Log-Linear Models

Kevin Duh
Intro to NLP, Fall 2019

Motivation

- We want to **estimate p(y|x)**
  - e.g. P(Y=word | X = previous_words)
  - e.g. P(Y=label | X = document)
- In the n-gram lectures, we estimated this by counts:
  - P(Y=word | X = previous words) = Count(word, previous_words) / Count(previous_words)
  - Smoothing alleviates unreliable estimates

Outline

1. What is a Log-Linear Model and why use it?
2. How to train it?
3. Interactive visualization

Motivation

- A different approach to estimate p(y|x)
- Log-linear model:
  - enables us to incorporate our knowledge of the problem by defining features of x,y

Log-Linear Model Definition

$$p(y|x) \propto score(x, y)$$

$$score(x, y) = \sum_k \theta_k f_k(x, y)$$

1. Define K feature functions f(), each measuring something about x and y

2. The probability depends on a weighted combination of f()

3. Weights are learned from training data

Example features for word prediction $p(w_n \mid w_{n-1}, w_{n-2})$

$$P(W_n = \text{“}Sam\text{”} \mid W_{n-1} = \text{“}am\text{”}, W_{n-2} = \text{“}I\text{”})$$

$$\begin{aligned}
f_1(x,y) &= Count(\text{“I am Sam”}) \\
f_2(x,y) &= 1/Count(\text{“I am”}) \\
f_3(x,y) &= Count(\text{“am Sam”}) \\
f_4(x,y) &= 1/Count(\text{“am”}) \\
f_5(x,y) &= Count(\text{“Sam”}) \\
f_6(x,y) &= p_{add-one-smooth}(\text{“Sam”} \mid \text{“I am”}) \\
f_7(x,y) &= p_{add-two-smooth}(\text{“Sam”} \mid \text{“I am”}) \\
f_8(x,y) &= p_{add-one-smooth}(\text{“Sam”} \mid \text{“I”}) \\
f_9(x,y) &= I(\text{“Sam” is capitalized}) \\
f_{10}(x,y) &= I(\text{“Sam” is Noun and previous word is Verb})
\end{aligned}$$

Log-Linear Model Definition

We said $p(y|x) \propto score(x, y)$

Main Concept! Get familiar with it

More precisely, Log-Linear models are defined as:

$$p(y \mid x) = \frac{1}{Z(x)} \exp(score(x, y)) = \frac{1}{Z(x)} \exp(\sum_{k=1}^{K} \theta_k f(x, y))$$

Normalization:

$$Z(x) = \sum_{y'} \exp(score(x, y'))$$

It's called Log-Linear because log(p(y|x)) is a linear function.
Sometimes also called Maximum Entropy (MaxEnt) or Multinomial Logistic Regression

Outline

1. What is a Log-Linear Model and why use it?
2. How to train it?
3. Interactive visualization

Training the parameters

We're given training data: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$

Want to find parameters that maximize likelihood on training data:

$$p(\text{training data}; \vec{\theta}) = \prod_{n=1}^{N} p(y_n | x_n; \vec{\theta})$$

$$p(y_n \mid x_n; \vec{\theta}) = \frac{1}{Z(x)} \exp(\sum_{k=1}^{K} \theta_k f(x_n, y_n))$$

$$\vec{\theta} = [\theta_1, \theta_2, \ldots, \theta_K]$$

Equivalent to maximizing the log-likelihood:

$$\sum_{n=1}^{N} \log(p(y_n | x_n; \vec{\theta}))$$

Side-note: Probability vs. Likelihood

- **Parameter known. Data unknown.**
  - I know a coin is biased with this parameter: P(Flip=H)=0.7, P(Flip=T)=0.3
  - Question: If I flip 3 times, what is the probability I'll get HHH? 0.7x0.7x0.7=0.34
- **Data known. Parameter unknown.**
  - I know that I flipped 3 times and got HHH.
  - Question: What's my estimate of the biasness of the coin? i.e. P(Flip=H)=???
    - Maximum likelihood solution is P(Flip=H)=1.0, P(Flip=T)=0.0.
      - Try P(Flip=H)=1. Likelihood = 1.0 x 1.0 x 1.0 = 1.0
      - Try P(Flip=H)=0.7. Likelihood = 0.7x0.7x0.7=0.34

Regularization Term

- Large weights may lead to p(y|x) which may vary largely due to small changes in input
- Encourage weights to be small by adding a penalty
- L2 regularization:

$$||\vec{\theta}||^2 = \left(\sqrt{\theta_1^2 + \theta_2^2 + \ldots + \theta_K^2}\right)^2 = \sum_{k=1}^{K} \theta_k^2$$

Loss Function (or Objective Function)

- Goal: Find parameters that minimize this Loss Function, Negative Log-Likelihood + Regularizer

$$L(\vec{\theta}) = -\sum_{n=1}^{N} \log(p(y_n|x_n;\vec{\theta})) + ||\vec{\theta}||^2$$

- We can use various optimization techniques. Think back to your Calculus class.

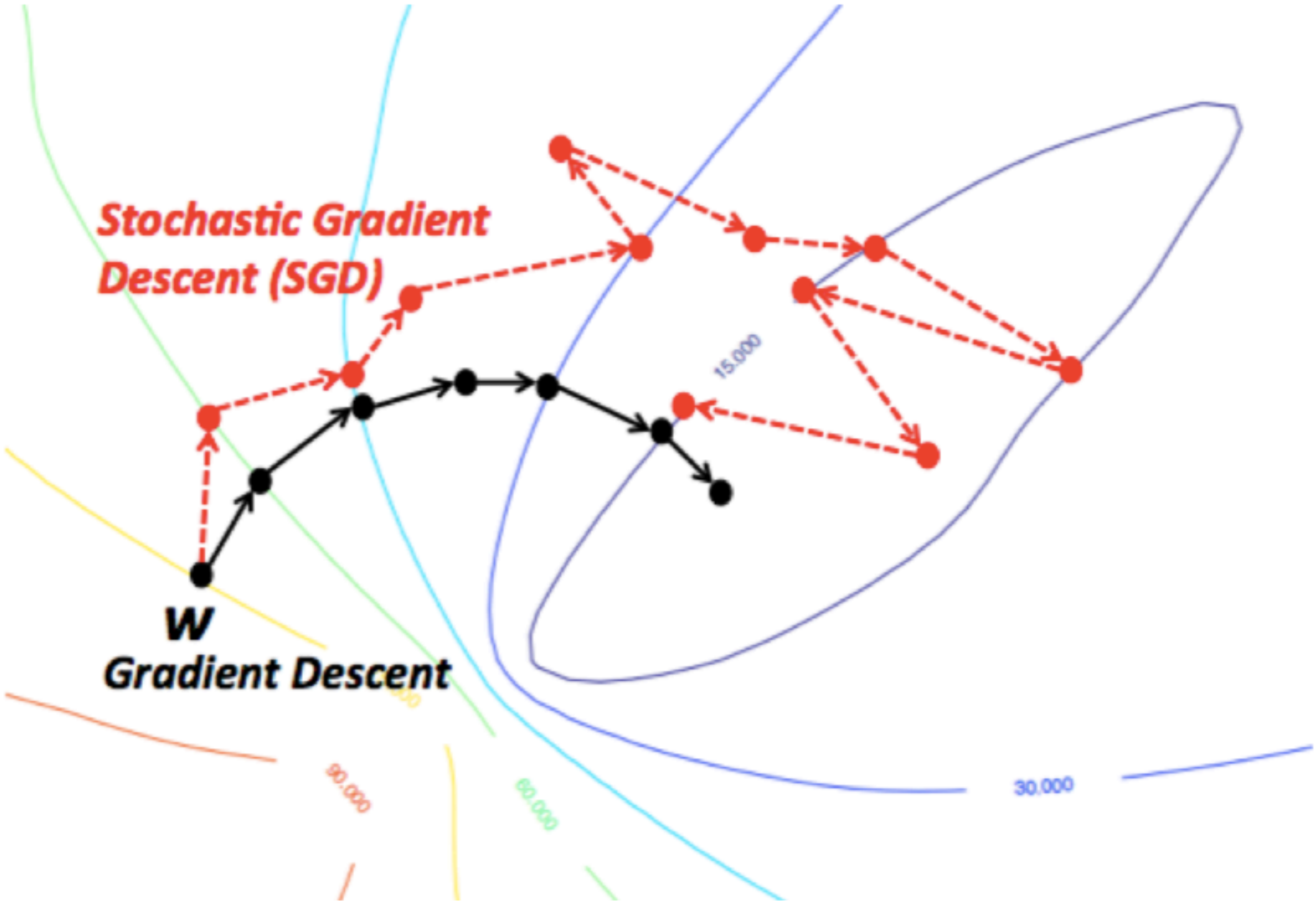
Training by Gradient Descent

- The gradient of a function points to the direction of steepest increase in that function

$$\nabla L(\vec{\theta}) = \left[\frac{\partial L(\vec{\theta})}{\partial \theta_1}; \frac{\partial L(\vec{\theta})}{\partial \theta_2}; \cdots; \frac{\partial L(\vec{\theta})}{\partial \theta_K} \right]$$

- Gradient Descent Algorithm: start with some random parameter, keep going in the opposite gradient direction

- Like how you ski down a mountain


Stochastic Gradient
Descent (SGD)
W
Gradient Descent
15,000
30,000
90,000
60,000

Training by Gradient Descent

- Exercise: Compute partial derivatives of loss function

$$L(\vec{\theta}) = -\sum_{n=1}^{N} \log(p(y_n|x_n;\vec{\theta})) + ||\vec{\theta}||^2$$

- Regularizer part:

$$\frac{\partial||\vec{\theta}||^2}{\partial\theta_1} = \frac{\partial\left(\sqrt{\theta_1^2+\theta_2^2+\ldots+\theta_K^2}^2\right)}{\partial\theta_1} = \frac{\partial\left(\theta_1^2+\theta_2^2+\ldots+\theta_K^2\right)}{\partial\theta_1} = \frac{\partial\left(\theta_1^2\right)}{\partial\theta_1} = 2\theta_1$$

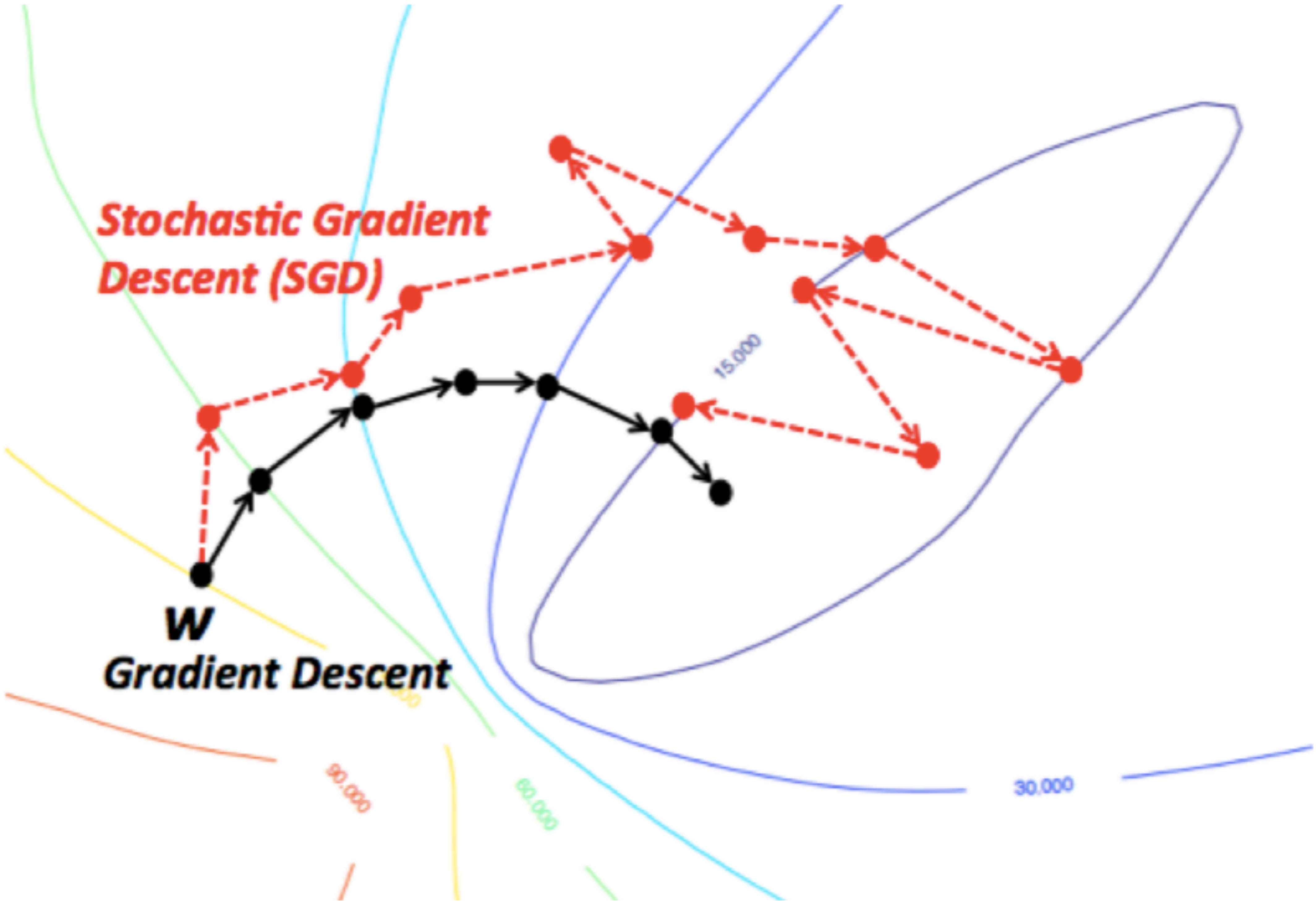
- Likelihood part:

$$\frac{\partial\left(-\sum_{n=1}^{N}\log p(y_n \mid x_n;\vec{\theta})\right)}{\partial\theta_1} = \sum_{n=1}^{N}\frac{\partial\left(-\log p(y_n \mid x_n;\vec{\theta})\right)}{\partial\theta_1}$$

Gradient Descent and Stochastic Gradient Descent (SGD)

- Gradient Descent Algorithm:
  - Start with some parameter $\vec{\theta}$
  - While not converged:
    - Update parameter $\vec{\theta} := \vec{\theta} - \eta \nabla L(\vec{\theta})$
- SGD:

$$\sum_{n=1}^{N} \frac{\partial \left(- \log p(y_n \mid x_n; \vec{\theta}) \right)}{\partial \theta_1} \approx \sum_{n:subset} \frac{\partial \left(- \log p(y_n \mid x_n; \vec{\theta}) \right)}{\partial \theta_1}$$


Stochastic Gradient Descent (SGD)
W
Gradient Descent
15,000
30,000
90,000
60,000

Summary

- Log-Linear Model has this form:

$$\boxed{p(y_n \mid x_n; \vec{\theta}) = \frac{1}{Z(x)} \exp\left(\sum_{k=1}^{K} \theta_k f(x_n, y_n)\right)}$$

- Features enable us to incorporate knowledge of the problem

- Given training data, we fit parameters to maximize likelihood (optionally with a regularization term)

$$p(\text{training data}; \vec{\theta}) = \prod_{n=1}^{N} p(y_n | x_n; \vec{\theta})$$

Interactive Visualization

- http://www.cs.jhu.edu/~jason/465/hw-prob/loglin/#1