Learning via Optimization

Lecture 7

Outline

1. Optimization
   – Convexity
2. Linear regression in depth
   – Locally weighted linear regression
3. Brief dips
   – Logistic Regression
      • [Stochastic] gradient ascent/descent
   – Support Vector Machines (SVM)
      • Kernel trick
   – Neural Networks
      • Backpropagation

What is Optimization?

(and why do we care?)

General Definition

$$\begin{aligned} \underset{x}{\text{minimize}} \;& f_0(x) \\ \text{s.t.} \;& f_i(x) \leq 0, i = \{1, \ldots k\} \\ & h_j(x) = 0, j = \{1, \ldots l\} \end{aligned}$$

Why Do We Care?

- Optimization is at the heart of many/most modern machine learning algorithms

- Example: Linear Regression

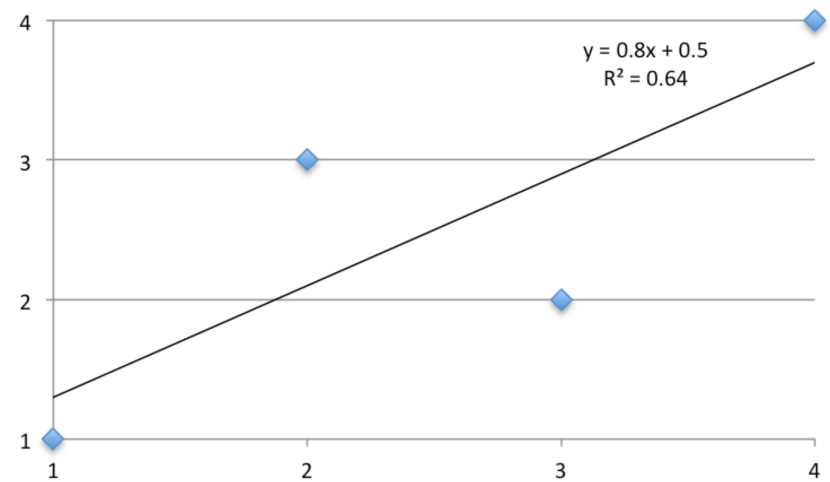
$$\underset{w}{\text{minimize}} \; ||Xw - y||^2$$

Linear Regression

Input

x	y
1	1
2	3
3	2
4	4

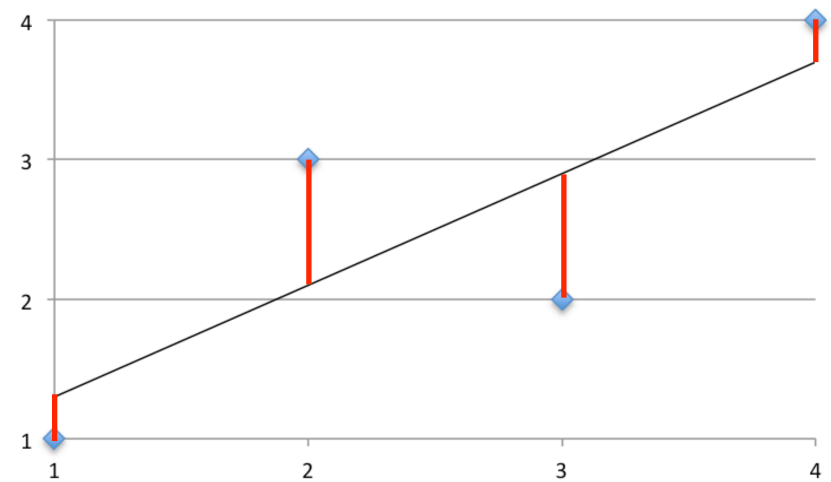
Output



Linear Regression as Optimization

- Why this line?
  - Minimizer **error**

- In 2D, the algorithm tries to find a slope and intercept that yields the smallest sum of the square of the error (SSE)



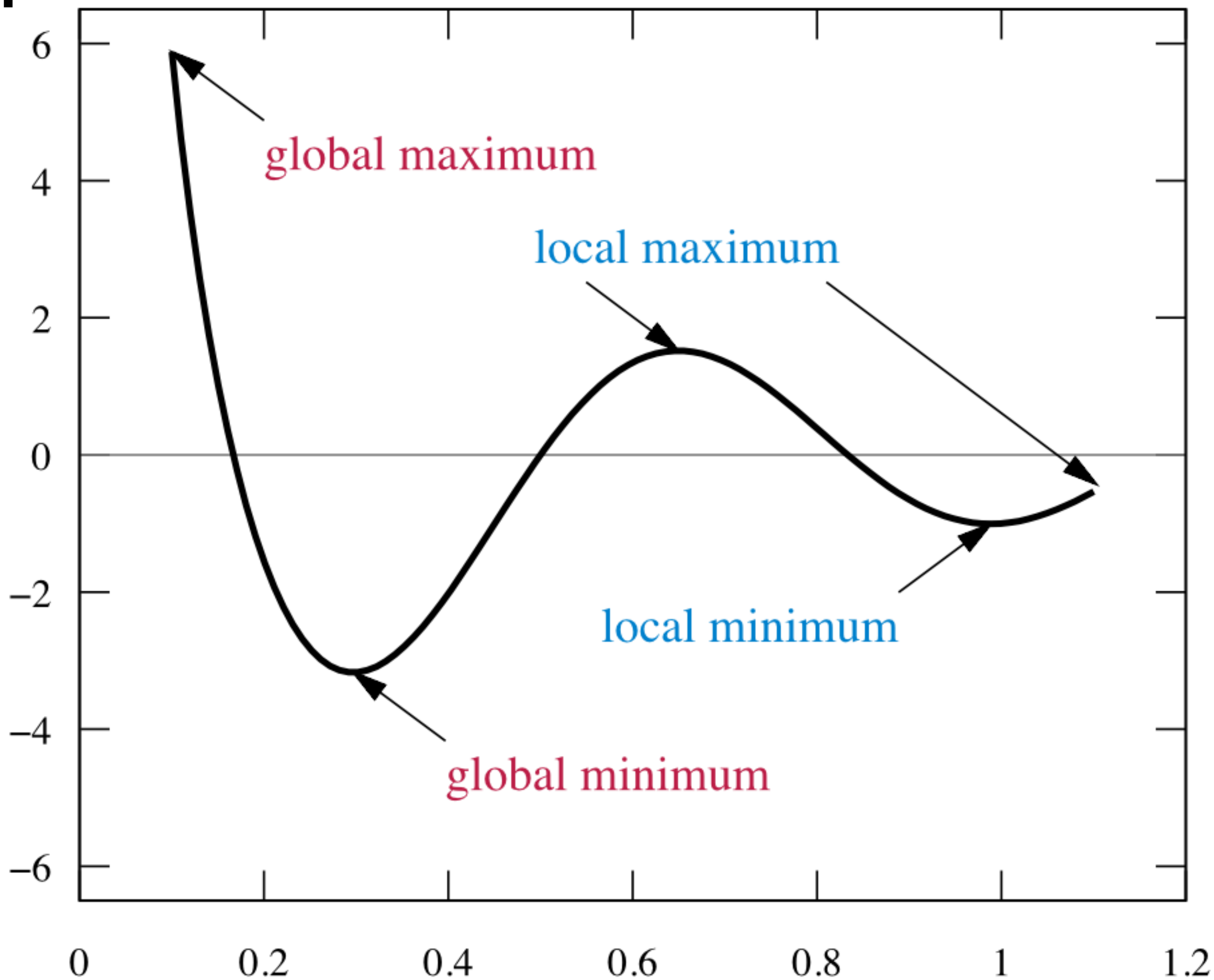
$$\arg\min_{m,b} \sum_{i=1}^{N} e_i^2 = (y_i - (mx_i + b))^2$$

Machine Learning via Optimization

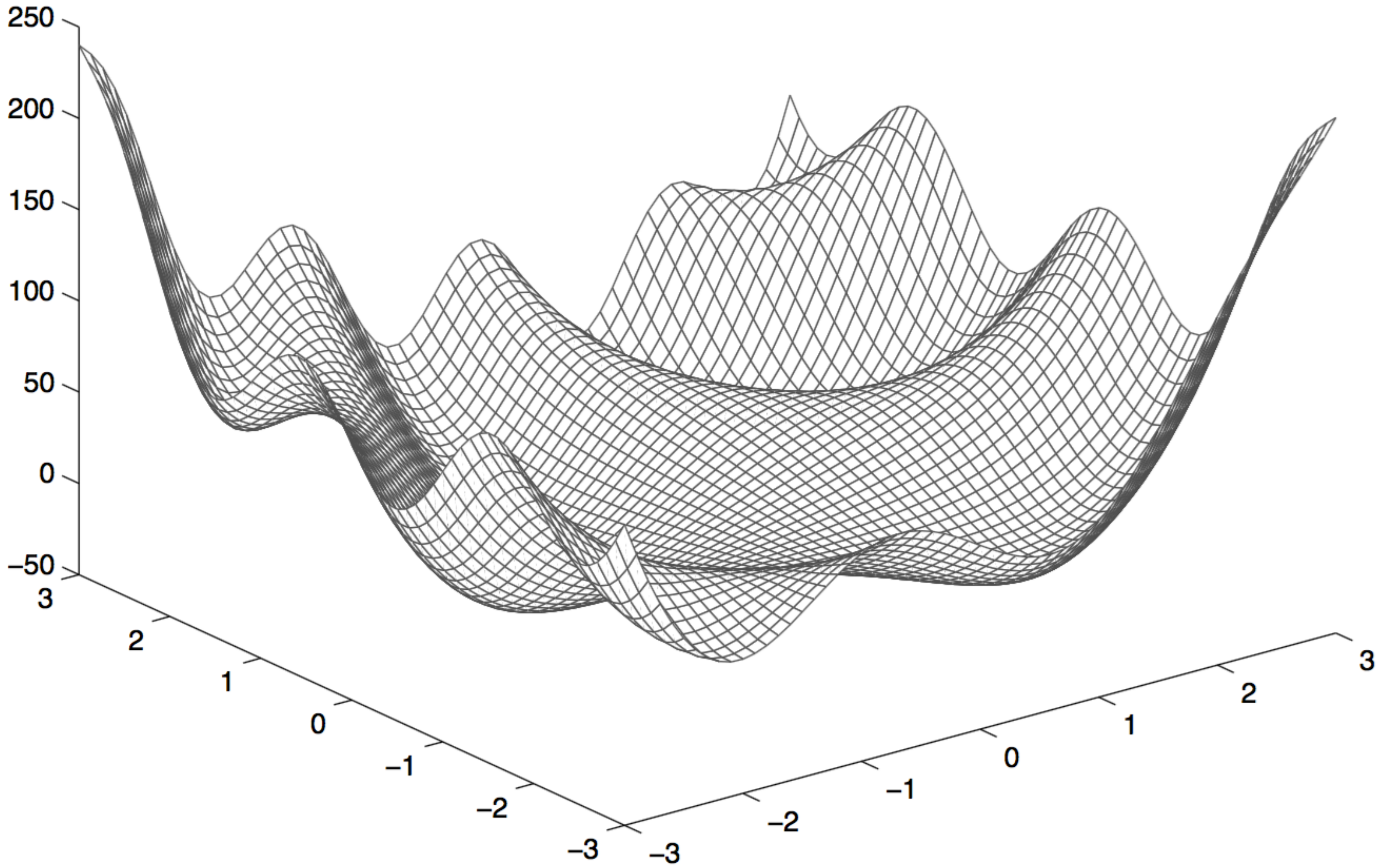
1. Define an error function
2. Find model parameters that minimize the error function given the data
   - Sometimes closed-form solution (e.g. linear)
   - Sometimes [iterative] solution [with guarantees] (e.g. convex)
   - Most of the time will require approximation
     - Iteration (limited by number, delta)
     - Softening constraints
     - Post-processing

     …

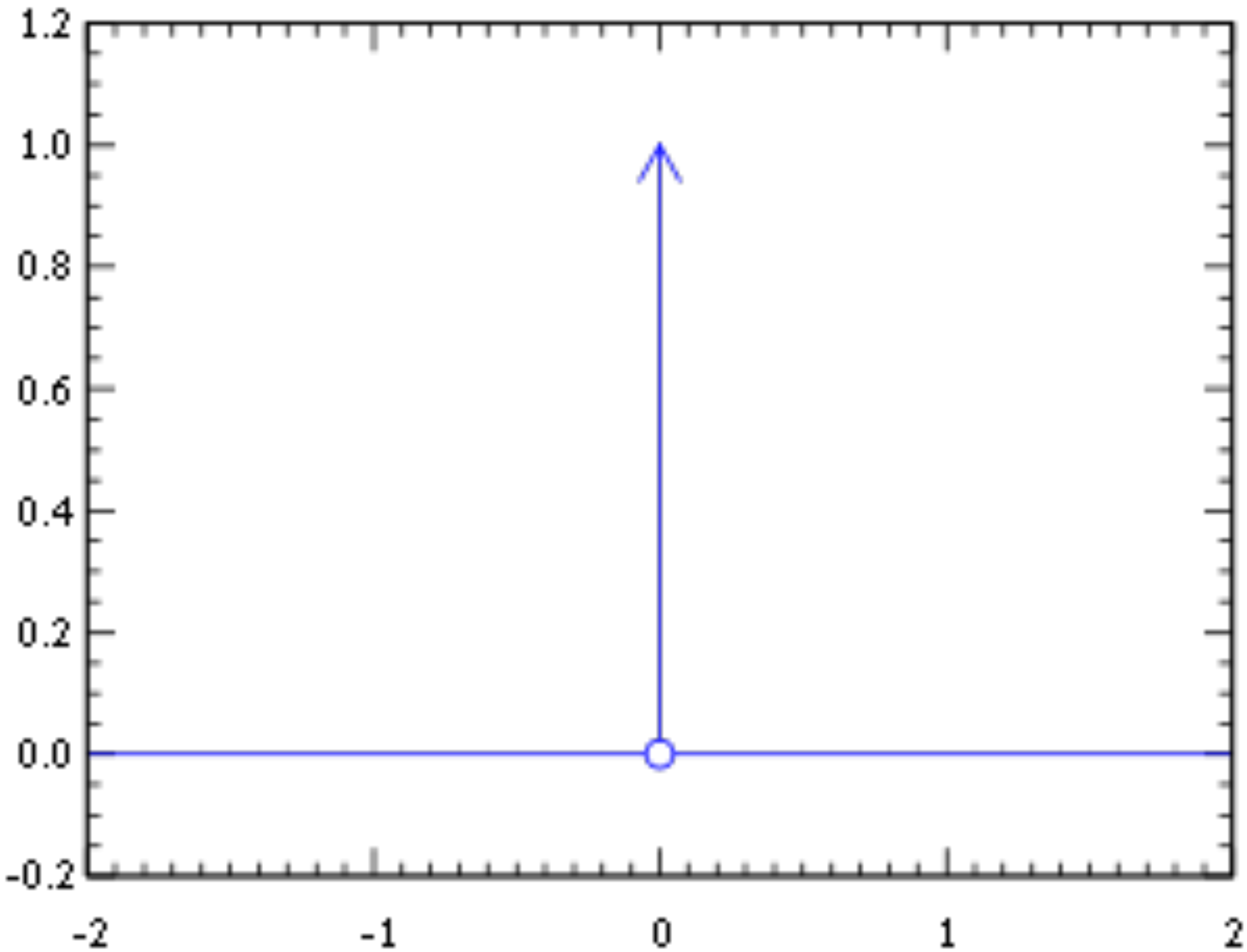
Optimization is Hard in General



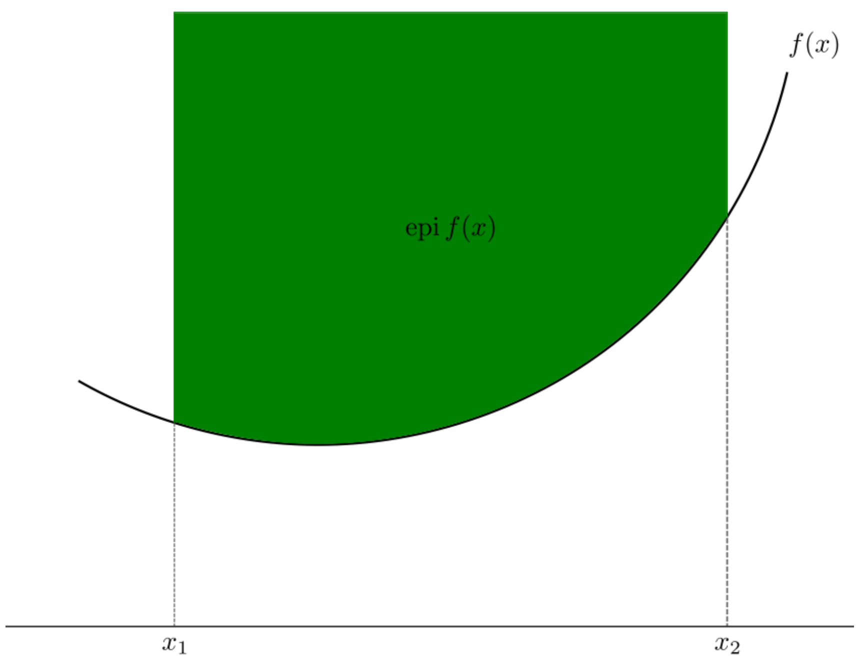
Consider Many [Cursed] Dimensions



Consider Discontinuities

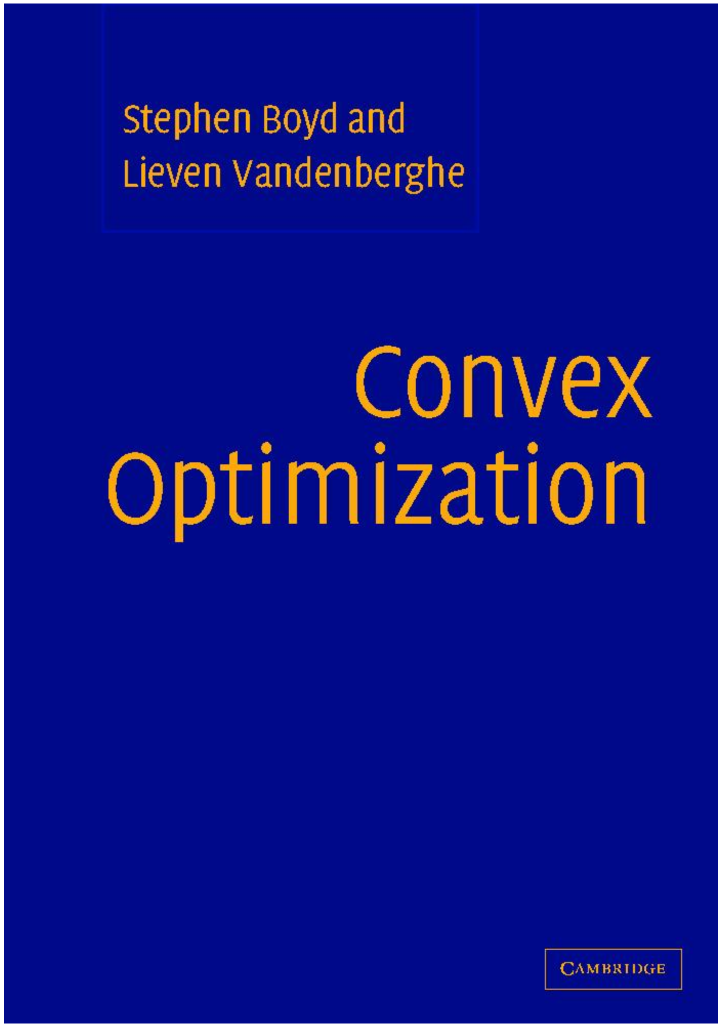


Optimizing Convex Functions is Easy



- The line segment between any two points on the graph of the function lies above or on the graph
- No more than one minimum (might be zero in an open set)
- Specialized algorithms exist to solve numerically (intuition: just go downhill!)

Plenty More About Convex Optimization



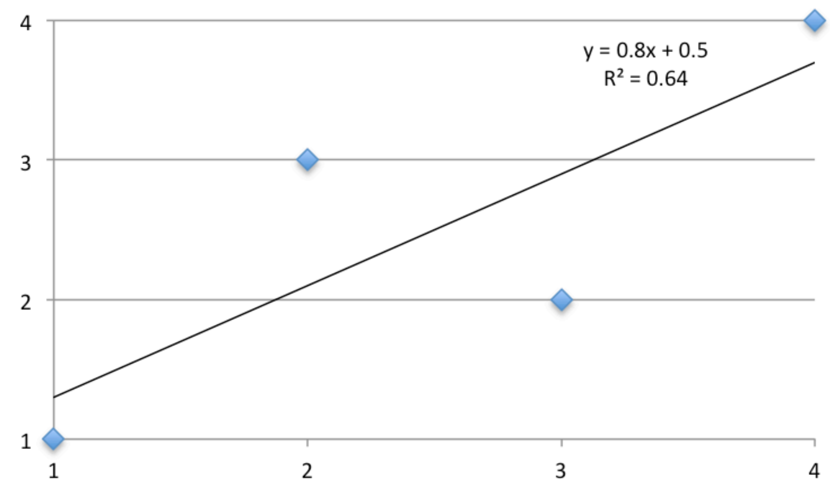
- Properties
- Analysis; how to prove if a set/function is convex
- Methods

http://stanford.edu/~boyd/cvxbook/

Linear Regression

Recipe

1. Define error function
2. Find parameter values that minimize error given the data

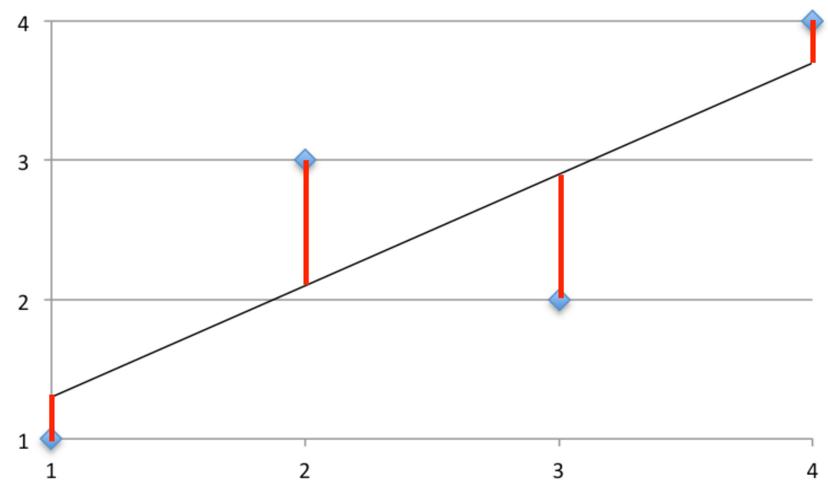


Error Function

Sum of Squared Error (SSE)

aka Residual Sum of Squares (RSS)

$$\mathrm{SSE}_{\mathrm{line}} = \sum_{i=1}^{N} (y_i - f(x_i)) = (y_i - (mx_i + b))^2$$



Algebra (1)

$$\begin{aligned}
\mathrm{SSE}_{\mathrm{line}} &= \sum_{i=1}^{N} (y_i - (mx_i + b))^2 \\
&= \sum_{i=1}^{N} {y_i}^2 - 2y_i(mx_i + b) + (mx_i + b)^2 \\
&= \sum_{i=1}^{N} {y_i}^2 - 2mx_iy_i - 2by_i + m^2{x_i}^2 + 2mbx_i + b^2
\end{aligned}$$

Algebra (2)

$$\sum_{i=1}^{N} a_i = N\overline{a}$$

so…

$$\text{SSE}_{\text{line}} = \sum_{i=1}^{N} {y_i}^2 - 2mx_iy_i - 2by_i + m^2{x_i}^2 + 2mbx_i + b^2$$
$$= N\overline{y^2} - 2Nm\overline{xy} - 2Nb\overline{y} + Nm^2\overline{x^2} + 2Nmb\overline{x} + Nb^2$$

Recall: Critical Points

- For a differentiable function of several variables, a critical point is a value in its domain where all partial derivatives are zero

- So to find the point at which error is minimized, we take partial derivatives of the error function w.r.t. the parameters, set these equal to 0, solve

Calculus (1)

$$\text{SSE}_{\text{line}} = N\overline{y^2} - 2Nm\overline{xy} - 2Nb\overline{y} + Nm^2\overline{x^2} + 2Nmb\overline{x} + Nb^2$$

$$\frac{\partial \text{SSE}_{\text{line}}}{\partial m} = -2N\overline{xy} + 2Nm\overline{x^2} + 2Nb\overline{x} = 0$$

$$\frac{\partial \text{SSE}_{\text{line}}}{\partial b} = -2N\overline{y} + 2Nm\overline{x} + 2Nb = 0$$

Algebra (3)

$$\frac{\partial \mathrm{SSE}_{\mathrm{line}}}{\partial b} = -2N\overline{y} + 2Nm\overline{x} + 2Nb = 0$$

$$0 = -2N\overline{y} + 2Nm\overline{x} + 2Nb$$

$$0 = -\overline{y} + m\overline{x} + b$$

$$\overline{y} = m\overline{x} + b \qquad (\overline{x}, \overline{y})$$

Algebra (4)

$$\frac{\partial \mathrm{SSE}_{\mathrm{line}}}{\partial m} = -2N\overline{xy} + 2Nm\overline{x^2} + 2Nb\overline{x} = 0$$

$$0 = -2N\overline{xy} + 2Nm\overline{x^2} + 2Nb\overline{x}$$

$$0 = -\overline{xy} + m\overline{x^2} + b\overline{x}$$

$$\overline{xy} = m\overline{x^2} + b\overline{x}$$

$$\frac{\overline{xy}}{\overline{x}} = m\frac{\overline{x^2}}{\overline{x}} + b$$

$$\left(\frac{\overline{x^2}}{\overline{x}}, \frac{\overline{xy}}{\overline{x}}\right)$$

And Finally…

$$(\overline{x}, \overline{y})$$

$$(\frac{\overline{x^2}}{\overline{x}}, \frac{\overline{xy}}{\overline{x}})$$

$$m = \frac{\frac{\overline{xy}}{\overline{x}} - \overline{y}}{\frac{\overline{x^2}}{\overline{x}} - \overline{x}}$$

$$= \boxed{\frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2}}$$

$$\overline{y} = m\overline{x} + b$$

$$b = \boxed{\overline{y} - m\overline{x}}$$

A Quick Aside: Meaning of Slope (1)

- **Covariance**. A measure of how much two random variables change together

$$\mathrm{Cov}(X,Y) = \sigma(X,Y) = \mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]$$

  - If both X/Y increase relative to their means, positive; else negative

$$\mathrm{Cov}(X,X) = \mathrm{Var}(X)$$

Meaning of Slope (2)

$$\begin{aligned}\mathrm{Cov}(X,Y) = \sigma(X,Y) &= \mathrm{E}[(X-\mathrm{E}[X])(Y-\mathrm{E}[Y])] \\ &= \mathrm{E}[XY - X\mathrm{E}[Y] - Y\mathrm{E}[X] + \mathrm{E}[X]\mathrm{E}[Y]] \\ &= \mathrm{E}[XY] - \mathrm{E}[X\mathrm{E}[Y]] - \mathrm{E}[Y\mathrm{E}[X]] + \mathrm{E}[\mathrm{E}[X]\mathrm{E}[Y]] \\ &= \mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y] - \mathrm{E}[X]\mathrm{E}[Y] + \mathrm{E}[X]\mathrm{E}[Y] \\ &= \mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y]\end{aligned}$$

Meaning of Slope (3)

Given data, we can approximate the expected value of a random variable by the sample mean

$$E[A] \approx \overline{A}$$

And Finally…

$$\begin{aligned}\mathrm{Cov}(X,Y) &= \mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y] \\ &= \overline{XY} - \bar{X}\bar{Y}\end{aligned}$$

But remember…

$$\begin{aligned}m &= \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} \\ &= \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{xx} - \bar{x}\bar{x}}\end{aligned}$$

$$\begin{aligned}m &= \frac{\mathrm{Cov}(X,Y)}{\mathrm{Cov}(X,X)} \\ &= \frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(X)}\end{aligned}$$

Evaluating Linear Regression

The natural question to ask: to what extent is the line capturing the variation in *y* as a result of the variation in *x*

- To quantify: look at the ratio of the error of the line and the error of *y*

$$\mathrm{R}^2 = 1 - \frac{\mathrm{SSE}_{\mathrm{line}}}{\mathrm{SSE}_{\overline{\mathrm{Y}}}}$$

$$\mathrm{SSE}_{\overline{\mathrm{Y}}} = (y_1 - \overline{Y})^2 + (y_2 - \overline{Y})^2 + \ldots$$

The Multi-Dimensional Case

- The preceding discussion assumed a single independent variable (x), and thus derived a single slope (m) and intercept to linearly approximate the dependent variable (y)

- We now consider the multi-dimensional case, where each independent variable (x_i) is associated with a slope/intercept

Problem Setup

We begin with an analogous representation

$$Y = XB + e$$

where…

- **Y** is N x 1
- **X** is N x (k+1); extra 1 to multiply intercept
- **B** is (k+1) x 1; first intercept, then coefficients
- **e** is N x 1

k-Dimensional Linear Regression

$$
\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}
=
\begin{bmatrix} 1 & x_{11} & x_{12} & \cdot & \cdot & \cdot & x_{1k} \\ 1 & x_{21} & x_{22} & \cdot & \cdot & \cdot & x_{2k} \\ \cdot & \cdot & & & & & \\ \cdot & \cdot & & & & & \\ \cdot & \cdot & & & & & \\ 1 & x_{N1} & x_{N2} & \cdot & \cdot & \cdot & x_{Nk} \end{bmatrix}
\begin{bmatrix} b \\ m_1 \\ m_2 \\ \cdot \\ \cdot \\ \cdot \\ m_k \end{bmatrix}
+
\begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ \cdot \\ e_N \end{bmatrix}
$$

Step 1: Error Function

- We will use the same error method as last time, which is SSE (i.e. square the difference between Y and XB)

$$\begin{aligned}\text{SSE} &= e^{\intercal} e \\ &= (Y - XB)^{\intercal}(Y - XB)\end{aligned}$$

Matrix Algebra (1)

$$
\begin{aligned}
\mathrm{SSE} &= (Y - XB)^\intercal (Y - XB) \\
&= (Y^\intercal - B^\intercal X^\intercal)(Y - XB) \\
&= Y^\intercal Y - Y^\intercal XB - B^\intercal X^\intercal Y + B^\intercal X^\intercal XB \\
&= Y^\intercal Y - 2Y^\intercal XB + B^\intercal X^\intercal XB
\end{aligned}
$$

Matrix Calculus (1)

$$\text{SSE} = Y^\intercal Y - 2Y^\intercal XB + B^\intercal X^\intercal XB$$

$$\frac{\partial \text{SSE}}{\partial B} = -2X^\intercal Y + 2X^\intercal XB$$

Matrix Algebra (2)

$$0 = -2X^\intercal Y + 2X^\intercal XB$$

$$-2X^\intercal XB = -2X^\intercal Y$$

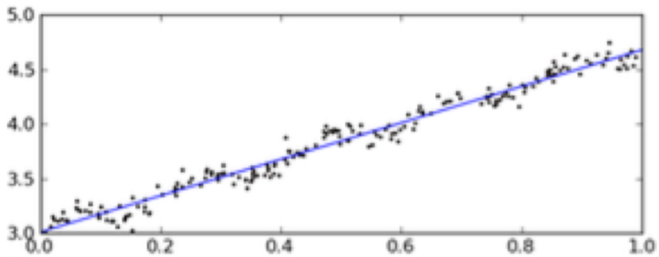
$$X^\intercal XB = X^\intercal Y$$

$$B = (X^\intercal X)^{-1} X^\intercal Y$$

Look familiar?

Local Weighting

- One weakness of linear regression is that it weights all data points equally



- **Locally Weighted Linear Regression** (LWLR) introduces weights for each data point, allowing near points to "count more" than distal points

LWLR

- Weights computed as …

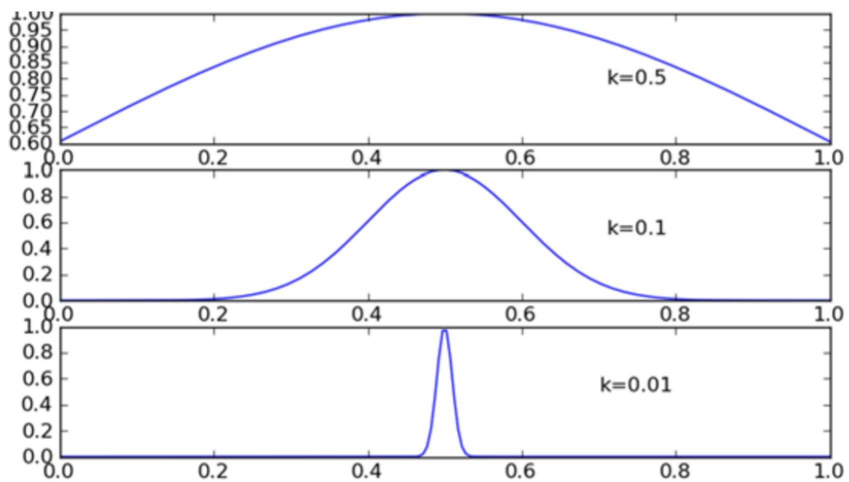
$$B = (X^{\intercal} W X)^{-1} X^{\intercal} W Y$$

- Where W is an arbitrary weight matrix (all non-diagonal elements are zero); common to use a Gaussian weighting kernel

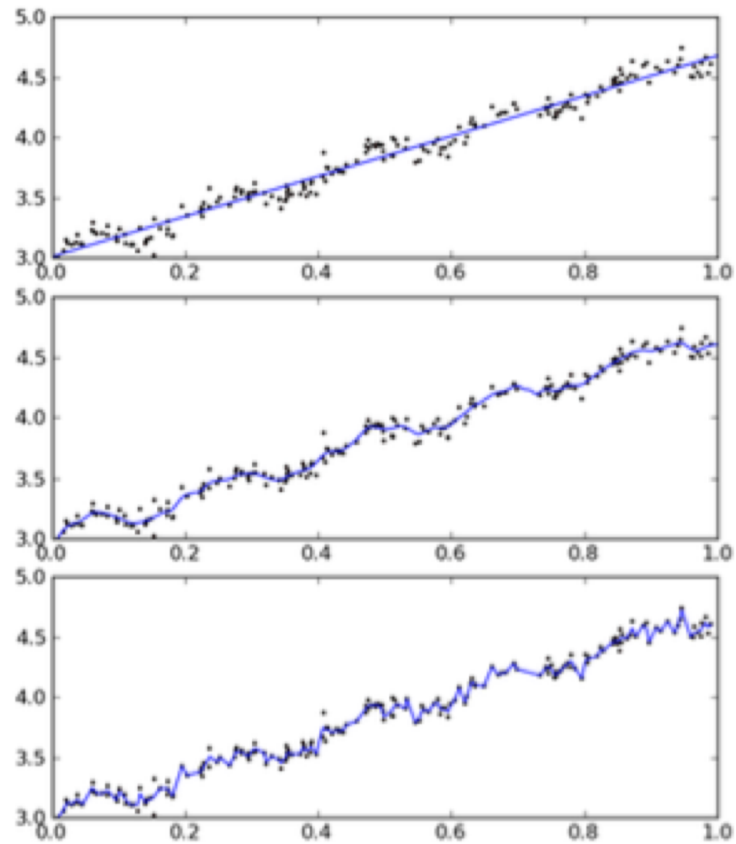
$$W(i,i) = e^{\frac{||x_i - x_0||}{-2k^2}}$$

LWLR: Changing *k*

Weight Matrix



Model



Linear Regression: Other Issues

- More features than data points
  - Computing the inverse without a full-rank matrix
- Shrinking coefficients/regularization
  - Ridge regression, the Lasso
- Uncertainty in X

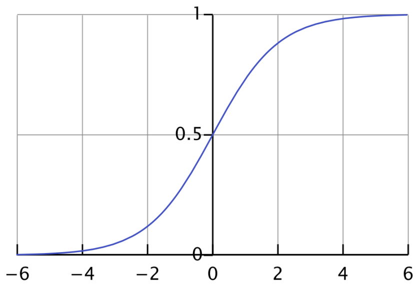
Other Optimization-based ML Methods

- Logistic Regression
- Support Vector Machines (SVMs)
- Neural Networks

Logistic Regression

- Despite the name, the goal is **binary classification**

- The goal: find a set of weights to <u>optimally</u> transform input data to either side of a logistic/sigmoid function (later: why this fn)



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

Basic Idea

- Take input data, multiply by a weight vector, compute output of sigmoid
  - If $\sigma(w^{\intercal}x) \geq 0.5$, output 1; else 0
    - The function is usefully bounded (vs. LR)
  - The sigmoid forms a **decision boundary**

- But what determines the "optimal" set of weights? An **error function**.

Logistic Regression: Error

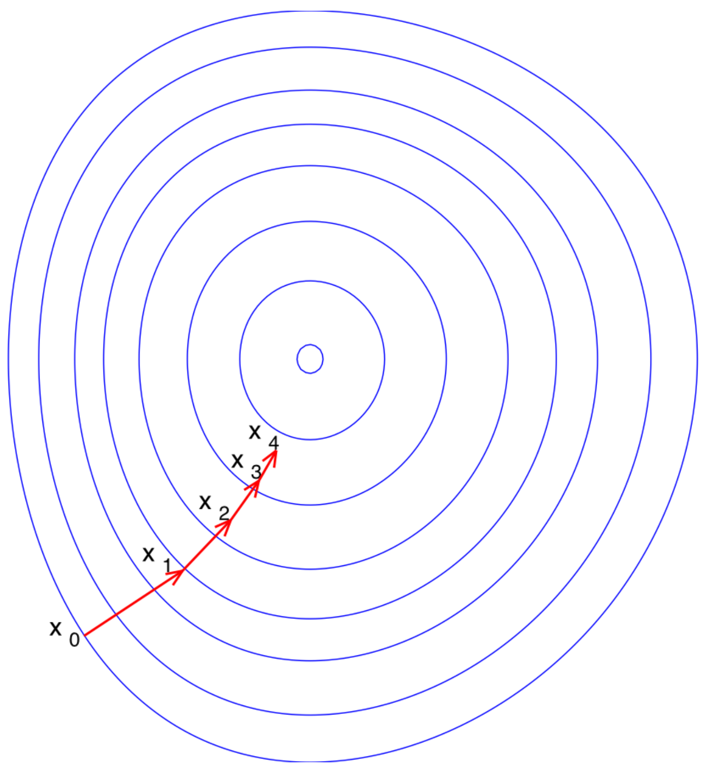
$$-y\log(\sigma(w^\intercal x)) - (1-y)\log(1-\sigma(w^\intercal x))$$

Intuition: if not correct, large smooth value

- Based on MLE
- Convex!

Gradient Descent

- Simple, iterative optimization algorithm
- Intuition: at each point, move a proportional step in the direction of the gradient



$$w := w + \alpha \nabla_w f(w)$$

Logistic Regression: Pseudocode

- Start with weights = 1
- Loop
  - h = sigmoid(x * weights)
  - error = (labels – h)
  - weights = weights + alpha*x*error

- Notes
  - Gradient (hence sigmoid fn): $(\sigma(w^\intercal x) - y)X$
  - Iterate while improving
  - Step size is an issue (small=slow, big=chaos)
    - There is some theory; depends on the problem

Stochastic Gradient Descent

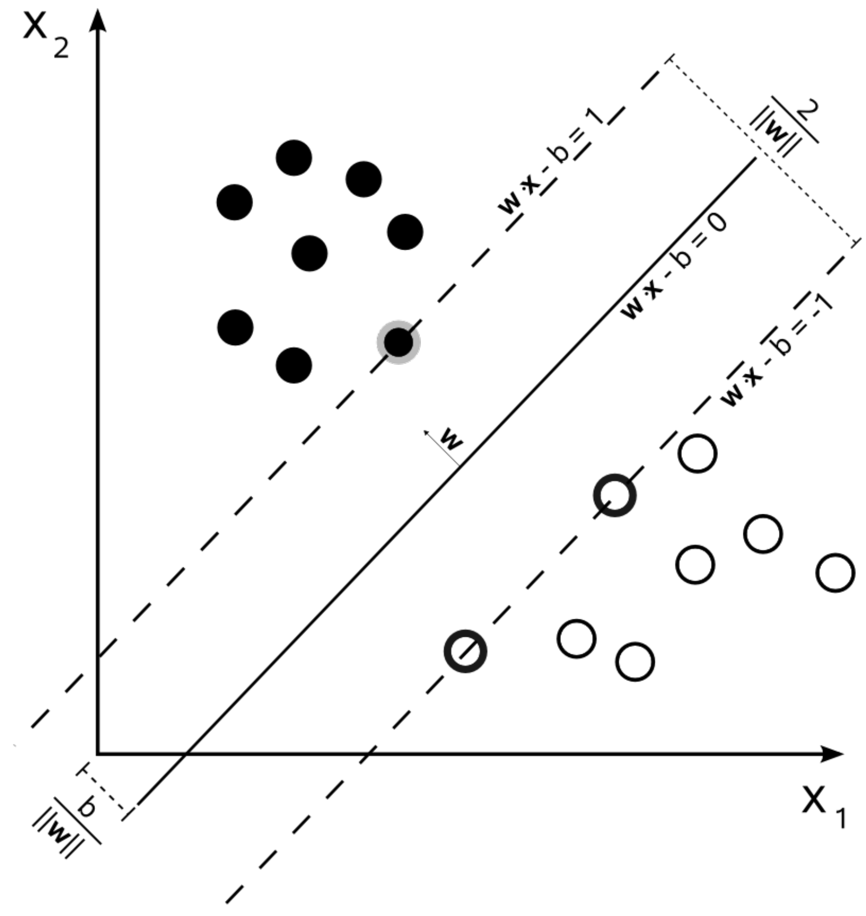
- Given large numbers of examples, gradient descent is typically not feasible
- Stochastic gradient descent uses only a single example each iteration, shuffling between passes
- Mini-batch is a compromise to take advantage of vectorization libraries

Support Vector Machines (SVM)

- Big picture: learn a separating hyperplane (if one exists) that maximizes the distance from it to the nearest data point on each side (the "support vectors")

- Derivation involves a good deal of advanced theory/ methods (e.g. Lagrange multipliers, slack variables)

- Standard off-the-shelf classifier (e.g. libSVM)

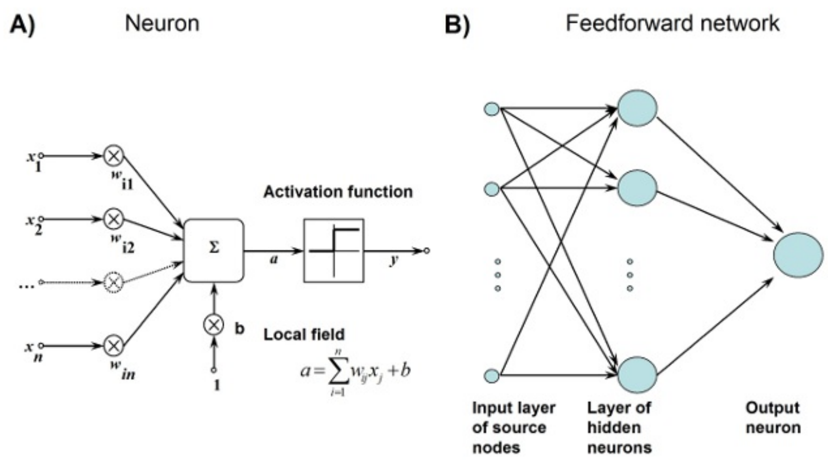


Kernel Trick

- Kernel methods require only a user-specified kernel (i.e. a similarity function) over pairs of data points in raw representation
  - Allows reasoning in a higher dimensional space without having to explicitly compute coordinates
- In SVM: Radial Basis Function (RBF)
  - Allows SVM to learn non-linear hyperplanes

Neural Networks

- Input nodes are connected to other nodes via weights
- Weights are summed, and then filtered through an activation function
- Training involves using errors from output neurons to update weights
  - The **backpropagation** algorithm computes the gradient of the error function, which is then combined with an optimization algorithm (e.g. stochastic gradient descent) to incrementally update weights



Summary

- Optimization is a crucial component for modern machine learning algorithms
- To begin, define an error function, and then optimize this function with respect to input data
- Most interesting problems will not have closed-form solutions, and will require iterative and/or approximate optimization techniques
  – If the error function is convex, hill-climbing methods like [stochastic] gradient descent work well