Machine Learning EN.601.475/675

DR. PHILIP GRAFF

Logistic Regression

FROM REGRESSION TO CLASSIFICATION

Classification

Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ $\mathbf{x}_i \in \mathbb{R}^M$ $y_i \in L$ ***L*** is a set of labels

Learn: a mapping from **x** to discrete value y

- f(**x**) = y

Examples:

- Spam classification
- Document topic classification
- Identifying faces in images

Binary Classification

- We'll focus on binary classification: $y_i \in \{0, 1\}$

  sometimes... $y_i \in \{-1, 1\}$

- Usually easy to generalize to multi-class classification

Different Definition

Fitting a function to data

Fitting: Optimization, what parameters can we change?

Function: Model, loss function

Data: Data/model assumptions? How we use data?

ML Algorithms: minimize a function on some data

Evaluation

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N}$$

Other measurements appropriate for some tasks

- Ex. we care more about certain types of mistakes

Regression

Least squares regression

- Outputs real number for each example

It seems that classification should be easier!

Let's use regression for classification

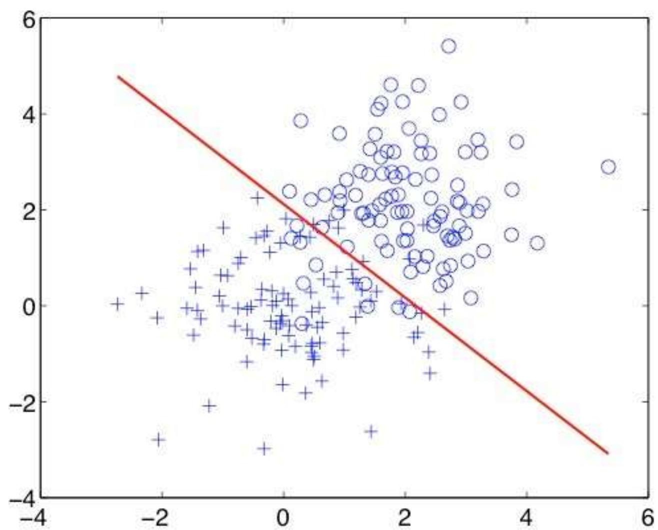
- Learn least squares regression model $f_w(x)=y$
  - $f_w(x)=w^T x$
- If $y>0$, predict "True (1)"
- If $y \leq 0$ predict "False (0)"

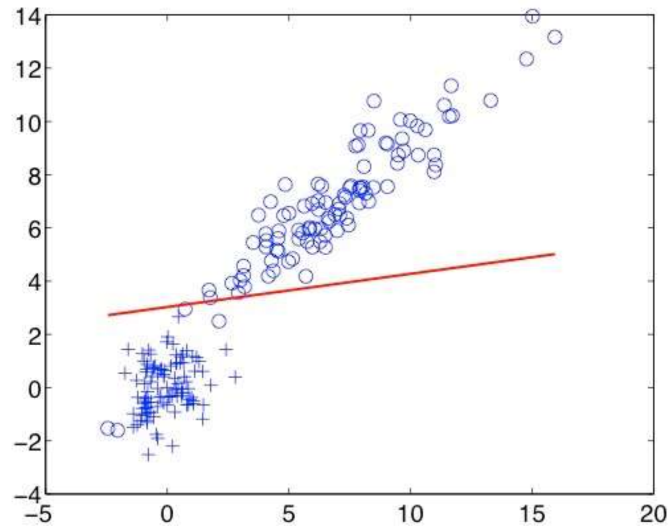
Regression for Classification

$f_w(x)=0$ partitions the input space into two class specific regions

- Linear decision boundary

Good





Bad

Regression for Classification

Mismatch between regression loss and classification

- Classification: accuracy
- We don’t care about large vs. small values of output

Outliers problematic

- Prediction of 42 for example is fine for purposes of classification, bad for regression

We need output to be either 1 or 0

Machine Learning

Fitting a function to data

Fitting: Solve for w given y and x

Function: Regression uses squared loss

- Bad match for our task!

Data: assume dependent variable linear combination of independent variables

Our loss function doesn't match classification goals

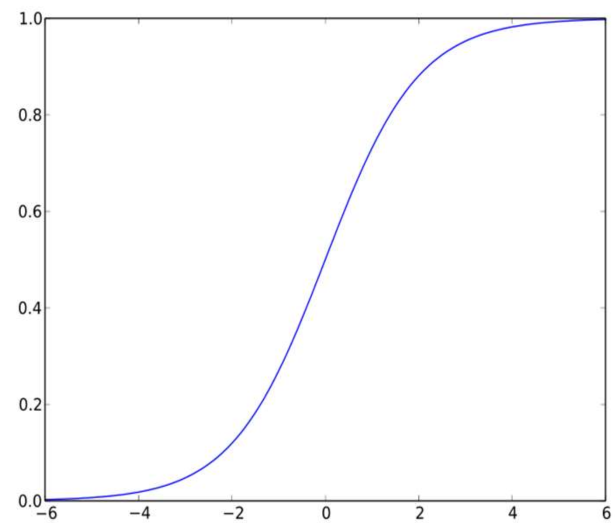
Logistic Function

Quick fix: apply a function to the output that gives the desired value

Logistic function: $$g_\alpha(x) = \frac{1}{1 + e^{-\alpha x}}$$

- ❖ Outputs are between 0 and 1
- ❖ Scaling parameter α

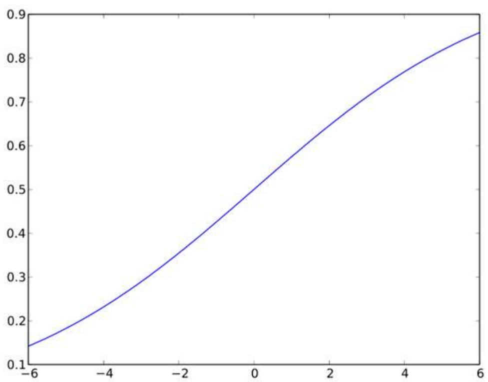
➢ Most outputs are close to 1 or 0



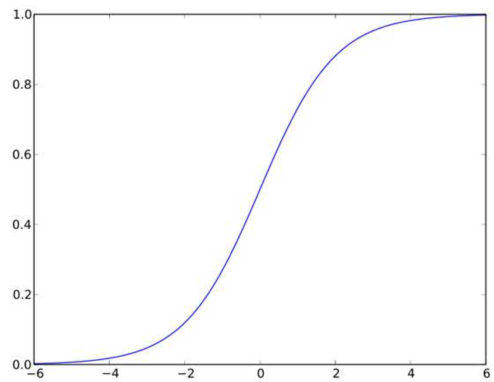
$\alpha = 1$

Logistic Function

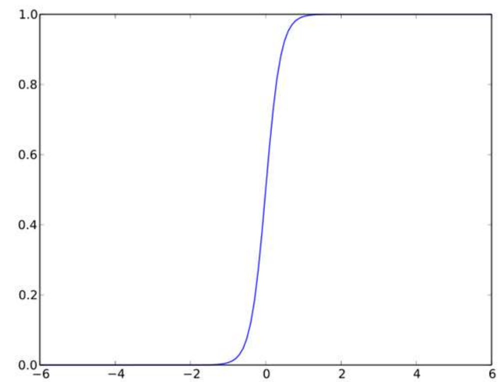
$$g_\alpha(x) = \frac{1}{1 + e^{-\alpha x}}$$



$\alpha = 0.3$



$\alpha = 1$



$\alpha = 5$

Logistic Regression

We can combine the logistic function and our regression model

$$g(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

Notice: as $\mathbf{w}^T\mathbf{x}$ becomes:

- Large: output closer to 1
- Small: output closer to 0

Probabilistic View

In regression we modeled probability of the output – Likelihood

Probability of the example and classification?

- p(x,y)?
- Since we know x, we want to maximize y
- p(x,y) = p(x|y)p(y) = p(y|x)p(x)
- Since p(x) is fixed:

$$\underset{y=0,1}{\operatorname{argmax}} \left(p(x|y)p(y)\right) = \underset{y=0,1}{\operatorname{argmax}} \left(p(y|x)\right)$$

Why?

We can now write the distribution as $$p_{\mathbf{w}}(y=1|\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

Which implies that $$p_{\mathbf{w}}(y=0|\mathbf{x}) = \frac{e^{-\mathbf{w}^T\mathbf{x}}}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

The odds of the event is then $$\frac{p_{\mathbf{w}}(y=1|\mathbf{x})}{p_{\mathbf{w}}(y=0|\mathbf{x})} = e^{\mathbf{w}^T\mathbf{x}}$$

And the log-odds are $$\log\left(\frac{p_{\mathbf{w}}(y=1|\mathbf{x})}{p_{\mathbf{w}}(y=0|\mathbf{x})}\right) = \mathbf{w}^T\mathbf{x}$$

Generalized Linear Models

Decision boundary/surface

- An n-1 dimensional hyper-plane that separates the data into two groups
- These are linear functions of x, even though logistic is not linear

Generalized linear models

- A linear model whose output is passed through non-linear function

Hypothesis class

- Linear decision boundaries

Discriminative Model

We just care about $p(y|x)$ so we can *discriminate* between classes

How does this differ from what we have done before?

Logistic Regression Decisions

Given parameters w, how do we make predictions?

$$p_{\mathbf{w}}(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

If output > 0.5, predict 1, else predict 0

In addition to prediction, we have confidence in prediction

- Confidence is the probability of the prediction

Logistic Regression

Fitting a function to data

Fitting: Solve for **w** given y and **x**

Function: Generalized linear function – logistic over regression

Data: Assume dependent variable linear combination of independent variables, modulated by the logistic function

Objective Function: Likelihood

Conditional data likelihood

$$p_{\mathbf{w}}(Y|\mathbf{X}) = \prod_{i=1}^{N} p_{\mathbf{w}}(y_i|\mathbf{x}_i)$$

$$\mathcal{L}(Y, \mathbf{X}, \mathbf{w}) = \log(p_{\mathbf{w}}(Y|\mathbf{X})) = \sum_{i=1}^{N} \log(p_{\mathbf{w}}(y_i|\mathbf{x}_i))$$

Logistic Regression

Fitting a function to data

Fitting: Solve for w given y and x

Function: Generalized linear function – logistic over regression – conditional log likelihood

Data: assume dependent variable linear combination of independent variables

Function Optimization

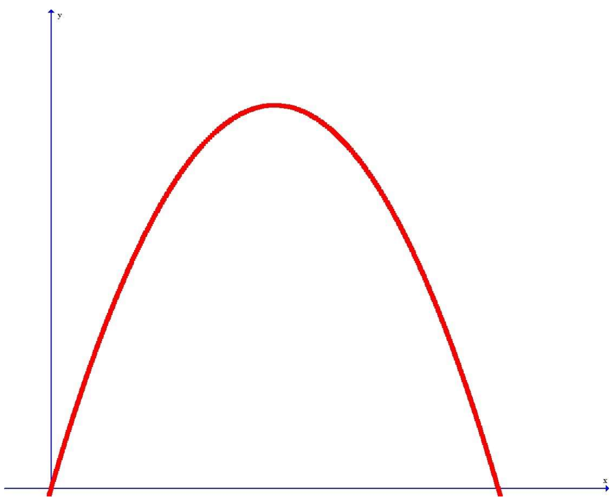
We have a function and want to maximize/minimize it

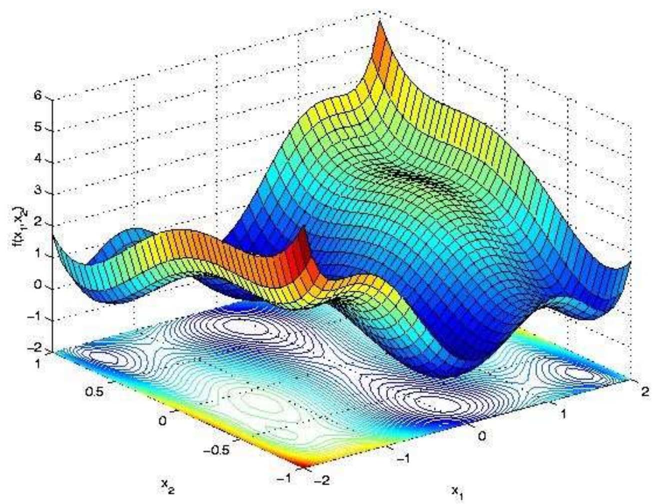
How do we find the point at which the function reaches its max/min?

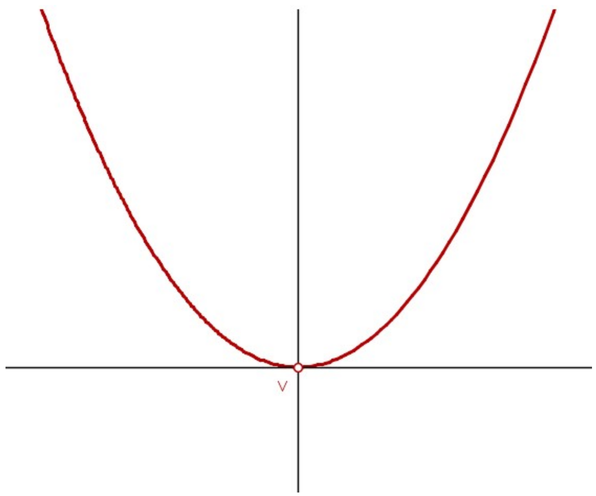
Function Optimization

Take the derivative, set it equal to 0, solve!

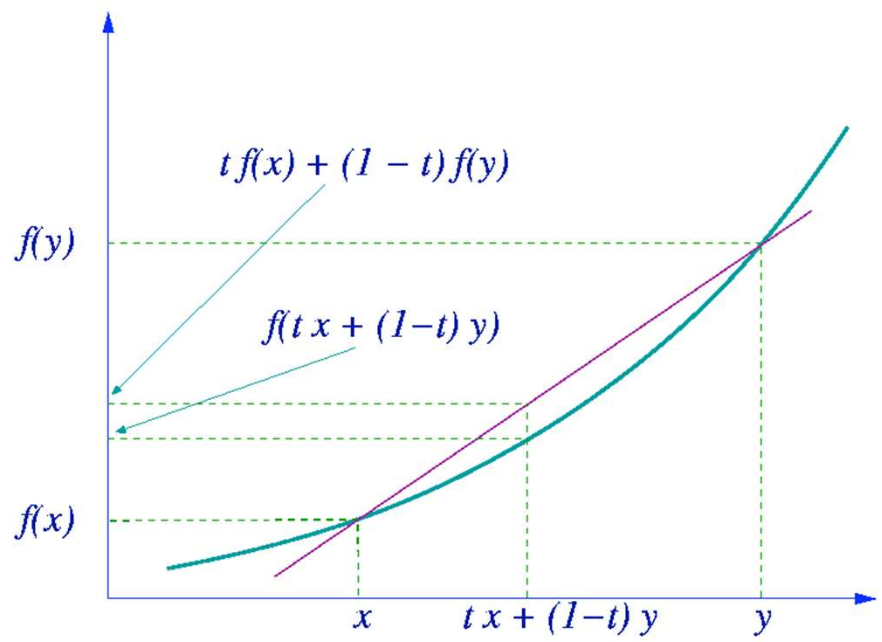
Will this work for every function?







Convex Functions



Maximum Likelihood Estimation (MLE)

Find the value at which the likelihood is maximized

- We'll talk about other options later in the semester

Given the conditional log likelihood

- Take the derivatives for parameters **w**
- Set each derivative to 0
- M equations and M variables
- Solve for **w**

Problem

- No closed form (analytical) solution for **w**

Convex Optimization

The conditional maximum likelihood is concave

- There is a *single* maximal solution

We can maximize using convex optimization techniques

- Its easy to optimize convex functions
- There are **many** convex optimization algorithms

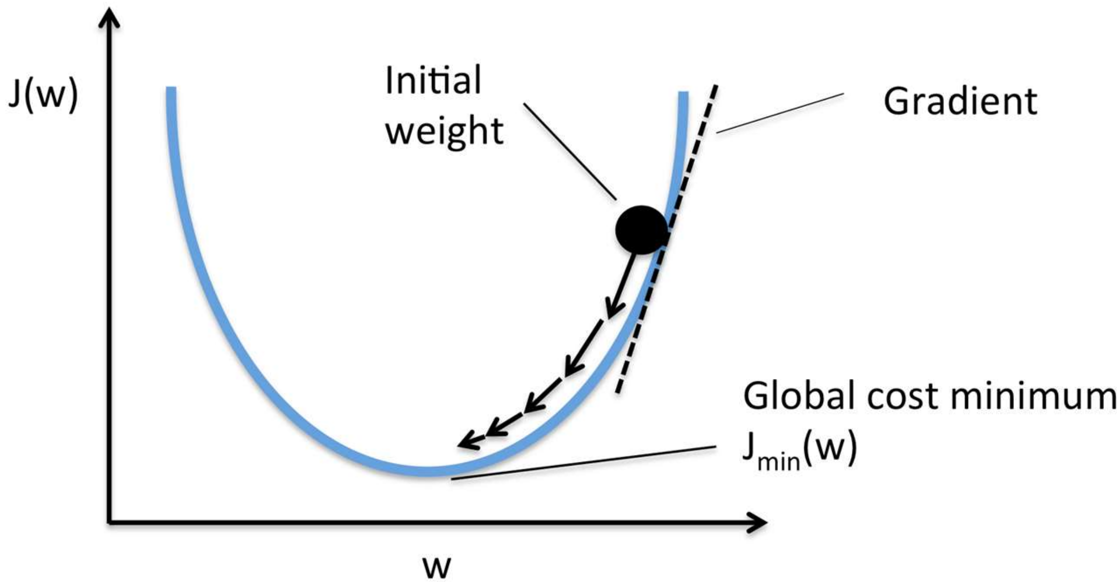
Gradient Descent

First order method: needs first order derivatives

Assuming F(x) is defined and differentiable, then F(x) decreases fastest if we go from x in the direction of the gradient of F

- Vector of partial derivatives of F: $\nabla F(\mathbf{x})$
- Update: $\mathbf{x}' = \mathbf{x} - \gamma \nabla F(\mathbf{x})$
- For sufficiently small values of γ, the value of the function will get smaller
- Large γ can over-shoot
- Repeat until convergence

Gradient Descent



Derivatives

$$\frac{\partial \mathcal{L}(Y, \mathbf{X}, \mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \left(y_i - p(y_i = 1 | \mathbf{x}_i, \mathbf{w})\right) \mathbf{x}_i$$

- The derivative is 0 when $y_i = p(y_i=1|\mathbf{x}_i,\mathbf{w})$
- Minimize the prediction error

Gradient Descent Solution

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} + \gamma \frac{\partial \mathcal{L}(Y, \mathbf{X}, \mathbf{w})}{\partial \mathbf{w}}$$

$$\frac{\partial \mathcal{L}(Y, \mathbf{X}, \mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \left(y_i - p(y_i = 1 | \mathbf{x}_i, \mathbf{w})\right) \mathbf{x}_i$$

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} + \gamma \sum_{i=1}^{N} \left(y_i - p(y_i = 1 | \mathbf{x}_i, \mathbf{w})\right) \mathbf{x}_i$$

Algorithm: Logistic Regression

Train: given data **X** and Y

$$p_{\mathbf{w}}(y = 1|\mathbf{x}) = p(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

- Initialize **w** to starting value
- Repeat until convergence
  - Compute the value of the derivative for **x**, y, and **w**
  - Update **w** by taking a gradient step

Predict: given an example **x**

- Using the learned **w**, compute p(y|**x**,**w**)

Note: many other optimization routines available

Gradient Based Optimization

Multiple methods available for optimizing the same objective function

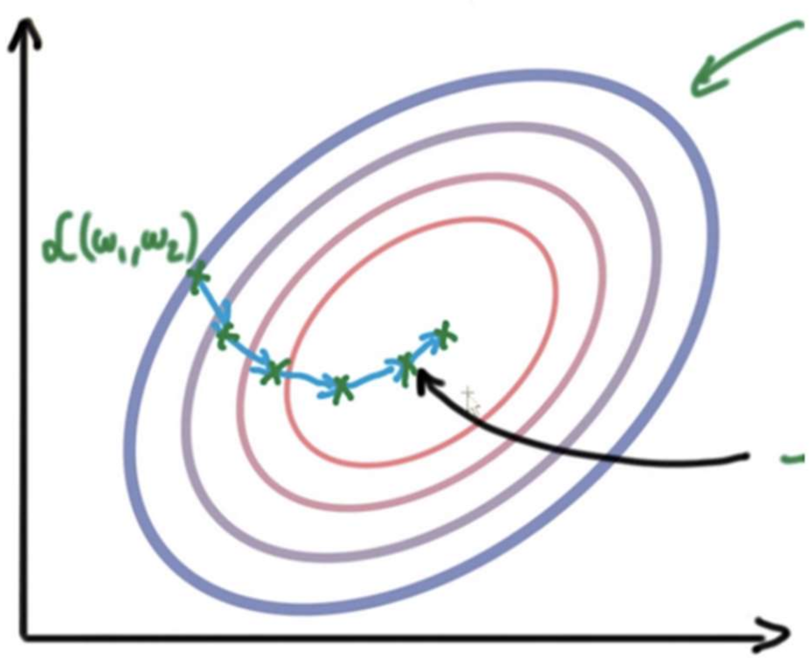
- Zero order methods
- First order methods
- Second order methods
- Adaptive methods
- …

Alternate Methods

Batch gradient descent

- Utilize the gradient of all the data
- Slow: need to consider all the data before making a single update

Gradient Descent



Stochastic Updates

Compute the gradient on a single example at a time

Instead of:

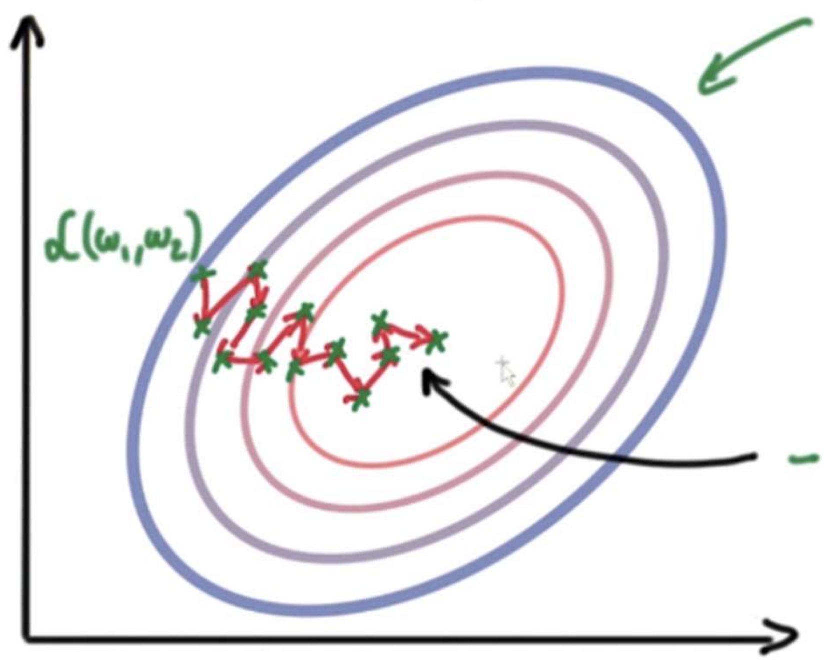
$$\mathbf{w}^{t+1} = \mathbf{w}^{t} + \gamma \sum_{i=1}^{N} (y_i - p(y_i = 1|\mathbf{x}_i, \mathbf{w}))\, \mathbf{x}_i$$

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} + \gamma\, (y_1 - p(y_1 = 1|\mathbf{x}_1, \mathbf{w}))\, \mathbf{x}_1 + (y_2 - p(y_2 = 1|\mathbf{x}_2, \mathbf{w}))\, \mathbf{x}_2 + \ldots$$

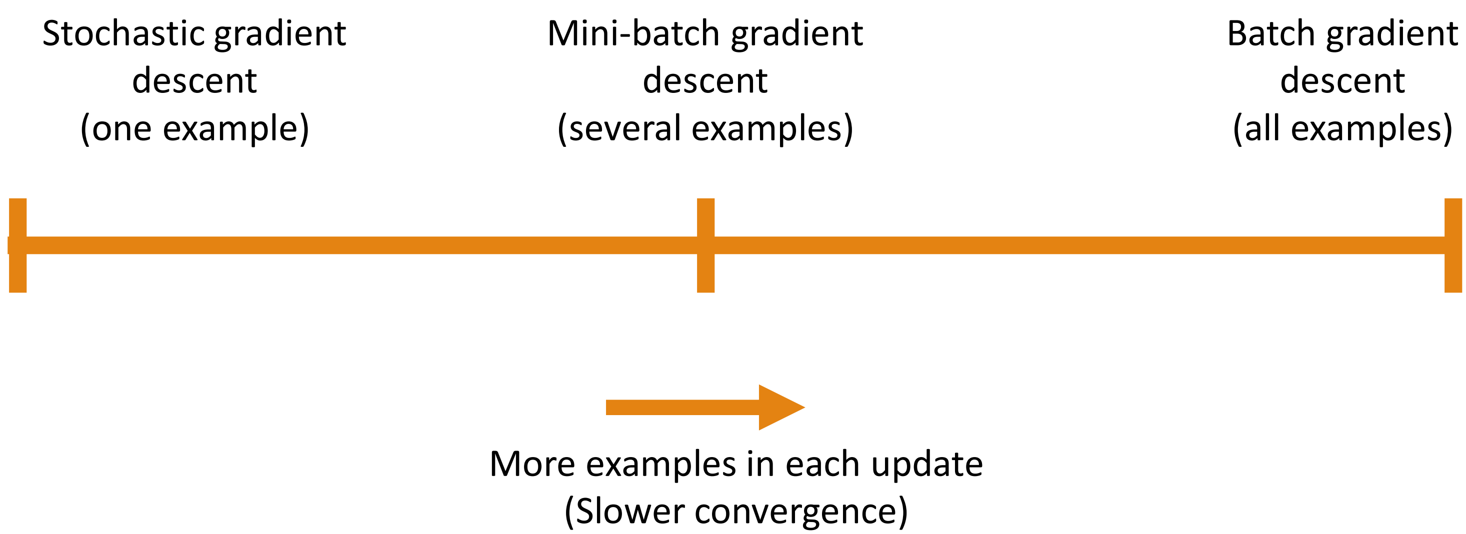
Use one data point:

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} + \gamma\, (y_i - p(y_i = 1|\mathbf{x}_i, \mathbf{w}))\, \mathbf{x}_i$$

Stochastic Gradient Descent



Update Frequency



Regularization

Same over-fitting problems as least squares

Add regularization term to objective to favor different considerations

Similar options

- Quadratic regularization (L_2)
- L_1 regularization (sparse solutions)
- For each regularization we optimize new objective function with new $L_q(\mathbf{w})$ term subtracted from the log-likelihood

Summary

Logistic regression

- Learn $p(y|\mathbf{x},\mathbf{w})$ directly with functional form of distribution
- Maximize the data conditional log-likelihood
- Equivalent to linear prediction
  - Decision rule is a hyper-plane
- Regularization to prevent over-fitting