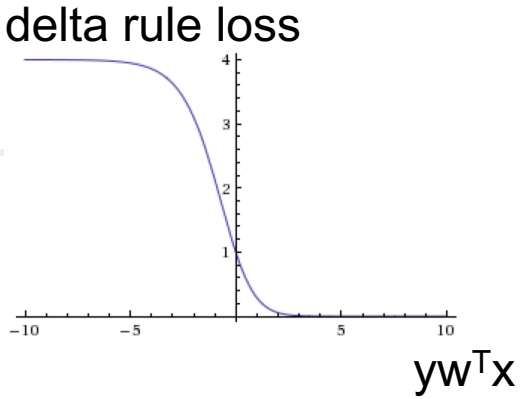
CMSC 510 – L07
Regularization Methods for Machine Learning

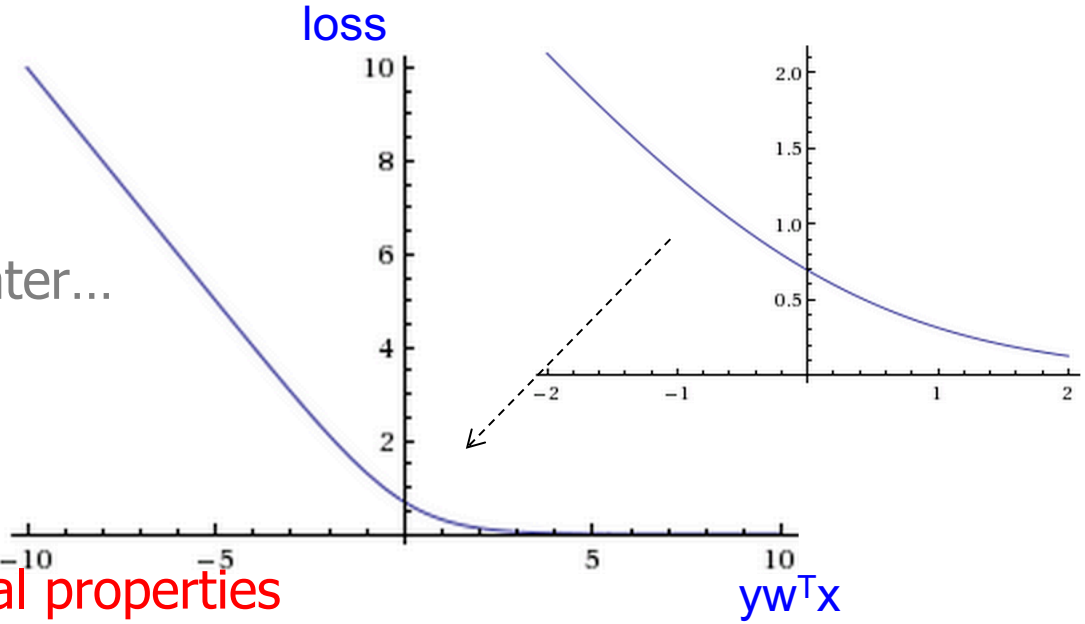
Instructor:
Dr. Tom Arodz

Recap: Logistic loss

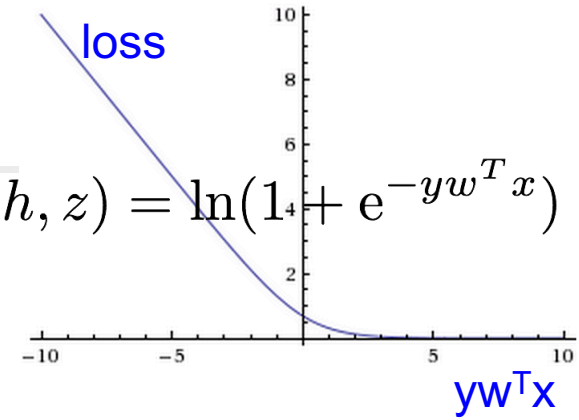


$$\ell(h, z) = \ln(1 + \mathrm{e}^{-yw^T x})$$

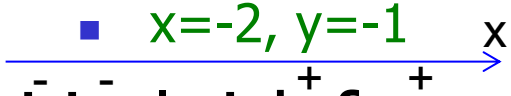
- Derived from: $a(u) = \frac{1}{1+\mathrm{e}^{-u}}$
  - Cross-entropy loss over a(wᵀx)
  - Maximum likelihood estimate for P(y|x,w) =a(wᵀx)
    - We will see that later…
  - Good mathematical properties



Logistic regression

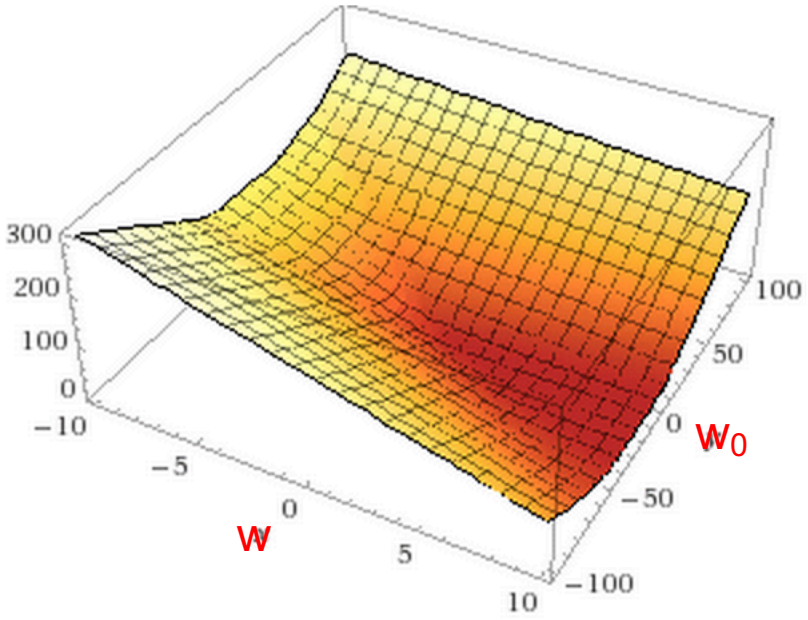


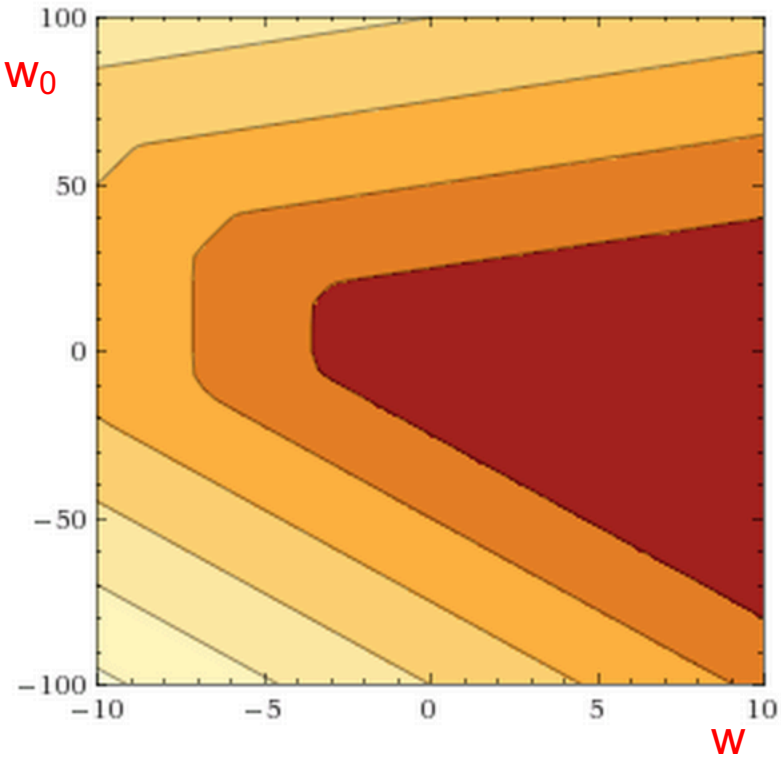
- Logistic loss - Four samples:
  - x=7, y=1
  - x=4, y=1
  - x=-1, y=-1
  - x=-2, y=-1



$$\min \frac{1}{m} \sum_{i=1}^{m} \ln(1+\mathrm{e}^{-y_i w^T x_i})$$

- Empirical risk for any w, w_0

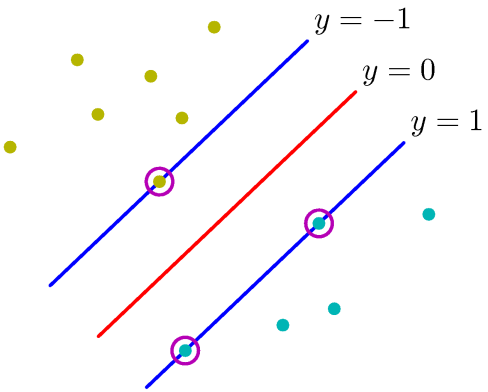


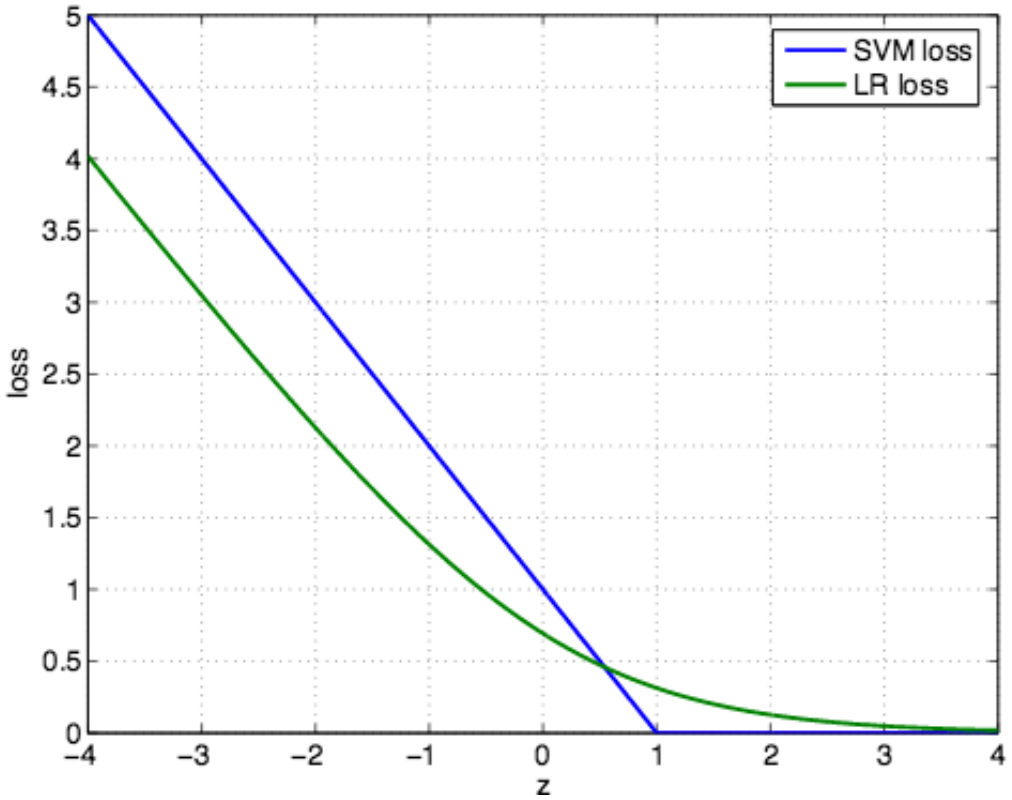


- No local minima!

Hinge loss

- Yet another loss: hinge loss
  - Loss = $[1\text{-}yh(x)]_+$ = ReLU(1-yh(x)) = max(0,1-yh(x))
- Popularized by Support Vector Machines

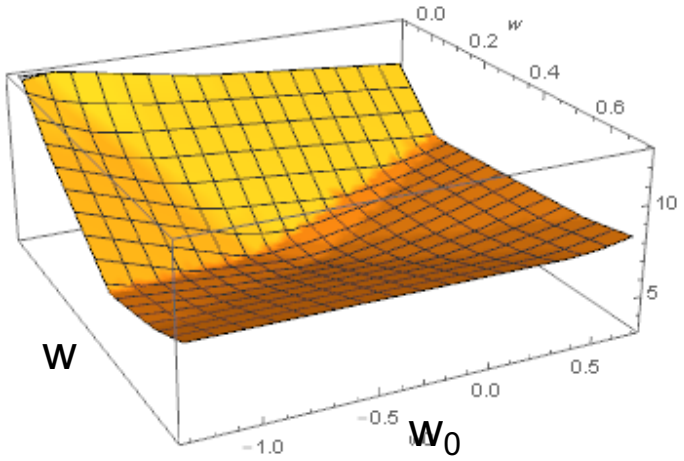




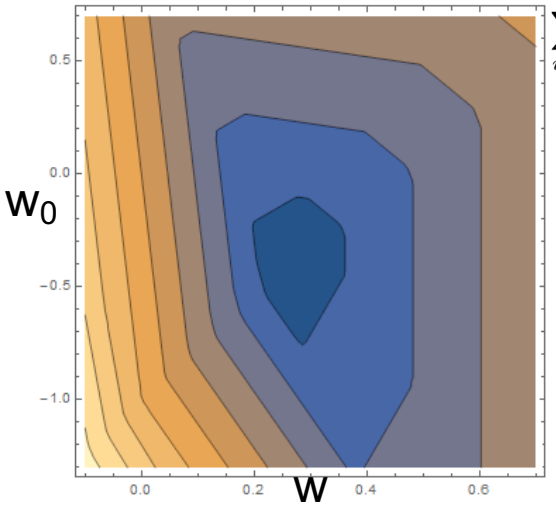
- Can be solved efficiently without gradients, using QP

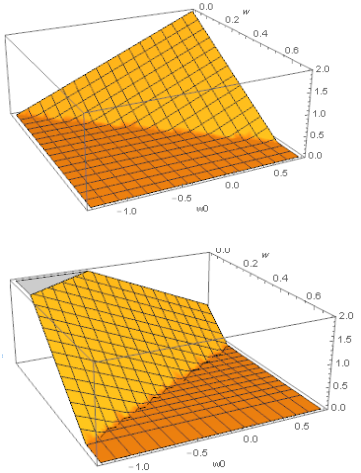
Support Vector Machine

- Hinge risk:

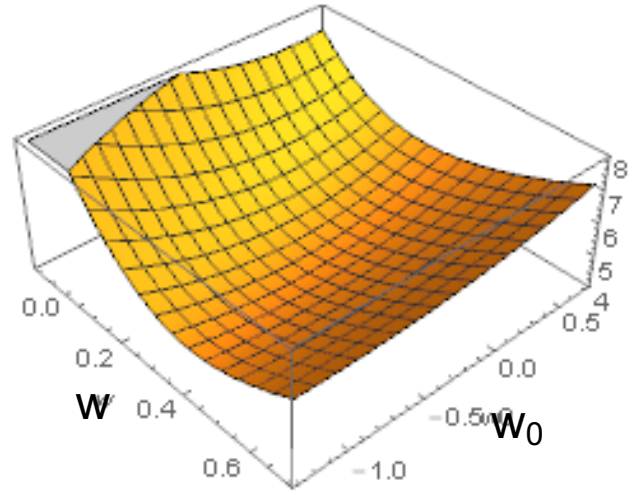


$$\sum_{i=1}^{m} [1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)]_+$$

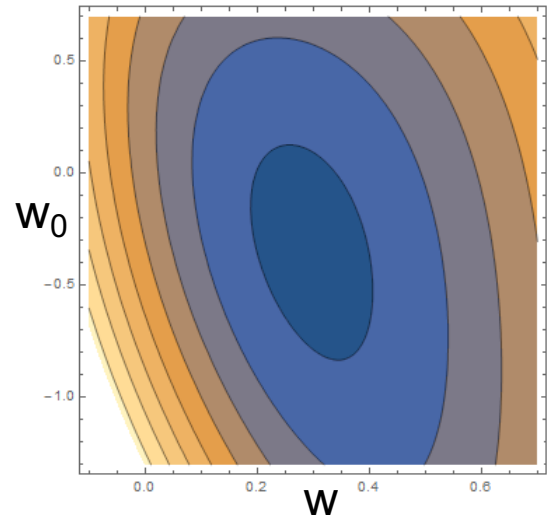


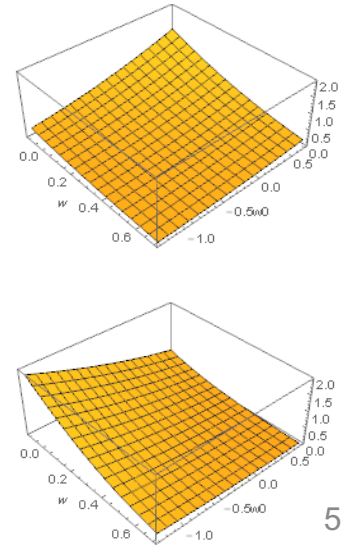


- Logistic risk



$$\sum_{i=1}^{m} \ln(1 + e^{-y_i w^T x_i})$$



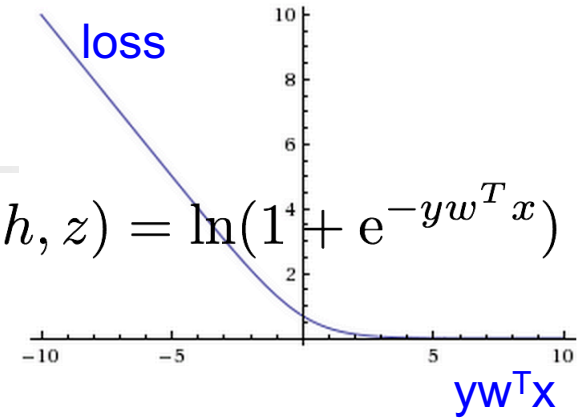


Designing a classification method

1. Define the space of possible decision boundaries
   - What will be the form of classifiers?
     - DONE: Space of all possible lines/planes/hyperplanes
2. Define the loss/risk function
   - How to evaluate the quality of a specific classifier from the space of possible classifiers?
     - DONE: cross-entropy / logistic loss
       - Or hinge loss
3. Define the method for minimizing the risk using data from the training set
   - How to reach a high-quality classifier?
     - DONE: gradient descent over the space of feature weights

- AND NOW… REGULARIZATION…

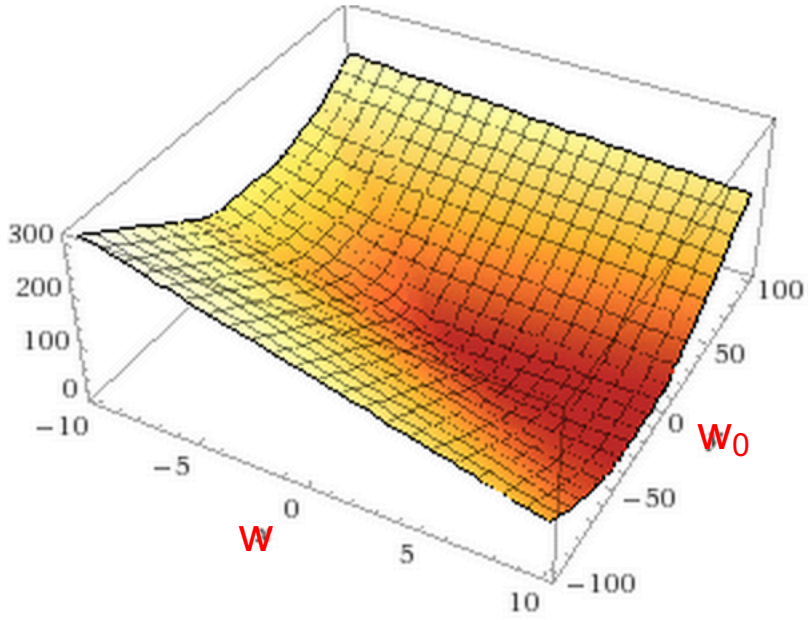
Logistic regression

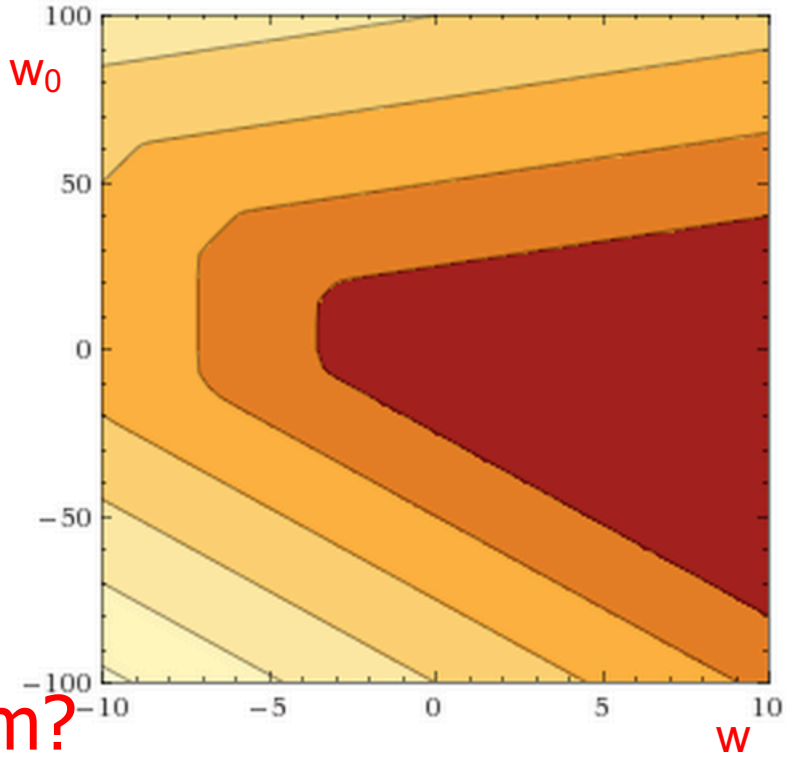


- Logistic loss - Four samples:
  - x=7, y=1
  - x=4, y=1
  - x=-1, y=-1
  - x=-2, y=-1

$$\min \frac{1}{m} \sum_{i=1}^{m} \ln(1 + e^{-y_i w^T x_i})$$

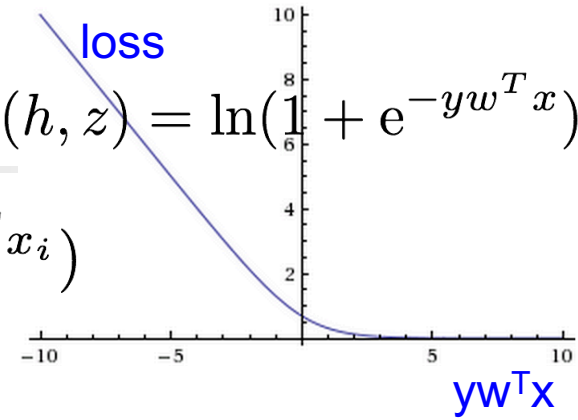
- Empirical risk for any w, w_0



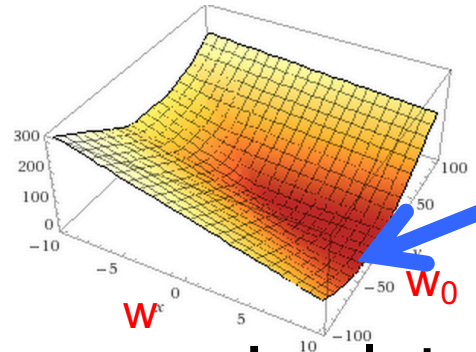


- Where's the global minimum?

Logistic regression

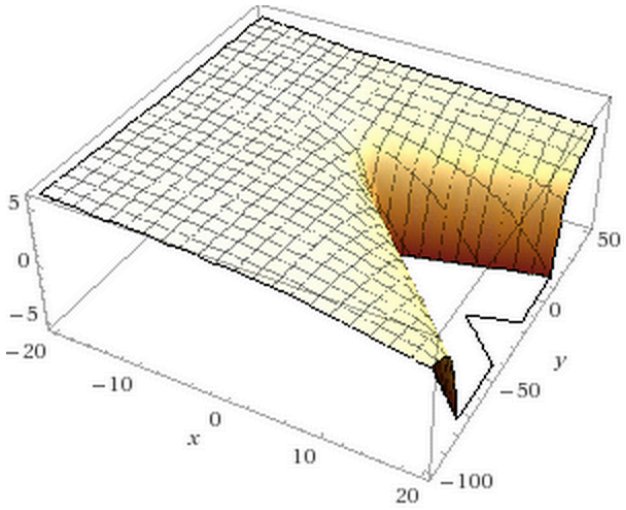


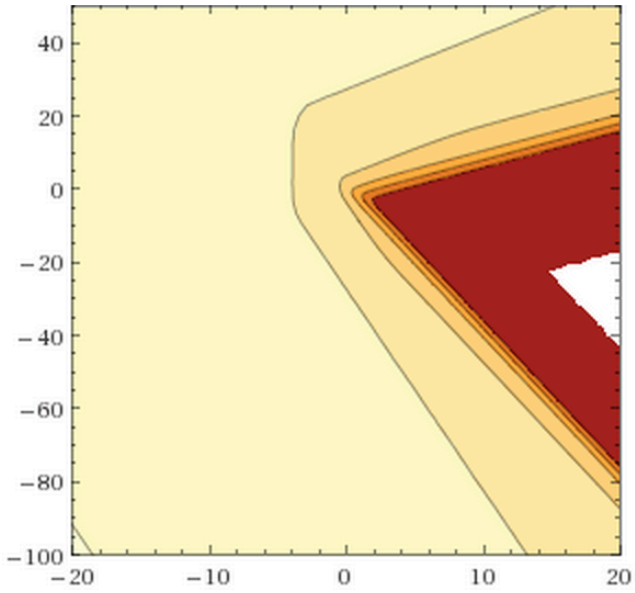
- Logistic loss $\min \frac{1}{m} \sum_{i=1}^{m} \ln(1+e^{-y_i w^T x_i})$

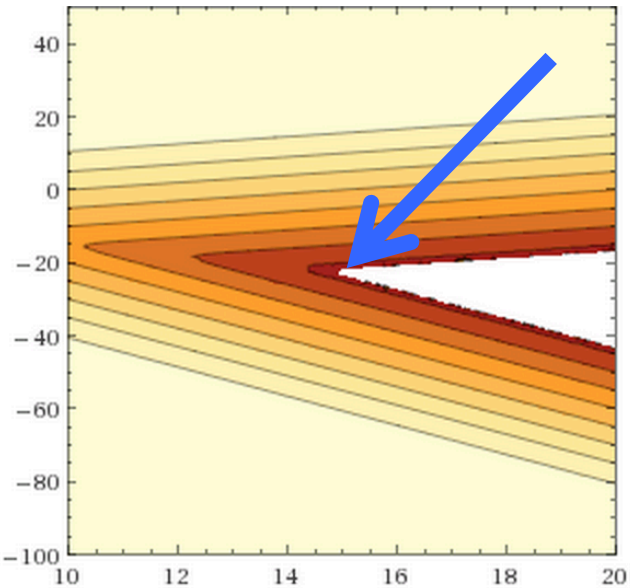


Where's the global minimum?

- log z-axis plots: empirical risk for any w, w_0



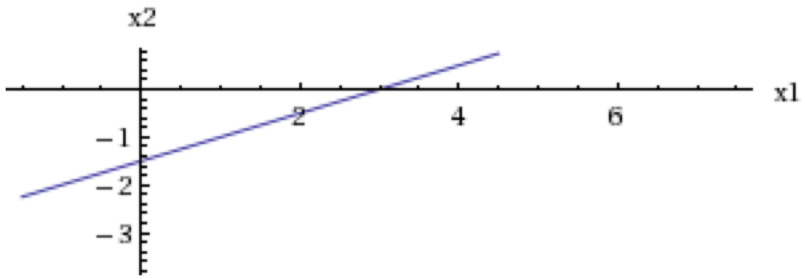


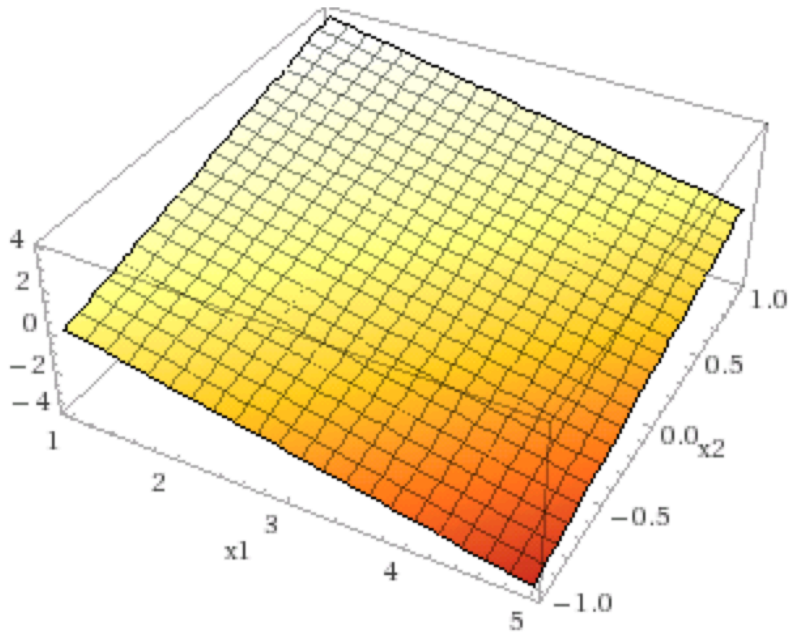


- Optimum: farther and farther: for w=15, w_0=-20 risk still >0

Problem with defining w

- Our decision boundary is $w^T x$
  - e.g 2 features, we have: $w_1x_1+w_2x_2+w_3=0$
  - Let's say the perfect separation of classes comes for $w=[-1,2,3]$ $\quad -x_1+2x_2+3=0$





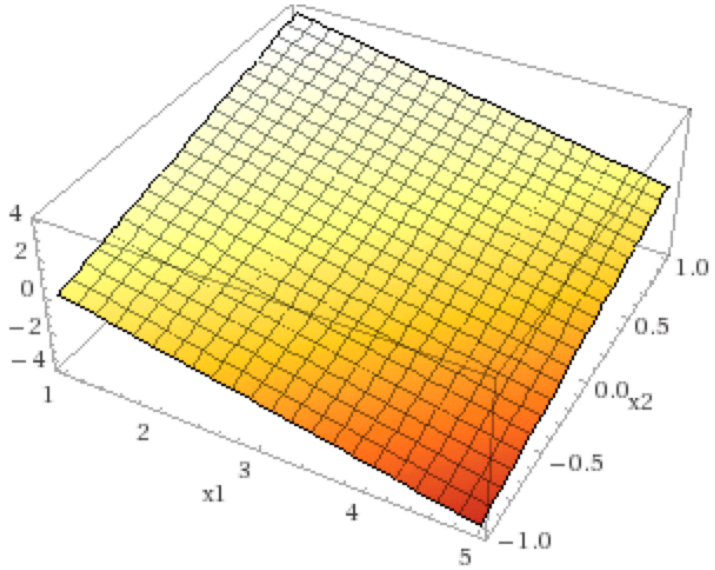
  - Is $w=[-1,2,3]$ the only vector w giving those exact same predictions?

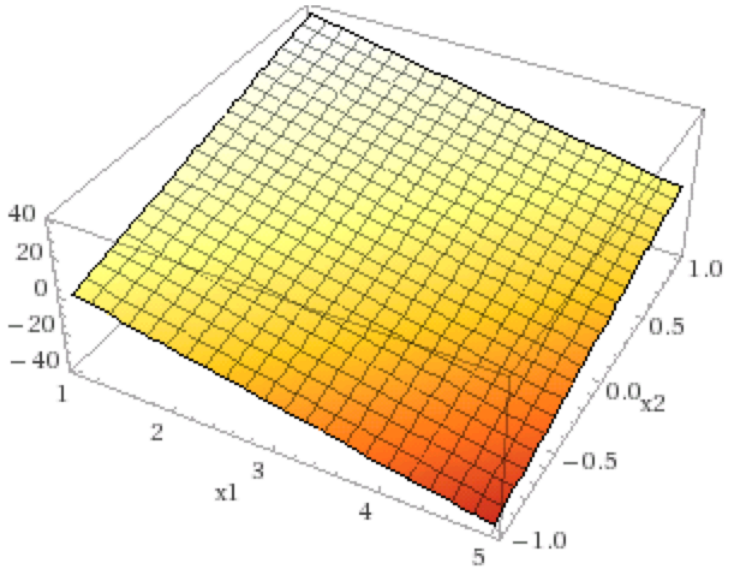
Problem with defining w

- Our decision boundary is $w^T x=0$
  - Let's say the perfect separation of classes comes for

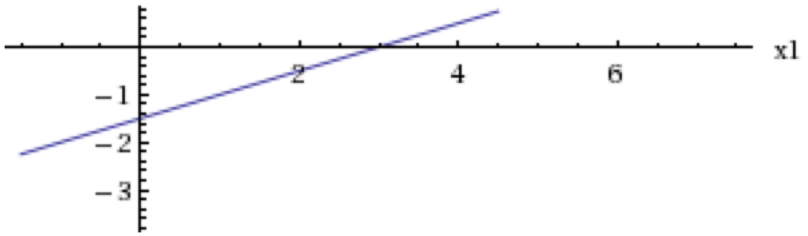
    $w=[-1,2,3]$ $\quad -x_1+2x_2+3=0$

    $w=[-10,20,30]$ $\quad -10x_1+20x_2+30=0$



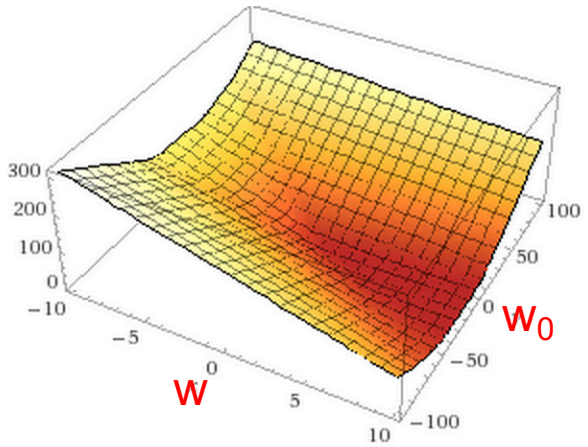


- Vector w multiplied by any positive number gives the same decision line!



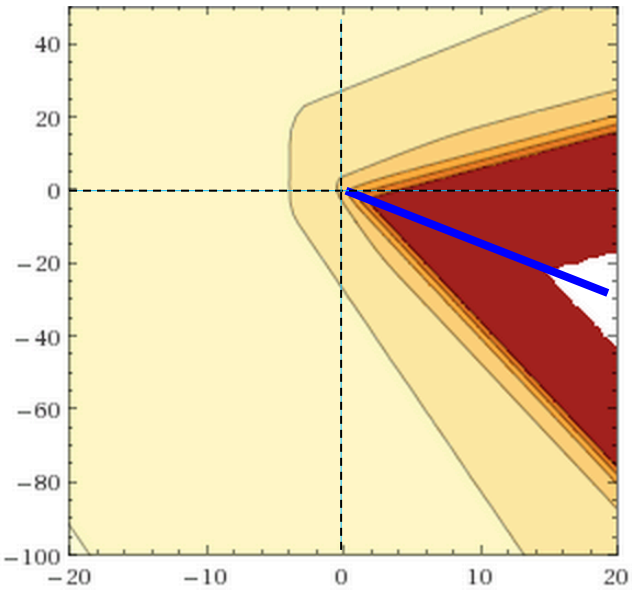
Problem with defining w

- Logistic loss: $\min \frac{1}{m} \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$



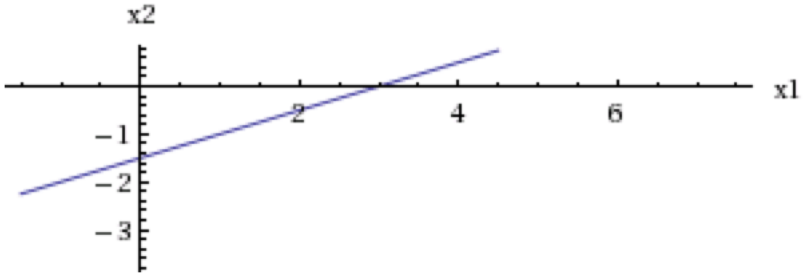
Global minimum escapes towards infinities

- Solutions w, w_0 on the blue line represent the same decision boundary!
- But different value of the empirical risk
- Do we care which one is picked?



Problem with defining w

- Our decision boundary is $w^T x=0$
  - You know somehow that perfect separation is achieved by this decision boundary:



  - Now you have options of picking w:

    w=[-1,2,3] $\quad -x_1+2x_2+3=0$

    w=[-10,20,30] $\quad -10x_1+20x_2+30=0$

    and many other

  - In an ideal world,
    it doesn't matter which one you pick

Large w affecting predictions?

- In real world, the feature values x_i we see don't represent the true feature values $\underline{x}_i$ accurately:
  - there's some measurement error Δ_i
  - x_1 = "true x_1" + Δ_1 = $\underline{x}_1 \pm \Delta_1$
  - x_2 = "true x_2" + Δ_2 = $\underline{x}_2 \pm \Delta_2$
- Say we get a sample with imperfectly measured features x_1, x_2 (say, Δ_i=±0.1). What are the decisions?
  - w_a=[-1,2,3] $\quad -x_1+2x_2+3=0$
    - $h_{small}(x)= -x_1+2x_2+3= -1(\underline{x}_1\pm\Delta_1) +2(\underline{x}_2\pm\Delta_2)+3$
      $= (-\underline{x}_1+2\underline{x}_2+3) \pm (-\Delta_1+2\Delta_2)$
      $h_{small}(x)= w_a^T\underline{x} \pm (-\Delta_1+2\Delta_2)$ => true prediction ±.3
      Δ will influence our decision
  - w_b=[-10,20,30] $\quad -10x_1+20x_2+30=0$
    - $h_{big}(x) = -10x_1+20x_2+30= -10(\underline{x}_1\pm\Delta_1) +20(\underline{x}_2\pm\Delta_2)+30$
      $= (-10\underline{x}_1+20\underline{x}_2+30) \pm 10(-\Delta_1+2\Delta_2)$
      $h_{big}(x) = w_b^T\underline{x} \pm 10(-\Delta_1+2\Delta_2) = 10\,[\, w_a^T\underline{x} \pm (-\Delta_1+2\Delta_2)\,]$
      exactly same decision! Large w => no problem !?

Large w affecting training?

- Say we got two 1D samples x^a (class +1) and x^b (class -1) with imperfectly measured feature ($\Delta=\pm 0.1$).
  - $x^a=\underline{x^a}\pm\Delta$ $x^b=\underline{x^b}\pm\Delta$ e.g: $\underline{x^a} > \underline{x^b}$
- We want to make a decision threshold in the middle between them, to get a classifier $h(x)=w^Tx+w_0$
  - $w=1$, $w_0=-\frac{1}{2}(\underline{x^a} + \underline{x^b})$ $h(x)=x - \frac{1}{2}(\underline{x^a} + \underline{x^b})$
    - If $1*x > \frac{1}{2}(\underline{x^a} + \underline{x^b})$ we predict class +1
  - $w=2$, $w_0=-(\underline{x^a} + \underline{x^b})$ $h(x)=2x - (\underline{x^a} + \underline{x^b})$
    - If $2*x > (\underline{x^a} + \underline{x^b})$ we predict class +1
- What will be the threshold?
  - $w=1$ $w_0 = \frac{1}{2}(x^a+x^b) = \frac{1}{2}(\underline{x^a}+\underline{x^b}) \pm 2\Delta$
  - $w=2$ $w_0= (\underline{x^a}+\underline{x^b}) \pm 4\Delta$

Large w affecting training?

- In real world: x_i="true x_i"+Δ_i=$\underline{x}_i$+Δ_i
- We now have two classifiers:
  - w=1 $w_0 = \frac{1}{2}(x^a+x^b) = \frac{1}{2}(\underline{x}^a+\underline{x}^b) \pm 2\Delta$
  - w=2 $w_0 = (\underline{x}^a+\underline{x}^b) \pm 4\Delta$
- A new sample x^c comes and we want to classify it:
  - $x^c=\underline{x}^c\pm\Delta^c$

- $h_{small}(x^c) = \underline{x}^c\pm\Delta^c - \frac{1}{2}(\underline{x}^a+\underline{x}^b) \pm 2\Delta$
  $= \underline{x}^c - \frac{1}{2}(\underline{x}^a+\underline{x}^b) \pm (\Delta^c+2\Delta)$
- $h_{big}(x^c) = 2\underline{x}^c\pm2\Delta^c - (\underline{x}^a+\underline{x}^b) \pm 4\Delta$
  $= 2 [\underline{x}^c - \frac{1}{2}(\underline{x}^a+\underline{x}^b) \pm (\Delta^c + 2\Delta)]$

- Same influence of error on decision, no matter what w
- Again, no problem with large w!?

Large w

- In real world: x_i="true x_i"+Δ_i=$\underline{x}_i$+Δ_i
  - Measurement = signal + noise
- No problem with large w!?
- Large w amplified the noise
- but also amplified the signal

Large w and correlated features

- In real world: x_i="true x_i"+Δ_i=$\underline{x}_i \pm \Delta$
- Let's say true classifier is: h(x)=x-t
  - Just one feature
- We saw that training and predictions are not affected if we change it to e.g. $h_{big}(x)=10x-10t$
- But now we have two copies of the feature: $\underline{x}_1=\underline{x}_2$
  - When we measure them, $x_1 = \underline{x}_1 \pm \Delta \neq \underline{x}_2 \pm \Delta = x_2$
- Instead of the simple classifier:
  - $h_{org}(x)=x_1-t$
- Training may result in:
  - $h_{small}(x)=2x_1-x_2-t$

  or:
  - $h_{big}(x)=20x_1-19x_2-t$

Do we care which of these three classifiers is given to us?

Large w and correlated features

- $x_1 = \underline{x_1} \pm \Delta \quad \neq \quad \underline{x_2} \pm \Delta = x_2$

- Instead of $h_{org}(x)=x_1-t$
- Training may result in:
  - $h_{small}(x)=2x_1-x_2-t$

  or:
  - $h_{big}(x)=20x_1-19x_2-t$

- What happens to our predictions?
  - $h_{small}(x)= 2\underline{x_1} \pm 2\Delta - \underline{x_2} \pm \Delta - t = \underline{x_1} - t \pm 3\Delta$
    - $h_{small}(x) = h_{org}(x) \pm 3\Delta$
  - $h_{big}(x)= 20\underline{x_1} \pm 20\Delta - 19\underline{x_2} \pm 19\Delta - t = \underline{x_1} - t \pm 39\Delta$
    - $h_{big}(x) = h_{org}(x) \pm 39\Delta$
- **Large w => lower accuracy!**

Large w and correlated features

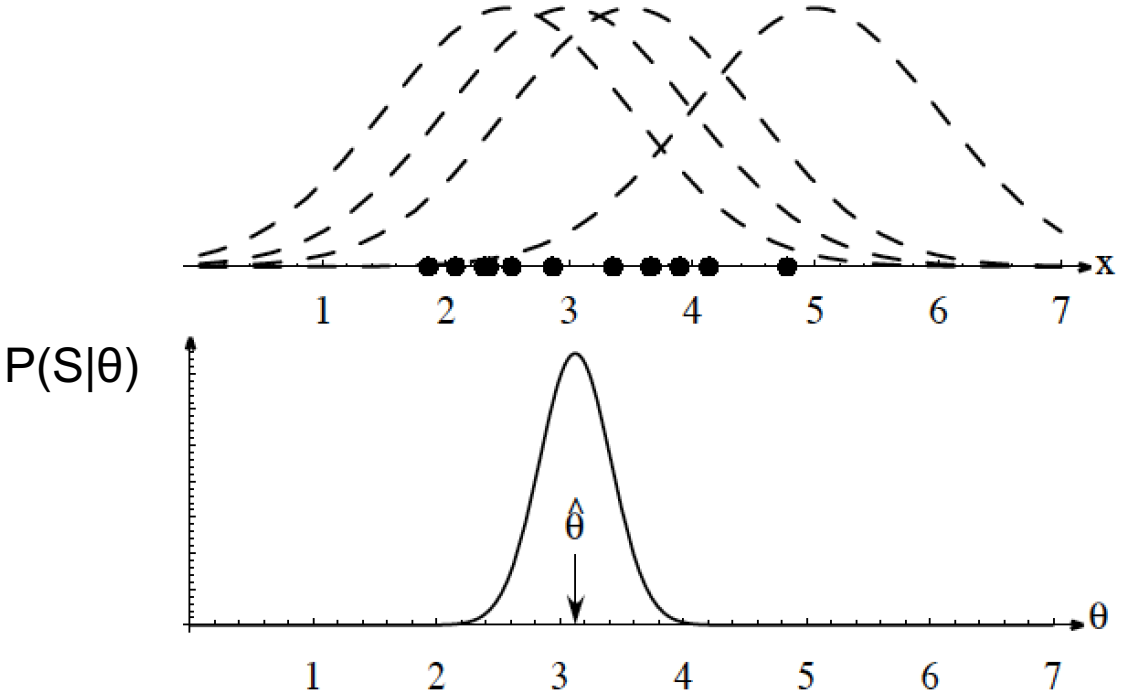
- In real world:
  - measurement = signal + noise
- In real world:
  - very often we have features that are highly correlated
- For example:
  - neighboring pixels in an image
  - expression of genes that perform some function together
- **We want to avoid large weights!**
  - **But how?**

Large w and correlated features

- **We want to avoid large weights!**
  - **But how?**
- **Let us address this problem probabilistically**
  - **We will rephrase our classification problem as an estimation problem**
  - **And then add a higher prior probability of small weights**

Maximum likelihood estimation

- Finding parameter θ that maximizes likelihood of seeing what we see in the training set S
  - Choose θ to maximize $P(S|\theta) = L(\theta \mid S) =$ *likelihood* of θ given dataset S
  - $L(\theta \mid S) = P(S|\theta) = \Pi_k P(x_k|\theta)$
- We assume θ for each class is fixed, but unknown to us
  - Some values of θ make the training set S more likely
  - Some values of θ make the training set S less likely



Simple example:
true mean of a normal distribution vs. average of some samples

We have samples S

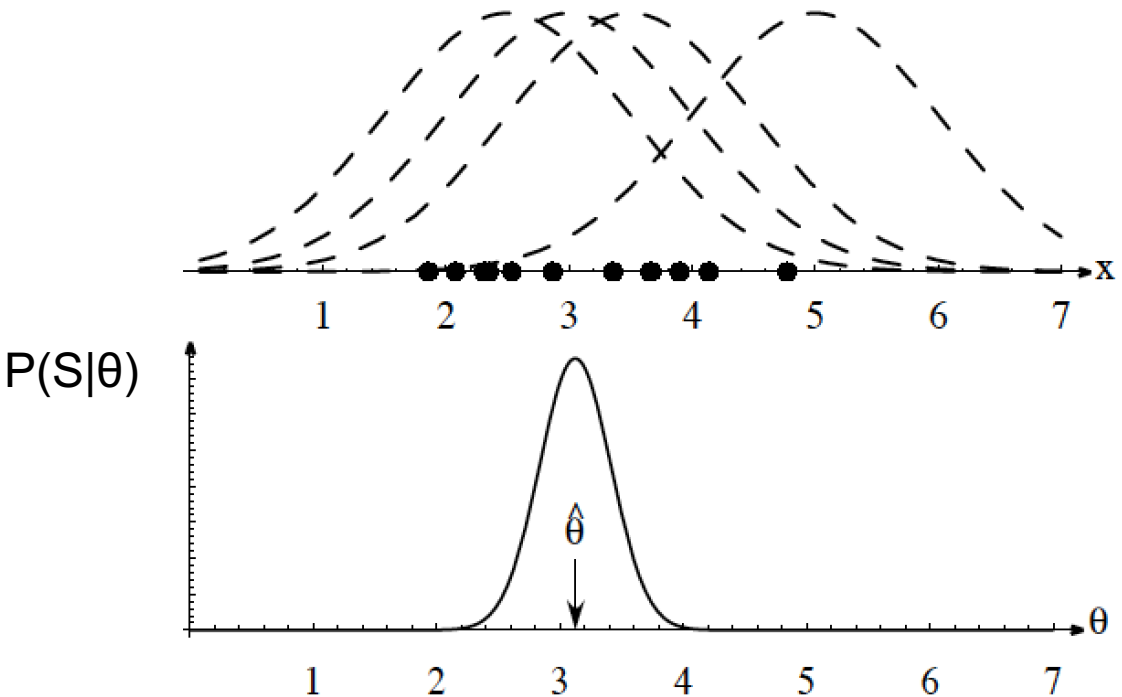
How can we estimate the mean (θ)

Try all possible means, calculate probability of seeing the samples

Pick mean with highest probability

Maximum likelihood estimation

- Finding θ that maximizes likelihood of seeing the training set S
  - Choose θ to maximize $P(S|\theta) = L(\theta \mid S) =$ *likelihood* of θ given dataset S
  - $L(\theta \mid S) = P(S|\theta) = \Pi_k P(x_k|\theta)$
- We assume θ for each class is fixed, but unknown to us
  - Some values of θ make the training set S more likely
  - Some values of θ make the training set S less likely



- How to deal with the multiplication?
- Transform it into a sum, easier to deal with mathematically!
- Same as: choose θ to maximize
  $\ln P(S|\theta) = \Sigma_k \ln P(x_k|\theta)$

Maximum likelihood estimation

- Maximize *log-likelihood*: $\ln P(S|\theta) = \Sigma_k \ln P(x_k|\theta)$
- How?
  - Math! We have the mathematical formula for the distribution P
  - Example: we have 1 feature x, we know P(x|θ) is Gaussian
    - The unknown parameter θ is a single number, the mean of the Gaussian
      - $P(x \mid \theta_i) = (2\pi\sigma^2)^{-.5} \exp(-(x-\theta_i)^2/2\sigma^2)$
      - $\ln P(x \mid \theta_i) = -.5 \ln 2\pi\sigma^2 - (x-\theta_i)^2/2\sigma^2$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

    - At maximum, derivative is 0: $d\,\Sigma_k \ln P(x_k|\theta_i) / d\,\theta_i = 0$
    - $d\,\Sigma_k [-.5 \ln 2\pi\sigma^2 - (x-\theta_i)^2/2\sigma^2] / d\,\theta_i = 0$
    - $-1/2\sigma^2 \; \Sigma_k \; d\,(x_k-\theta_i)^2 / d\,\theta_i = 0$
    - $\Sigma_{k=1,..,m} \; d\,(x_k^2 + \theta_i^2 - 2x_k\theta_i) / d\,\theta_i = 0$
    - $m2\theta_i - \Sigma_{k=1,..,m} 2x_k = 0$
    - $\theta_i = 1/m \; \Sigma_{k=1,..,m} \; x_k$ *we just derived "average" as the estimator of mean*
      - *Average is the maximum likelihood estimator of the mean (for Gaussians, at least)*

Maximum likelihood estimation

- Maximum likelihood (ML):
  - Choose θ to maximize $L(\theta|S)=P(S|\theta) = \Pi_k P(x_k,\theta)$
  - Maximize *log-likelihood*: $\ln P(S|\theta) = \Sigma_k \ln P(x_k|\theta)$
- Maximum a posteriori (MAP):
  - Finding θ that maximizes $P(\theta|S) \sim P(S|\theta)P(\theta)$
  - maximize: $P(S|\theta)P(\theta) = \Pi_k P(x_k|\theta)P(\theta)$
  - Max.: $\ln P(S|\theta)P(\theta) = \Sigma_k \ln P(x_k|\theta) + \ln P(\theta)$
- In both versions:
  - We assume θ for each class is fixed, but unknown to us
  - We find the best single estimate of θ based on how a choice of θ influences S
  - We use θ for predictions

MLE/MAP vs Bayesian

- We're predicting something (some z) based on training set S and some new information u

  Law of total probability (we can condition on "weather in Iceland", or θ):

  - $p(z \mid S, u) = \Sigma_\theta\, p(z \mid \theta, S, u)\, p(\theta \mid S, u)$

  or in fact $= \int p(z \mid \theta, S, u)\, p(\theta \mid S, u)\, d\theta$

- Assume z and S are conditionally independent given θ, i.e., if we know θ, knowing also S doesn't change our knowledge of z
  - Then:
    - $p(z \mid S, u) = \Sigma_\theta\, p(z \mid \theta, u)\, p(\theta \mid S, u)$
- Assume also that $p(\theta \mid S, u) = p(\theta \mid S)$
  - the new info u alone (without z) does not impact our knowledge of θ
  - u may impact how we use θ, but that's in $p(z \mid \theta, u)$
- End result: $p(z \mid S, u) = \Sigma_\theta\, p(z \mid \theta, u)\, p(\theta \mid S)$