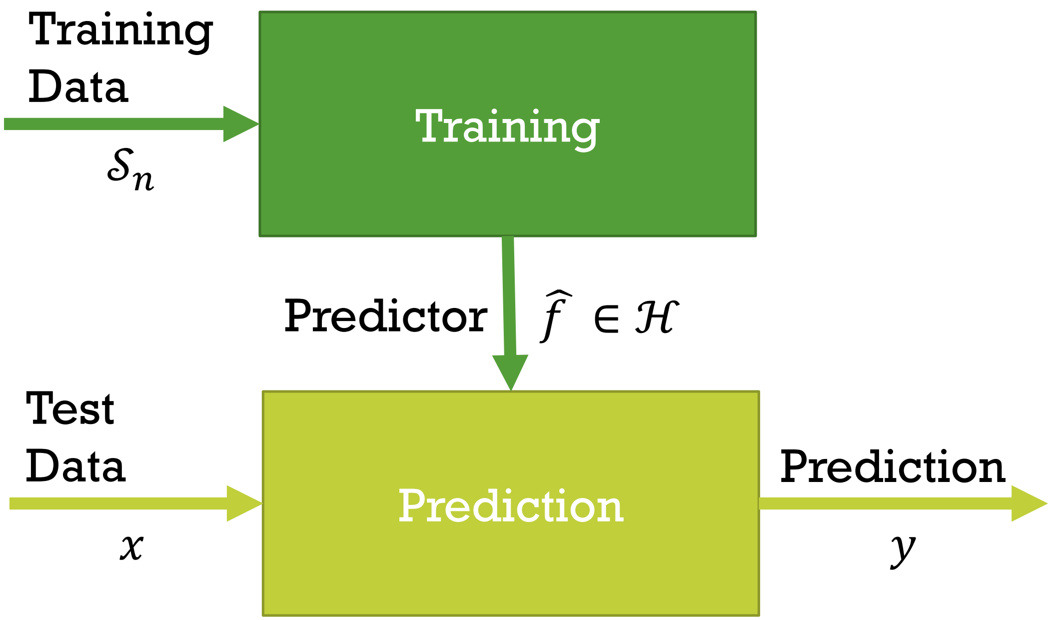
Find predictor $\hat{f} \in \mathcal{H}$ that minimizes $\mathcal{L}(f; \mathcal{S}_n)$.

Training Algorithm

Set gradient to zero, and solve equations.

Training is also sometimes called *Learning*.

LEARNING AND PREDICTION



Assumption. Test data and training data are identically distributed.

GENERALIZATION

The goal of machine learning is to find a predictor $\hat{f} \in \mathcal{H}$ that generalizes well, i.e. that predicts well on test data $\mathcal{S}_*$.

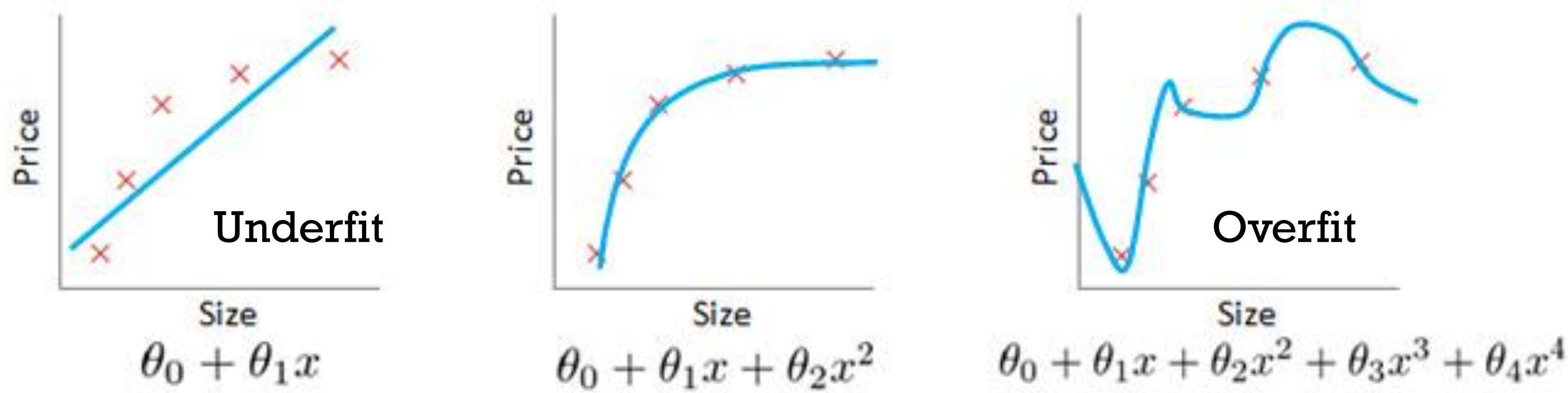
Sometimes, we write $\mathcal{R}(\hat{\theta}; \mathcal{S}_*)$ instead of $\mathcal{R}(\hat{f}; \mathcal{S}_*)$

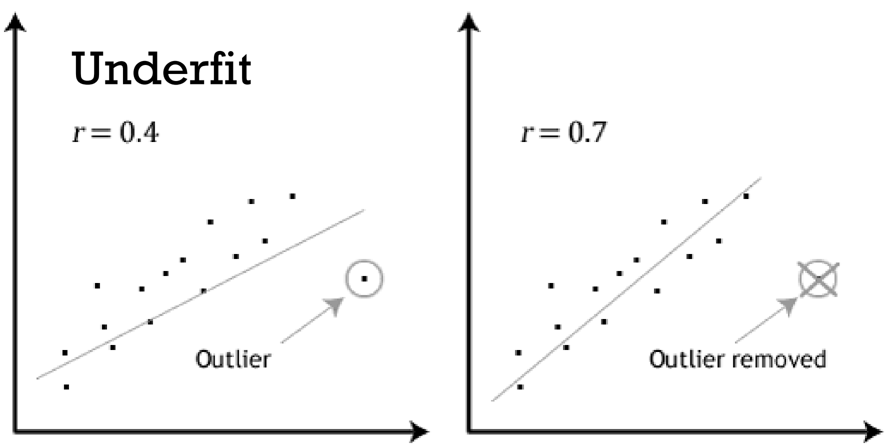
Test Loss/Objective

$$\mathcal{R}(\hat{f}; \mathcal{S}_*) = \frac{1}{n} \sum_{(x,y)\in\mathcal{S}_*} \frac{1}{2}\left(y - \hat{f}(x)\right)^2$$

We often use some test loss $\mathcal{R}(\hat{f}; \mathcal{S}_*)$ to measure how well a predictor $\hat{f}$ generalizes. The test loss can be different from the training loss $\mathcal{L}(f; \mathcal{S}_n)$.

UNDERFITTING AND OVERFITTING





MODEL SELECTION

Overfitting. If model $\mathcal{H}$ is too big, then $\hat{f} \in \mathcal{H}$ performs

- well on training data, but poorly on test data.

Underfitting. If model $\mathcal{H}$ is too small, then $\hat{f} \in \mathcal{H}$ performs

- poorly on training data, and poorly on test data.

Finding a model with the right size is called model selection.

OPTIMIZATION

LOSS AND RISK

Loss Function $\text{Loss}(z) = \frac{1}{2}z^2$

Squared error.
Penalize big errors more heavily.
CONVEX!!

Empirical Risk / Training Loss

$$\mathcal{L}_1(\theta; x, y) = \text{Loss}\big(y - f(x;\theta)\big)$$

Point loss

$$\mathcal{L}_n(\theta; \mathcal{S}_n) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \mathcal{L}_1(\theta; x, y)$$

Average loss

$$= \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \frac{1}{2}\big(y - f(x;\theta)\big)^2$$

Risk = "Expected Loss"
Empirical = "of the Data"

GRADIENT

$$\nabla \mathcal{L}_n(\theta; \mathcal{S}_n) = \begin{pmatrix} \frac{\partial \mathcal{L}_n}{\partial \theta_1}(\theta; \mathcal{S}_n) \\ \frac{\partial \mathcal{L}_n}{\partial \theta_2}(\theta; \mathcal{S}_n) \\ \vdots \\ \frac{\partial \mathcal{L}_n}{\partial \theta_d}(\theta; \mathcal{S}_n) \end{pmatrix} = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \begin{pmatrix} \frac{\partial \mathcal{L}_1}{\partial \theta_1}(\theta; x, y) \\ \frac{\partial \mathcal{L}_1}{\partial \theta_2}(\theta; x, y) \\ \vdots \\ \frac{\partial \mathcal{L}_1}{\partial \theta_d}(\theta; x, y) \end{pmatrix}$$

$$\nabla \mathcal{L}_n(\theta; \mathcal{S}_n) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \nabla \mathcal{L}_1(\theta; x, y)$$

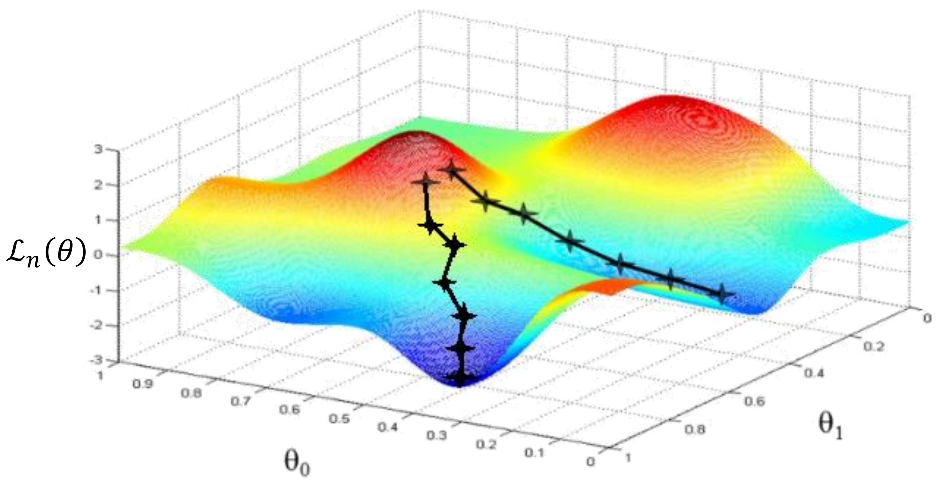
EXACT SOLUTION

If there are no constraints on the parameters,

1. Set the gradient to zero, and solve for the parameters.
2. Run through all the solutions to find the parameter that has the smallest training loss.

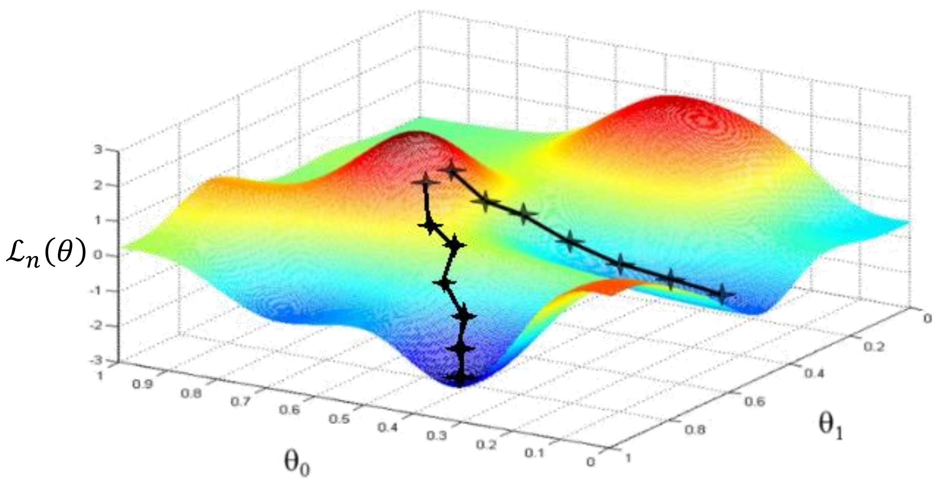
GRADIENT DESCENT

1. Initialize θ randomly.
2. Update $\theta \longleftarrow \theta - \eta_k \, \nabla \mathcal{L}_n(\theta)$, η_k learning rate, k iteration number.
3. Repeat (2) until convergence.
   (e.g. when improvement in $\mathcal{L}_n(\theta)$ is small enough)

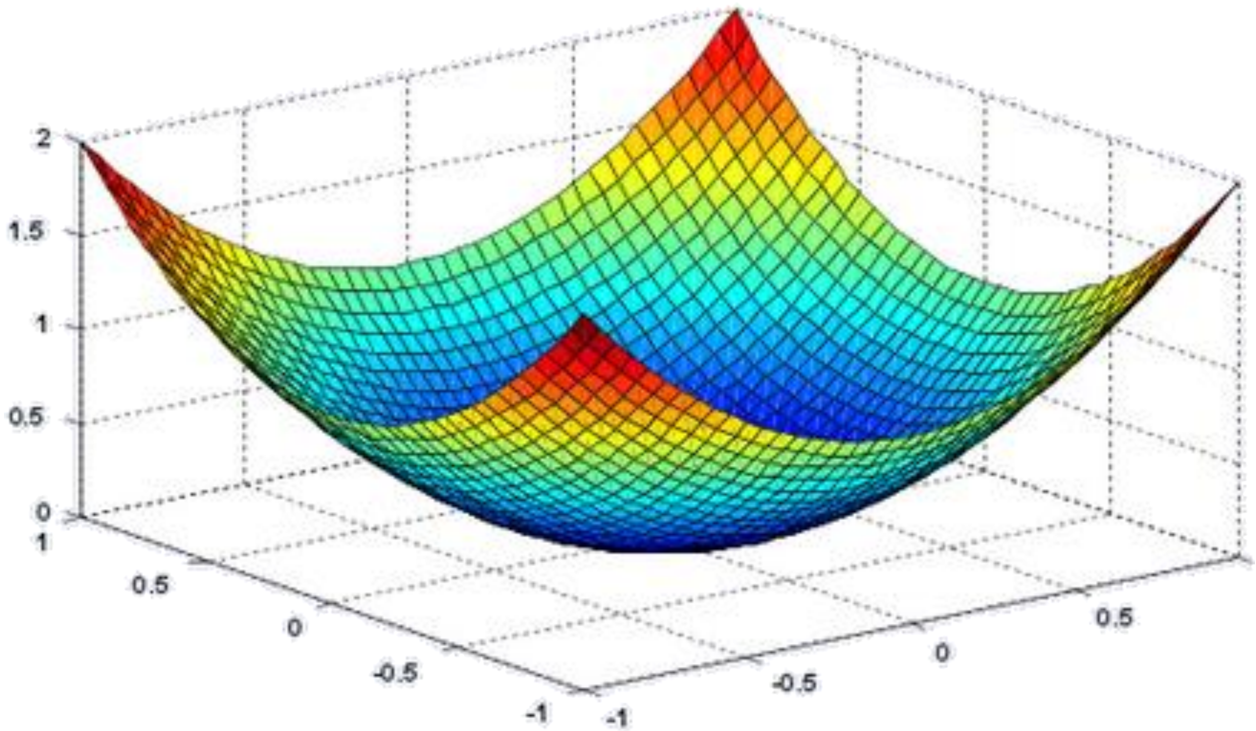


LOCAL MINIMA

- Gradient descent leads us to a local minimum, which is not necessarily the global minimum. Different starting points may lead to different local minima.
- Typically, we perform gradient descent from several starting points, and run through all the local minima to find the parameter that has the smallest training loss.



CONVEX OPTIMIZATION



Local Minimum = Global Minimum.
Fast Algorithms.

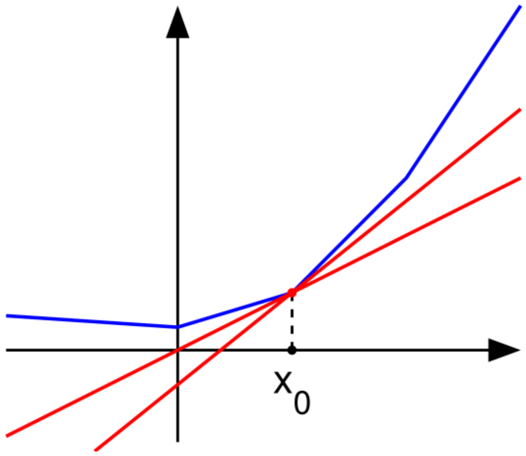
* not in syllabus

SUB-GRADIENTS

- A sub-gradient $v \in \partial f(x)$ is a vector such that for all y,

$$f(y) - f(x) \geq v^\top (y - x).$$

- At non-differentiable points of the training objective function, the gradient does not exist, but we can use any sub-gradient instead for descent.



STOCHASTIC GRADIENT DESCENT

training gradient = average of point gradients

$$\nabla \mathcal{L}_n(\theta; \mathcal{S}_n) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \nabla \mathcal{L}_1(\theta; x, y)$$

This average can take a long time to compute for large data sets.

Trick

Estimate the gradient by averaging over a smaller *minibatch* (subset of the training data).

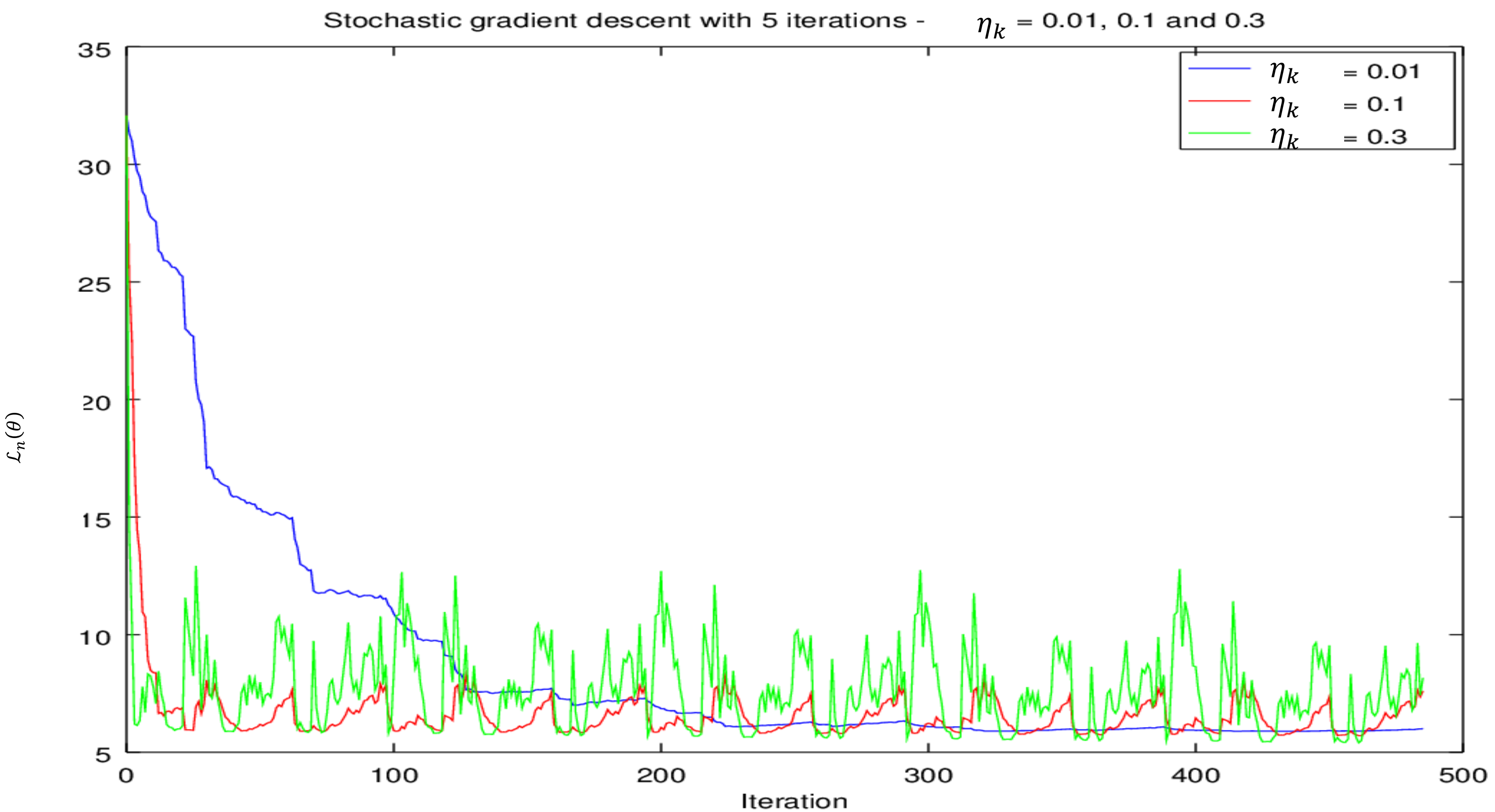
$$\begin{aligned} \nabla \mathcal{L}_n(\theta; \mathcal{S}_n) &\approx \nabla \mathcal{L}_m(\theta; \mathcal{B}_m) \\ &= \frac{1}{m} \sum_{(x,y) \in \mathcal{B}_m} \nabla \mathcal{L}_1(\theta; x, y) \end{aligned}$$

STOCHASTIC GRADIENT DESCENT

1. Initialize θ randomly.
2. Select minibatch $\mathcal{B}_m$ of data from $\mathcal{S}_n$ at random.
   a. $\theta \longleftarrow \theta - \eta_k \nabla \mathcal{L}_m(\theta; \mathcal{B}_m)$.
3. Repeat Step (2) until convergence.

* not in syllabus

STOCHASTIC GRADIENT DESCENT



* not in syllabus

STOCHASTIC GRADIENT DESCENT

Learning Rate

Small learning rates help convergence, but big learning rates speed up descent. We want the best of both worlds, so we choose a learning rate that starts big and ends small, e.g. $\eta_k = 1/(k+1)$.

Momentum

Reduce fluctuations in gradient by taking a weighted sum of the previous update $\Delta^{(t-1)}$ with the current gradient.

$$\theta^{(t+1)} = \theta^{(t)} - \eta_k \Delta^{(t)}, \quad \Delta^{(t)} = (1-\epsilon)\Delta^{(t-1)} + \epsilon\, \nabla \mathcal{L}_m(\theta; \mathcal{B}_m)$$

Software

All these tricks are implemented in the ADAM optimizer.

MULTIVARIATE LINEAR REGRESSION

LEAST SQUARES

Data

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right),\ \ x \in \mathbb{R}^d, y \in \mathbb{R}$$

Model

$$f(x; \theta, \theta_0) = \theta_1 x_1 + \cdots + \theta_d x_d + \theta_0 = \theta^\top x + \theta_0$$

$$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

Training Objective

$$\mathcal{L}_1(\theta, \theta_0; x, y) \ = \ \frac{1}{2}\left(y - (\theta^\top x + \theta_0)\right)^2$$

$$\mathcal{L}_n(\theta, \theta_0; \mathcal{S}_n) \ = \ \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \mathcal{L}_1(\theta, \theta_0; x, y)$$

CONSTANT FEATURE TRICK

Define $x_0 = 1$ and set
$\tilde{x} = (x_d, \dots, x_1, x_0) \in \mathbb{R}^{d+1}$

Data

$$\left(\tilde{x}^{(1)}, y^{(1)}\right), \left(\tilde{x}^{(2)}, y^{(2)}\right), \dots, \left(\tilde{x}^{(n)}, y^{(n)}\right), \ \ \tilde{x} \in \mathbb{R}^{d+1}, y \in \mathbb{R}$$

Model

$$f(x; \theta, \theta_0) = \theta_1 x_1 + \cdots + \theta_d x_d + \theta_0 x_0 = \tilde{\theta}^\top \tilde{x}$$
$$\tilde{\theta} = (\theta, \theta_0) \in \mathbb{R}^{d+1}$$

Training Objective

$$\mathcal{L}_1(\tilde{\theta}; \tilde{x}, y) \qquad = \frac{1}{2}\left(y - \tilde{\theta}^\top \tilde{x}\right)^2$$
$$\mathcal{L}_n(\tilde{\theta}; \mathcal{S}_n) \qquad = \frac{1}{n} \sum\nolimits_{(x,y)\in\mathcal{S}_n} \mathcal{L}_1(\tilde{\theta}; \tilde{x}, y)$$

POINT GRADIENT (ACTIVITY)

Compute the point gradient

$$\nabla \mathcal{L}_1(\theta; x, y) = \begin{pmatrix} \frac{\partial \mathcal{L}_1}{\partial \theta_1}(\theta; x, y) \\ \frac{\partial \mathcal{L}_1}{\partial \theta_2}(\theta; x, y) \\ \vdots \\ \frac{\partial \mathcal{L}_1}{\partial \theta_d}(\theta; x, y) \end{pmatrix}$$

$$\mathcal{L}_1(\theta; x, y) = \frac{1}{2}(y - \theta^\top x)^2$$

$$\mathcal{L}_n(\theta; \mathcal{S}_n) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \mathcal{L}_1(\theta; x, y)$$

POINT GRADIENT

$$\frac{\partial \mathcal{L}_1}{\partial \theta_i}(\theta; x, y) = -x_i\,(y - \theta^\top x)$$

$$\nabla \mathcal{L}_1(\theta; x, y) = \begin{pmatrix} -x_1(y - \theta^\top x) \\ \vdots \\ -x_d(y - \theta^\top x) \end{pmatrix}$$

$$= -\begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}\left(y - \theta^T x\right) = -x(y - \theta^\top x)$$

$$\boxed{\begin{aligned} \mathcal{L}_1(\theta; x, y) &= \tfrac{1}{2}(y - \theta^\top x)^2 \\ \mathcal{L}_n(\theta; \mathcal{S}_n) &= \tfrac{1}{n}\textstyle\sum_{(x,y)\in\mathcal{S}_n} \mathcal{L}_1(\theta; x, y) \end{aligned}}$$

TRAINING GRADIENT (ACTIVITY)

Let $X = \left[x^{(1)}, \ldots, x^{(n)}\right]^\top,\ Y = \left[y^{(1)}, \ldots, y^{(n)}\right]^\top$

Write the training gradient in terms of X, Y.

$$\nabla\mathcal{L}_n(\theta; x, y) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \nabla\mathcal{L}_1(\theta; x, y)$$

Hints.

$$\frac{1}{\mathrm{n}}\sum_{t=1}^{n} x^{(t)}y^{(t)} = \frac{1}{n}\left[x^{(1)}, \ldots, x^{(n)}\right]\left[y^{(1)}, \ldots, y^{(n)}\right]^\top = \frac{1}{n}X^\top Y$$

$$\theta^\top x = x^\top\theta$$

$$\boxed{\begin{gathered}\mathcal{L}_1(\theta; x, y) = \frac{1}{2}(y - \theta^\top x)^2 \\ \mathcal{L}_n(\theta; \mathcal{S}_n) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}_n} \mathcal{L}_1(\theta; x, y)\end{gathered}}$$

TRAINING GRADIENT

$$\nabla \mathcal{L}_n(\theta) = \frac{1}{n}\sum_{\text{data }(x,y)} -x(y - \theta^\top x)$$

$$= \frac{1}{n}\sum_{\text{data }(x,y)} -xy + x(\theta^\top x)$$

$$= \frac{1}{n}\sum_{\text{data }(x,y)} -xy + x(x^\top \theta) = -B + A\theta$$

where

$$B = \frac{1}{\mathrm{n}}\sum_{t=1}^{n} x^{(t)}y^{(t)} = \frac{1}{n}\left[x^{(1)}, \dots, x^{(n)}\right]\left[y^{(1)}, \dots, y^{(n)}\right]^\top = \frac{1}{n}X^\top Y$$

$$\mathrm{A} = \frac{1}{\mathrm{n}}\sum_{t=1}^{n} x^{(t)}x^{(t)\top} = \frac{1}{n}\left[x^{(1)}, \dots, x^{(n)}\right]\left[x^{(1)}, \dots, x^{(n)}\right]^\top = \frac{1}{n}X^\top X$$

GRADIENT DESCENT

$$\theta \longleftarrow \theta - \eta_k \left[\frac{1}{n} (X^\top X)\theta - \frac{1}{n} X^\top Y \right]$$

See Homework 1.

Bonus. Stochastic Gradient Descent.

EXACT SOLUTION

Optimization problem is convex, so the minimum is attained when the gradient is zero.

$$\nabla \mathcal{L}_n(\hat{\theta}) = 0 \quad \Leftrightarrow \quad \frac{1}{n}(X^\top X)\,\hat{\theta} = \frac{1}{n} X^\top Y$$
$$\Leftrightarrow \quad \hat{\theta} = (X^\top X)^{-1} X^\top Y$$

Issues.

1. Need $X^\top X$ to be invertible
   - Feature vectors $x^{(1)}, \dots, x^{(n)}$ must span $\mathbb{R}^d$
   - Must have more data than features, $n \geq d$
   - Use regularization if $X^\top X$ is not invertible
2. What if $X^\top X \in \mathbb{R}^{d \times d}$ is a large matrix?
   - Takes long time to invert
   - Use stochastic gradient descent if $X^\top X$ is too large

REGULARIZATION

RIDGE REGRESSION

For simplicity, we ignore θ_0.

Weight Age Temp. on Mars

Height $y \approx \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$

How do we ensure that $\theta_k = 0$ when feature x_k is irrelevant?

Pick simplest model that explains data ➔ **generalization**

RIDGE REGRESSION

Add a penalty.

Pressure to fit data

Pressure to go to zero

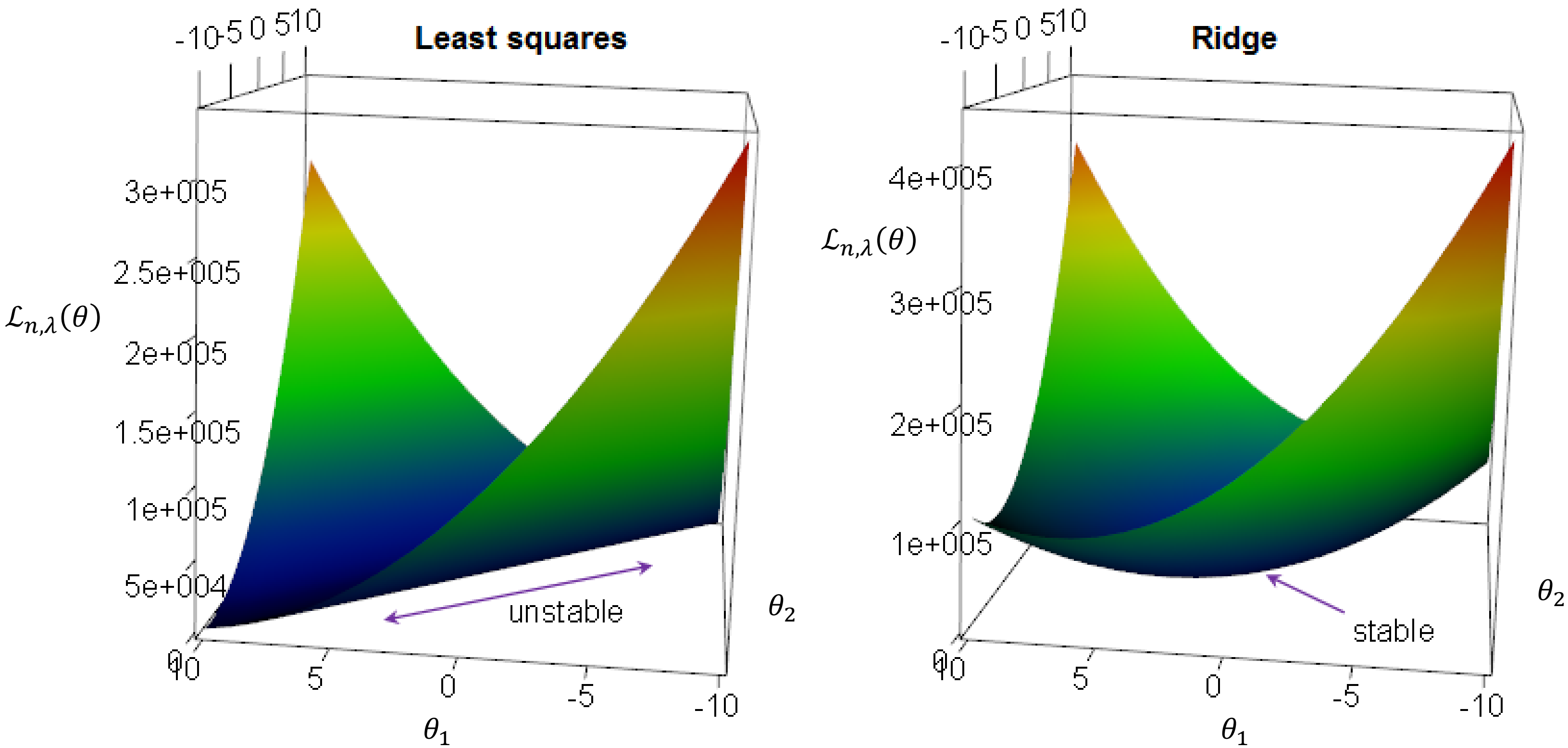
$$\mathcal{L}_{n,\lambda}(\theta) = \frac{1}{n}\sum\nolimits_{\text{data }(x,y)} \frac{1}{2}(y-\theta^\top x)^2 + \frac{\lambda}{2}\|\theta\|^2$$

Regularization parameter $\lambda \geq 0$

Regularizer

(Unfortunately, to include the parameter θ_0, we cannot simply apply the constant feature trick. Why?)

RIDGE REGRESSION



TRAINING ALGORITHMS

Gradient

$$\nabla \mathcal{L}_{n,\lambda}(\theta) = \lambda\theta + \frac{1}{n}(X^\top X)\theta - \frac{1}{n}X^\top Y$$

Exact Solution

$$\nabla \mathcal{L}_{n,\lambda}(\hat{\theta}) = 0 \quad \Leftrightarrow \quad \lambda\hat{\theta} + \frac{1}{n}(X^\top X)\,\hat{\theta} = \frac{1}{n}X^\top Y$$

$$\Leftrightarrow \quad \hat{\theta} = (n\lambda I + X^\top X)^{-1}X^\top Y$$

This matrix is always invertible when $\lambda > 0$.

TRAINING ALGORITHMS

Gradient

$$\nabla \mathcal{L}_{n,\lambda}(\theta) = \lambda\theta + \frac{1}{n}(X^\top X)\theta - \frac{1}{n}X^\top Y$$

Gradient Descent

$$\theta \longleftarrow (1 - \eta_k \lambda)\,\theta - \eta_k \left[\frac{1}{n}(X^\top X)\theta - \frac{1}{n}X^\top Y\right]$$

Without regularization, i.e. $\lambda = 0$, this shrinkage factor equals 1.

TRAINING LOSS VS TEST LOSS

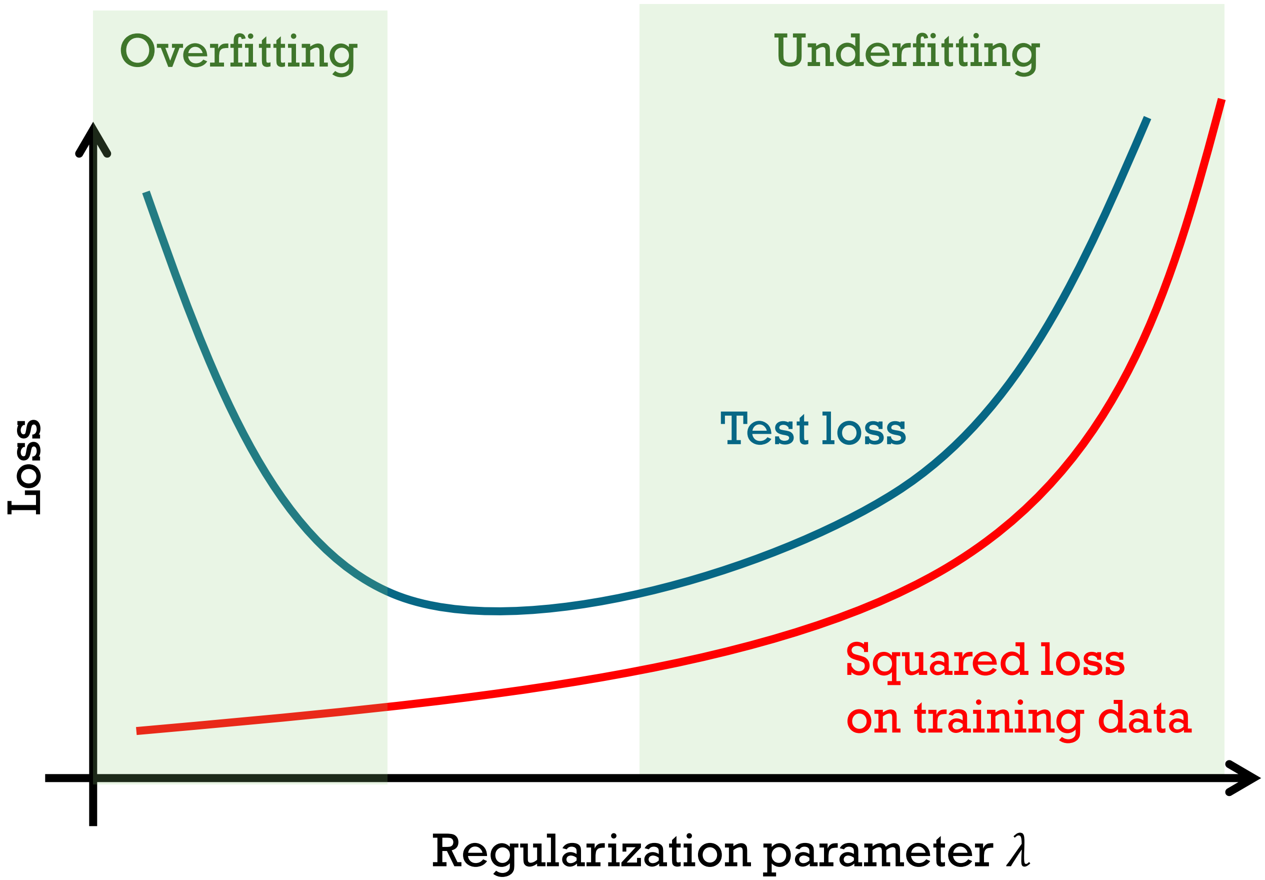
Training Loss

$$\mathcal{L}_{n,\lambda}(\theta) = \frac{1}{n} \sum_{\boxed{\text{trg data } (x,y)}} \frac{1}{2} (y - \theta^\top x)^2 + \boxed{\frac{\lambda}{2} \|\theta\|^2}$$

Test Loss

$$\mathcal{R}(\theta) = \frac{1}{n} \sum_{\boxed{\text{test data } (x,y)}} \frac{1}{2} (y - \theta^\top x)^2$$

EFFECT OF REGULARIZATION



* not in syllabus

SYMBOLIC COMPUTATION

* not in syllabus

AUTOMATIC DIFFERENTIATION

In optimization, we often need to differentiate the objective function by hand to obtain the gradients for a descent algorithm.

Wouldn't it be nice if this step was automated? :)

Let us work through an example in Theano. Similar ideas are used in other software packages such as Google's TensorFlow.

SUMMARY

- Methodology
  - Features, Response
  - Training, Prediction
  - Training, Test Data
  - Model, Hypothesis, Parameters
  - Training, Test Loss
  - Training Algorithm
  - Generalization
  - Underfitting, Overfitting
  - Model Selection
- Optimization
  - Loss Functions
  - Empirical Risk
  - Exact Solution
  - Gradient Descent
  - Convex Optimization
  - Stochastic Gradient Descent

SUMMARY

- Multivariate Linear Regression
  - Model
  - Training Loss
  - Constant Feature Trick
  - Gradient
  - Gradient Descent
  - Exact Solution
  - Issues with Exact Solution
- Regularization
  - Generalization
  - Ridge Regression
  - Regularizer
  - Regularization Parameter
  - Exact Solution and Invertibility
  - Gradient Descent and Shrinkage
  - Training Loss vs Test Loss
  - Effect on Test Loss

INTENDED LEARNING OUTCOMES

Methodology

- Given a machine learning example, identify the components:
  - Features, Response
  - Training, Prediction
  - Training data, Test data
  - Model, Hypothesis, Parameters
  - Training loss, Test loss
  - Training algorithm
- State that the goal of machine learning is generalization.
- Give an example of underfitting, overfitting and model selection.

INTENDED LEARNING OUTCOMES

Optimization

- Give examples of loss functions, and define empirical risk in terms of the loss function.
- List two general types of algorithms used in optimization, e.g. exact solution, and gradient descent. Outline the broad steps involved in each of them.
- Explain why framing a problem as convex optimization is highly desirable, in terms of speed and local minima.
- Explain the motivation behind performing stochastic gradient descent, rather than traditional gradient descent.

INTENDED LEARNING OUTCOMES

Multivariate Linear Regression

- State the model and the training loss.
- Explain how the 'constant feature' trick can be used to reduce the problem to one without the constant parameter θ_0.
- Describe two training algorithms that may be applied.
- Derive the gradient of the training loss.
- Derive the formula for the exact solution.
- Describe two potential weaknesses of the exact solution, and possible solutions for these weaknesses.
- Apply the above algorithms to a given data set.

INTENDED LEARNING OUTCOMES

Regularization

- Explain why regularization can help with generalization.
- State the training loss and test loss in ridge regression.
- Identify the regularizer and regularization parameter in the training loss of a given machine learning problem.
- Explain why regularization solves the invertibility problem in traditional linear regression.
- Describe the difference in gradient descent between traditional and regularized linear regression.
- Describe how the test loss and training loss varies with the regularization parameter.