ECE M146 Introduction to Machine Learning

Lecture 3 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Last time

- Perceptron algorithm for binary classification

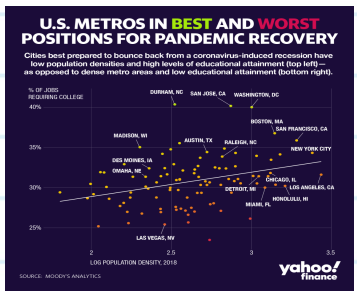
labeled classes with +1, -1

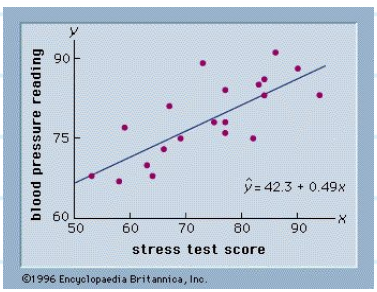
- on-line alg. that updates weight vector $\vec{w}$ one misclassified point at a time
- proof of convergence
- works for linearly separable data

Today's Lecture: Linear regression

- Broad set of applications
  - Evaluating trends and sales estimates
  - Assessing insurance cost
  - Analyzing price elasticity
  - Sports betting
  - …

In recent news:





https://www.britannica.com/science/estimated-regression-equation

https://finance.yahoo.com/news/cities-bounce-back-coronavirus-pandemic-moodys-144452350.html?guccounter=1

Set-up

- There are N data points (x_k, y_k) for $1 \leq k \leq N$.
- Here x_k is a vector, and y_k is a real number.

- Example: y_k is the price of the house and components in x_k are number of bedrooms, square footage, number of levels etc.

- There is a function that maps x to y.
- This function is unknown.

Set-up

- We are going to find the best linear fit for this function.
- How ?

- Minimize the quadratic loss
- Formula:

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T \cdot \vec{x}_i - y_i)^2$$

so that the loss ⊥ N

- assume bias term is already absolved, so that $x_0 = 1$

Set-up

- For convenience, we will assume that we have already performed the transformation from the current dimension to one higher so we don't have to deal with the intercept term separately.
- Formula:

(Reminder)

Suppose we start with $\tilde{x}$ (dim D)

$\rightarrow$ $\vec{x} = \begin{bmatrix} 1 \\ \tilde{x} \end{bmatrix}$ (dim D+1)

$\tilde{w}$ is of dimension D

scalar term w_0 expanded

$\vec{w}$ is of dimension D+1

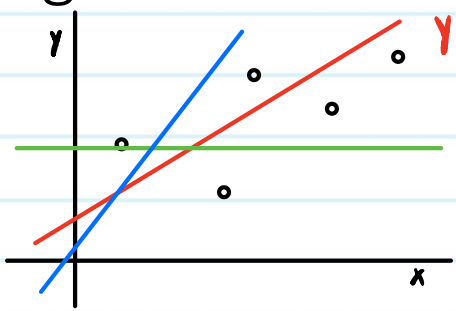
$\vec{w}^T\vec{x}$ ($\vec{x}^T\vec{w}$) : scalar

Model selection

- Note that the data may not match this model perfectly.
- It could be that the labeled data truly has a linear relationship, and that noise makes it appear non-linear.
- It could also be that the true relationship is non-linear.

Linear regression in one dimension

- Picture:



x, y scalars

- Formula for the linear relationship:

looking for w_0, w_1

- Loss function:

$$L = \frac{1}{N}\sum_{i=1}^{N}(w_0 \cdot 1 + w_1 \cdot x_i - y_i)^2$$

How to minimize the loss ?

- Take a derivative and set it to zero.
- We have two unknowns. w_0, w_1
- Derivative with respect to w_0:

$$\frac{\partial L}{\partial w_0} = \frac{\partial}{\partial w_0}\left(\frac{1}{N}\sum_{i=1}^{N}(w_0 \cdot 1 + w_1 \cdot x_i - y_i)^2\right) = 0$$

$$\Downarrow$$

$$0$$

$$N \cdot w_0 + w_1 \sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i = 0$$

$$N \cdot w_0 + w_1 \cdot x_s - y_s = 0$$

$$w_0 = \frac{y_s - w_1 \cdot x_s}{N} = \bar{y} - w_1 \bar{x} \qquad \left(\bar{y} = \frac{y_s}{N},\ \bar{x} = \frac{x_s}{N}\right)$$

Loss minimization.

- Now set the derivative with respect to w_1 to zero.

$$\frac{\partial L}{\partial w_1} = \frac{\partial}{\partial w_1}\left(\frac{1}{N}\sum_{i=1}^{N}(w_0 \cdot 1 + w_1 \cdot x_i - y_i)^2\right) = 0$$

$$= \frac{1}{N}\sum_{i=1}^{N} 2(w_0 + w_1 x_i + y_i) \cdot x_i = 0$$

$$\sum_{i=1}^{N}(w_0 + w_1 x_i + y_i) \cdot x_i = 0$$

$$w_0 \sum_{i=1}^{N} x_i + w_1 \sum_{i=1}^{N} x_i^2 + \sum_{i=1}^{N} y_i x_i = 0$$

$$\bar{y} - w_1 \cdot \bar{x} \qquad x_s = N\bar{x}$$

Derivative w.r.t. w_1 – continued.

$$(\bar{y} - w_1 \cdot \bar{x}) \cdot N \cdot \bar{x} + w_1 \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} y_i x_i$$

$$w_1 \left(\sum_{i=1}^{N} x_i^2 - N \cdot \bar{x}^2 \right) = \sum_{i=1}^{N} y_i x_i - N \bar{x} \bar{y}$$

solve for w_0 w/ earlier expression

$$w_1 = \frac{\sum_{i=1}^{N} y_i x_i - N \bar{x} \bar{y}}{\sum_{i=1}^{N} x_i^2 - N \cdot \bar{x}^2}$$

$$w_1 = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N} (x_i - \bar{x})^2}$$

lastly, verify these 2 expressions are equivalent (algebra) ⇒ conclude w_1 agrees w/ above

So far

- We have derived the best linear fit for x a scalar.

  explicitly computed w_0, w_1 $y = w_0 + w_1 x$

  at test time: given x_{test}, $y_{test} = w_0 + w_1 \cdot x_{test}$
- In general, x may not have just one attribute.

- Next, we will generalize the result for x a vector.

Linear regression – general case

- Loss function is now written as follows:

$$L = \frac{1}{N} \|\vec{x} \cdot \vec{w} - \vec{y}\|^2$$

N data pts

- Here, y is a vector, w is a vector, and X is a matrix.
- Check dimensions:

$$\vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \vec{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix} \quad X = \begin{bmatrix} — \; x_1^T \; — \\ \vdots \\ — \; x_N^T \; — \end{bmatrix}$$

$N \qquad D+1 \qquad N \times (D+1)$

Let's take a closer look at the loss function

- Note that loss is still a **scalar**, even though the terms specifying it are vectors and matrices.

$$L = \frac{1}{N} \|\vec{x} \cdot \vec{w} - \vec{y}\|^2$$

- Expand the loss term:

$$L = \frac{1}{N} (\vec{y} - \vec{x} \cdot \vec{w})^T (\vec{y} - \vec{x} \cdot \vec{w})$$

$$= \frac{1}{N} (\vec{y}^T \vec{y} - 2\vec{w}^T \vec{x}^T \vec{y} + \vec{w}^T \vec{x}^T \vec{x} \cdot \vec{w})$$

Goal: minimize L w.r.t. vector w

- How ?
- As in the scalar case, take the derivatives. We do this term by term:

$$\frac{\partial L}{\partial w_0} = 0, \quad \frac{\partial L}{\partial w_1} = 0, \quad \dots \quad \frac{\partial L}{\partial w_D} = 0$$

- We group these partial derivatives into a vector.

$\nabla_{\vec{w}} L(\vec{w})$ vector of all ∂'s $(D+1 \times 1)$

- This vector is called a **gradient**.

Minimization = set gradient to zero

$$D_w L(\vec{w}) = D_w \frac{1}{n}(\vec{w}^T \vec{x}^T \vec{x} \vec{w} - 2\vec{w}^T \vec{x}^T \vec{y} + \vec{y}^T \vec{y}) = 0$$

$$D_w (\vec{y}^T \vec{y}) = 0$$

need: $D_w (\vec{w}^T \vec{x}^T \vec{x} \vec{w} - 2\vec{w}^T \vec{x}^T \vec{y} + \vec{y}^T \vec{y})$

Minimization = set gradient to zero

- Since we are dealing with vectors and matrices, we need rules for matrix calculus.

Matrix calculus rules

- Rule 1: $D_w(\vec{w}^T\vec{b}) = \vec{b}$

- Example: consider

$$[w_1 \;\; w_2 \;\; w_3]\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = w_1b_1 + w_2b_2 + w_3b_3$$

$$\frac{\partial(\vec{w}^T\vec{b})}{\partial w_1} = \frac{\partial}{\partial w_1}(w_1b_1 + w_2b_2 + w_3b_3)$$

$$= b_1$$

$$\frac{\partial(\vec{w}^T\vec{b})}{\partial w_2} = b_2 \qquad \frac{\partial(\vec{w}^T\vec{b})}{\partial w} = b_3$$

Matrix calculus rules

- Rule 2: $D_w(\vec{w}^T \cdot A \cdot \vec{w}) = (A + A^T) \cdot \vec{w}$

$1 \times M \quad M \times M \quad M \times 1$

Back to our set up.

- Recall that we want to set the gradient of the loss with respect to the vector w to zero.

$$D_w L(\vec{w}) = 0$$

$$D_w(\vec{w}^T \vec{x}^T \vec{x} \vec{w} - 2\vec{w}^T \vec{x}^T \vec{y} + \vec{y}^T \vec{y}) = 0$$

$$(\vec{x}^T \vec{x} + (\vec{x}^T \vec{x})^T) - 2\vec{x}^T \vec{y} = 0$$

$$2\vec{x}^T \vec{x} \cdot \vec{w} - 2\vec{x}^T \vec{y} = 0$$

Solving a system of linear equations

$$(\vec{X}^T\vec{X})\,\vec{w} = \vec{X}^T\vec{y}$$

$$\vec{w} = (\vec{X}^T\vec{X})^{-1}\vec{X}^T\vec{y}$$

- When is this matrix invertible ?
- When X^TX is full rank.

What if the matrix is not full rank ?

- Then, find pseudo inverse.

  A (not necessarily invertible), find $\tilde{A}^{-1} \cdot A = I$

- One way is via SVD decomposition:

$$A = \vec{u} \cdot \Sigma \cdot \vec{v}^T$$
$$\tilde{A}^{-1} = \vec{v} \cdot \Sigma^{-1} \cdot \vec{u}^T$$
$$\tilde{A}^{-1} A = I$$

More on the vector w

- Formula: $\vec{w} = (\vec{X}^T \vec{X})^{-1} \vec{X}^T \vec{y}$

- This vector w of regression coefficients completely specifies this linear regression – it is a **closed form solution** and it is **optimal** under quadratic loss.

Once we have $\vec{w}$, don't need (x, y) training set.

What is the training error ?

- Here is what the model predicts on the training data:

$$\hat{y} = \vec{x} \cdot \vec{w}$$
$$= \vec{x}(\vec{x}^T \vec{x})^{-1} \cdot \vec{x}^T \vec{y}$$
$$= H \cdot \vec{y}$$

$\hat{y}$: what the model predicts

- Matrix H of course need to be identity. Recall, we are not trying to be perfect on the training data.

"Hat matrix"

Now we have the system. How do we use it ?

- At test time:

we computed $\vec{w}$ based on

(x_i, y_i) $\quad 1 \leq i \leq N$

$$y_{TEST} = \vec{X}^T_{TEST} \cdot \vec{w}$$

Properties of the H matrix

$$H = \vec{x}(\vec{x}^T\vec{x})^{-1}\cdot\vec{x}^T$$

$$H^2 = H$$

$$H^k = H$$

- H is an idempotent matrix.
- Why ?

Linear least squares with regularization

- Add regularization term:

$$\lambda \cdot \|\vec{w}\|^2 \qquad \lambda \geq 0$$

- Gradient: old loss

Linear least squares with regularization

- Derivative of the additional term:

$$\nabla_w \|\vec{w}\|^2 = ?$$

$$\frac{\partial \|\vec{w}\|^2}{\partial w_1} = 2w_1, \ \ldots \ \frac{\partial \|\vec{w}\|^2}{\partial w_m} = 2w_m$$

- Putting it all together:

$$(X^T X + \lambda I)\vec{w} = X^T y$$

$$\vec{w} = (X^T X + \lambda I)^{-1} \cdot X^T y$$

So far:

- We learned about the perceptron algorithm, which is used for classification. Perceptron is an on-line algorithm.

- We learned about the linear least squares method, which is used for linear regression.

- This method can be used for classification as well, with y being a class membership indicator matrix, but there are other more robust methods for classification.

Appendix: consistency between scalar and vector cases

vector case: $\vec{w} = (\vec{X}^T \vec{X})^{-1} \cdot \vec{X}^T \vec{y}$

suppose x is scalar

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_N \end{bmatrix}$$

ECE M146 Introduction to Machine Learning

Lecture 4 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Today's Lecture

Recap:

- Linear regression via linear system of equations under the squared loss

New topics:

- Gradient descent
- Stochastic gradient descent

Review: Linear regression

- Picture:

$(x_1, y_1) \ldots (x_N, y_N)$

- Based on the labeled training set of size N, produce the best linear fit.
- At testing time, plug in the test data into this linear fit.

$\vec{w}^T \cdot \vec{x}_{TEST} = \vec{y}_{TEST}$

↓

produce $\vec{w}$ of coefficients

What is the “best” linear fit ?

- It is the one that minimizes the squared error – error is the difference between what the model predicts and what the true labels are.

- Formula for the loss: $L = \frac{1}{N}\|\vec{y} - \vec{x}\cdot\vec{w}\|^2$

- Minimization is done by taking the derivative and setting it to zero.

$\vec{y}$: labels $(N \times 1)$

X : matrix $(N \times D)$ $\begin{bmatrix} — \vec{x}_1 — \\ — \vec{x}_2 — \\ \vdots \\ — \vec{x}_n — \end{bmatrix}$ $\vec{x}_i$ is $1 \times D \rightarrow \vec{w}$ is $1 \times D$

Mathematical analysis -- continued

scalar

$$\frac{\partial L}{\partial \vec{w}} = 0$$

$$\frac{\partial}{\partial \vec{w}} \left[(x\vec{w} - \vec{y})^T (x\vec{w} - \vec{y}) \right] = 0$$

$$\frac{\partial}{\partial \vec{w}} \left[\vec{w}^T x^T x \vec{w} - \vec{y}^T x \vec{w} - (x\vec{w})^T \vec{y} + \vec{y}^T \vec{y} \right] = 0$$

same

scalar

$$\frac{\partial}{\partial \vec{w}} \vec{y}^T \vec{y} = 0$$

Closed form solution for the regression coefficients

$$\frac{\partial}{\partial \vec{w}}(\vec{w}^T A \vec{w}) - \frac{\partial}{\partial \vec{w}}(\vec{w}^T \vec{b}) = 0$$

$(A + A^T)\vec{w}$ $\quad \vec{b}$

$$A = X^T X \qquad \vec{b} = X^T \vec{y}$$

(symmetric

$$\vec{w} = A^{-1}\vec{b}$$

$$\vec{w} = (X^T X)^{-1} X^T \vec{y}$$

Relationship between the predicted label and true label on the training data

$$\vec{w} = (X^T X)^{-1} X^T \vec{y}$$

$$\hat{y} = X\vec{w} \quad \text{(predicted by model)}$$

$$\hat{y} = \underbrace{X(X^T X)^{-1} X^T}_{H \text{ (proj. matrix)}} \vec{y}$$

At test time

- Use regression coefficients as follows:

$$\vec{w}^T \cdot \vec{x}_{TEST} = y_{TEST}$$

- We have produced a linear model. Output is linear in w.

also linear in $\vec{x}$

Extension: regularization

- Add a penalty term with L2 norm

$$L = \frac{1}{2}\|\vec{x}\cdot\vec{w} - \vec{y}\|^2 + \frac{1}{2}\lambda\cdot\|\vec{w}\|^2$$

(cancels 2 in the exponent when taking derivatives

$$\vec{w} = (\vec{X}^T\vec{X} + \lambda\cdot I)^{-1}\cdot\vec{X}^T\vec{y}$$

Today's Lecture

Recap:

- Linear regression via linear system of equations under the square loss

New topics:

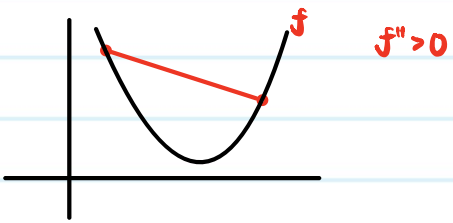
- Gradient descent
- Stochastic gradient descent

Another approach to minimization

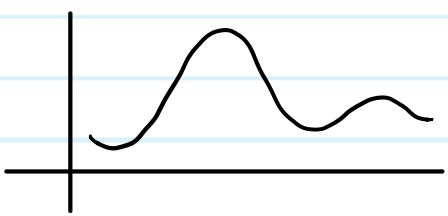
- An alternative way of minimizing a **convex function** is by **gradient descent**.

- Convex function:

  has a unique min



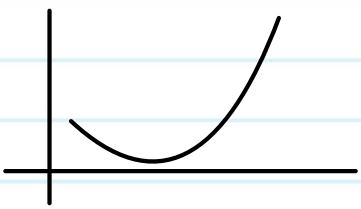
- Not a convex function:



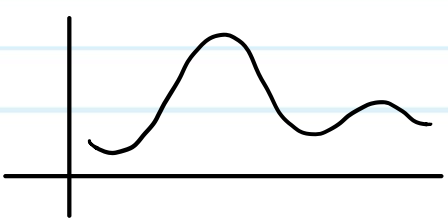
local minima

Gradient descent -- picture

- Move down the direction of the steepest descent.



- Finds **global minimum**. (for convex)
- Can also be used for minimizing a non-convex function, in which case it finds a **local minimum**.



Gradient descent – set up

- Suppose our current weight vector is w.

- The new weight vector is w+η*v, where η is **the step size** and v is a unit vector that points in some direction.

$$\|\vec{v}\|^2 = 1$$

- Choice of η :

$$\eta \neq 0, \quad 0 < \eta < 1$$

- What direction for v ?

Let's build intuition with the scalar case

(special case w/ $\vec{w} \cdot x$) for simplicity, have 1 $\vec{w}$

- Consider our earlier set up with x's being scalar.
- Old loss:

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}(w \cdot x_i - y_i)^2$$

- Here, w is also scalar.
- New loss:

$$L(\vec{w} + \eta\vec{v}) = \frac{1}{N}\sum_{i=1}^{N}((\vec{w} + \eta\vec{v})x_i - y_i)^2$$

Change in loss

- We want loss to decrease

$$\frac{1}{N}\sum_{i=1}^{N}((w+\eta v)x_i - y_i)^2 - \frac{1}{N}\sum_{i=1}^{N}(wx_i - y_i)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(2w\eta v x_i + \eta^2 \vec{v}^2 \vec{x}_i^2 - 2\eta v x_i y_i)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\eta^2 x_i^2 + \sum_{i=1}^{N} 2\eta v (wx_i - y_i)\cdot x_i$$

Change in loss – vector case

- Repeat scalar case:

$$\frac{1}{N}\sum_{i=1}^{N}\eta^2 x_i^2 + \eta\cdot\frac{1}{N}\sum_{i=1}^{N}2\cdot(wx_i - y_i)\cdot x_i\cdot v$$

- Vector case (exercise):

$$\frac{1}{N}\eta^2\sum_{i=1}^{N}\|x_i\|^2 + \eta\cdot D_w L(w)^T\cdot v$$

- Note the following:

$$\frac{\partial L}{\partial w} = D_w L(w) = D_w\left(\frac{1}{N}\sum_{i=1}^{N}(wx_i - y_i)^2\right)$$
$$= \frac{1}{N}\sum_{i=1}^{N}2\cdot(wx_i - y_i)\cdot x_i$$

Change in loss – vector case

- First term is non-negative so can't help towards making the change in loss more negative, and it does not depend on w.

can't do anything about $\frac{1}{N}\eta^2 \sum_{i=1}^{N} \|x_i\|^2$ term

- Let's then focus on the second term and see how negative can we make it.

$$\eta(\nabla_{\vec{w}} L(\vec{w}))^T \cdot \vec{v}$$

inner product

Detour: property of the inner product

- Consider the inner product of two vectors, a and v, with v being a unit vector.

$\vec{a}^T \cdot \vec{v}$ where $\|\vec{v}\| = 1$

$\vec{a}^T \cdot \vec{v} = \|\vec{a}\| \|\vec{v}\| \cos\theta$

- When is this inner product minimized ?

$-\|\vec{a}\| \|\vec{v}\| \leq \vec{a}^T \cdot \vec{v} \leq \|\vec{a}\| \|\vec{v}\|$

- Result: $\vec{a}^T \cdot \vec{v} = -\|\vec{a}\| \|\vec{v}\|$ ($\vec{v}$ is a unit vector)

$\vec{v} = \frac{-\vec{a}}{\|\vec{a}\|}$

We have arrived at the gradient descent rule

- Suppose we have vector w_t at step t.
- At step t+1, the vector w_{t+1} is:

$$\vec{w}_{t+1} = \vec{w}_t - \eta \cdot \frac{\nabla_{\vec{w}} L(\vec{w})}{\|\nabla_{\vec{w}} L(\vec{w})\|} \qquad v = -\frac{\nabla_{\vec{w}} L(\vec{w})}{\|\nabla_{\vec{w}} L(\vec{w})\|}$$

- Picture:

Effect of the step size

- Large η – makes large steps but may overshoot.
- Small η – makes small steps but could be too slow.
- Can adjust the value adaptively, so the step size starts off bigger and decreases with the iteration number.
- Can also normalize to avoid computing the norm:

instead of fixed η, use $\eta_t = \frac{\eta}{\|\nabla_{w_t} L(w_t)\|} \cdot \nabla_{w_t} L(w_t)$

Today's Lecture

Recap:

- Linear regression via linear system of equations under the square loss

New topics:

- Gradient descent
- Stochastic gradient descent

Let's recall perceptron

- Recall: $w^{(k)} + y_j \cdot x_j = w$

  x_j is currently misclassified point

- Perceptron criterion:

  $\vec{w}^T \cdot x_i \cdot y_i > 0 \quad \forall\, i,\ 1 \le i \le N$

  $y_i \in \{\pm 1\}$

- For a misclassified point: (x_j, y_j)

  $\vec{w}^T \cdot x_j \cdot y_j < 0$

Reformulation of the loss

- We want to minimize the following loss function:

$$L(w) = -\sum_{i \in M} w^T \cdot x_i \cdot y_i$$

index

M is set of misclassified points

- Each misclassified point contributes a positive amount.
- If all points are incorrectly classified, this function is (very) positive.
- If all points are correctly classified, this function is zero.

$L(w) \geq 0$ $L(w) > 0$ if ≥ 1 misclassified point

Gradient of the loss function

- At all points: $\frac{\partial L}{\partial w} = -\sum_{i \in M} \vec{x}_i \vec{b}_i$
- At one point: (x_i, y_i)
- For one point, the update is (recall):

$$w^{(k+1)} = w^{(k)} - (-x_j \cdot y_j) = w^{(k)} + x_j y_j$$

- So, perceptron performs stochastic gradient descent on an appropriately defined loss function!

Stochastic gradient descent

- Instead of computing the vector of N derivatives as in the gradient descent, compute it only at one data point.

$$w_t - \eta \frac{2(w_t^T x_i - y_i)x_i}{\|2(w_t^T x_i - y_i)x_i\|}$$

given a particular (x_i, y_i)

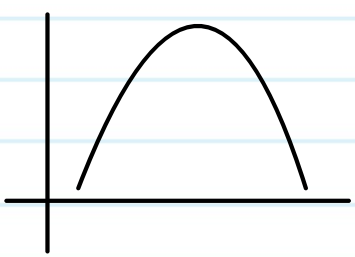
Further discussion

- When the function to be minimized is convex, gradient descent gives the same answer as the linear least square method (provided we use a good choice of η).

Maximization instead of minimization

- Conceptually same procedure, use **gradient ascent**.
- Instead of $-\eta$ we use $+\eta$

concave function



Further discussion

- Gradient descent makes progress on every step, i.e., each step moves in the direction of the steepest descent.
- In contrast, stochastic gradient descent does not since by design it depends on the evaluation at just one data point.
- But, stochastic gradient descent makes progress in expectation (hence the name).

Further discussion

- Stochastic gradient is much faster. It is a pervasive computational unit in neural networks in both forward and backpropagation.
- Gradient descent and stochastic gradient descent are routinely used whenever we need to minimize a function, need not be a convex function.
- Yes, it will likely find local minima and that is a fact of life (neural nets).

Further discussion

- Gradient descent is not guaranteed to reduce the loss at each step (we had an unavoidable positive term).
- Other methods exist that provably reduce loss at every step. For example, Newton method locally approximates a convex function with a quadratic.
- Faster than gradient descent but computationally more intensive.

Newton Method

- Recall Taylor Series Expansion

$$f(x) = f(a) + \sum_{n=1}^{\infty} \frac{1}{n!} f^{(n)}(a) \cdot (x-a)^n$$

$$L = f(x) = f(a) + f'(a)(x-a)$$

Newton Method

- Hessian

have $L(a)$, want $L(w)$

$$L(w) \approx L(a) + L'(a)(w-a)$$

set $L'(w) = 0$: $L'(w) \approx L'(a) + L''(a)(w-a) = 0$

$$a - w = \frac{L'(a)}{L''(a)} \rightarrow w = a - \frac{L'(a)}{L''(a)}$$

$$w_{t+1} = w_t - \frac{L'(w_t)}{L''(w_t)}$$

$$\vec{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

vector case: $L'(\vec{w}_t) \rightarrow D_{\vec{w}} L(\vec{w}) = \begin{pmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \end{pmatrix}$

$$L''(\vec{w}_t) \rightarrow \begin{pmatrix} \frac{\partial^2 L}{\partial w_1^2} & \frac{\partial^2 L}{\partial w_1 \partial w_2} \\ \frac{\partial^2 L}{\partial w_2 \partial w_1} & \frac{\partial^2 L}{\partial w_2^2} \end{pmatrix}$$

$$\vec{w}_{t+1} = \vec{w}_t - [H(\vec{w}_t)]^{-1} \cdot D_{w_t} L(\vec{w}_t)$$

Newton Method

- Newton Method converges in one step if the underlying function is quadratic.
- Compare with Gradient Descent
- Newton Method is not suitable for certain functions (non-smooth or when the 2^{nd} derivative is zero).

Recap

- Perceptron is on-line algorithm for binary classification; at test time, outputs one of two choices
- Linear least squares for linear regression; at test time, outputs a real-valued number
- (Stochastic) gradient descent can be used for both.

Logistic regression

- Can be viewed as an in-between method in the sense that it outputs a real value like in regression, but it is used for classification.
- Note the misnomer!

- Like the methods studied so far, it represents a linear model.

Logistic regression

- It is an instance of a probabilistic discriminative model, where we model p(y|x). As such, it is modeling-wise more complex than those e.g., perceptron that are described by a discriminant function.
- Later on, we will see generative models such as naïve Bayes, that model p(x|y).