Applied Machine Learning

Review 1

Isabeau Prémont-Schwarz

McGill
School of Computer Science

Model fitting

x | input features → **ML algorithm** **with parameters θ** → y | output labels

$$f(x^{(n)}; \theta)$$

the process of estimating the model parameters θ from given data $\mathcal{D}$, is the core of training ML models which often boils down into optimization of an loss function $\mathcal{L}(\theta)$

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^{N} l(y^{(n)}, f(x^{(n)}; \theta))$$

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$$

Model fitting: MLE

x | input features → **ML algorithm** (**with parameters θ**) → y | output labels

the process of estimating the model parameters θ from given data $\mathcal{D}$, is the core of training ML models which often boils down into optimization of an loss function $\mathcal{L}(\theta)$

A common approach is to use negative log probability as our loss function: $l(y, f(x,\theta)) = -\log p(y|f(x;\theta))$

Model fitting: MAP

x | input, features → **ML algorithm** (**with parameters θ**) → y | output, labels

the process of estimating the model parameters θ from given data $\mathcal{D}$, is the core of training ML models which often boils down into optimization of an loss function $\mathcal{L}(\theta)$

A common approach is to use negative log probability as our loss function:

$$l(y, f(x, \theta), \theta) = -\log p(\theta | x, y)$$

Parameter estimation

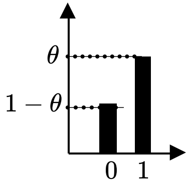
heads

$$f(,\theta) = \begin{bmatrix} \theta \\ 1-\theta \end{bmatrix}$$

tails

$$x = \emptyset$$

$$p(x|\theta) = \begin{cases} \theta & y = 1 \\ 1-\theta & y = 0 \end{cases}$$



$$\mathcal{D} = \{0, 0, 1, 1, 0, 0, 1, 0, 0, 1\}$$

θ which maximizes this is Maximum Likelihood Estimate (MLE)

Likelihood $L(\theta; \mathcal{D}) = p(\mathcal{D}|\theta) = \prod_{i \in \mathcal{D}} f(,\theta)_{y_i} = \theta^4(1-\theta)^6$

θ which maximizes this is Maximum A Posteriori (MAP)

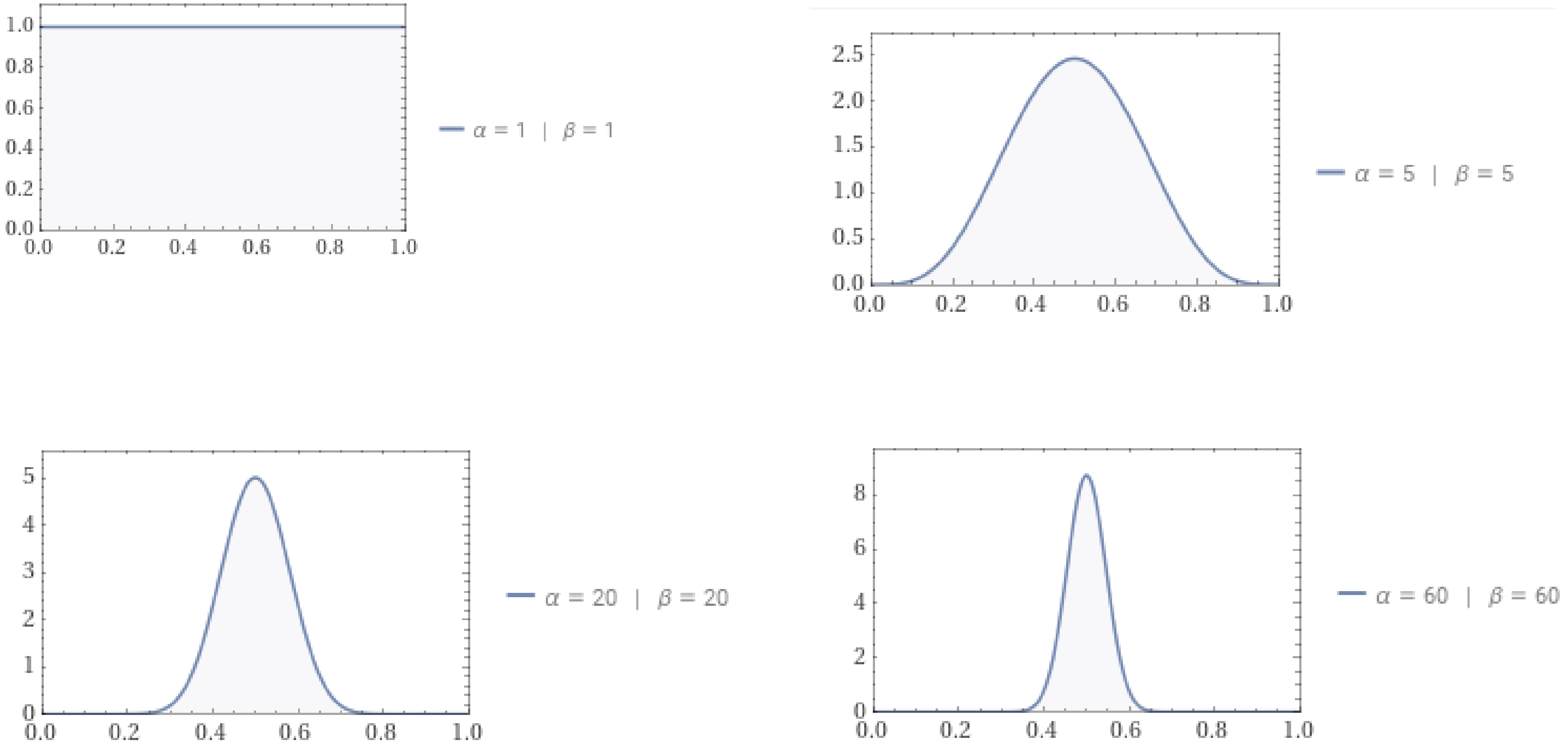
Not the same!

Posterior $p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\theta p(\mathcal{D}|\theta)p(\theta)}$

Not the same!

$$p(\text{heads}|\mathcal{D}) = p(y=1|\mathcal{D}) = \int_\theta p(\text{heads}|\theta)p(\theta|\mathcal{D})d\theta$$
$$= \int_\theta \theta p(\theta|\mathcal{D})d\theta$$

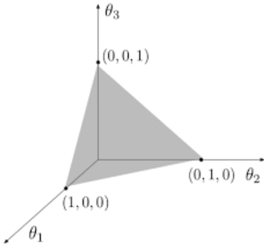
Posterior Predictive: probability of getting heads taking into account model uncertainty


α = 1 | β = 1
α = 5 | β = 5
α = 20 | β = 20
α = 60 | β = 60
0.0
0.2
0.4
0.6
0.8
1.0

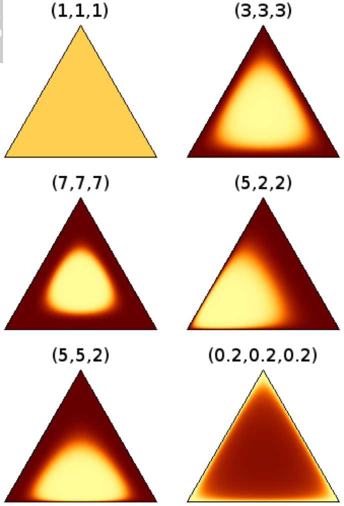
$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1 - \theta_1 - \theta_2 \end{bmatrix}$$

Dirichlet distribution

$\alpha =$



$K = 3$



$\mathrm{Dir}(\theta, [.2, .2, .2])$

is a distribution over the parameters θ of a Categorical dist.

is a generalization of Beta distribution to K categories

this should be a dist. over prob. simplex $\sum_k \theta_k = 1$

$$\mathrm{Dir}(\theta|\alpha) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \theta_k^{\alpha_k - 1}$$

normalization constant

vector of psedo-counts for K categories *(aka concentration parameters)*

$\alpha_k > 0 \; \forall k$

for $\alpha = [1, \ldots, 1]$, we get uniform distribution

for K=2, it reduces to Beta distribution

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix}$$

Maximizing the Likelihood:

prior ⟷ $p(\theta) \propto \theta_1^{25}\theta_2^{25}(1-\theta_1-\theta_2)^{25}$

$$\mathcal{D} = \{1, 1, 3\}$$

likelihood ↕

$$p(\mathcal{D}|\theta) = \theta_1^2(1-\theta_1-\theta_2)$$

$$\frac{\partial p(\mathcal{D}|\theta)}{\partial \theta_1} = 2\theta_1(1-\theta_1-\theta_2) - \theta_1^2 = 2\theta_1(1-\tfrac{3}{2}\theta_1-\theta_2)$$

$$\frac{\partial p(\mathcal{D}|\theta)}{\partial \theta_2} = -\theta_1^2$$

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1 - \theta_1 - \theta_2 \end{bmatrix}$$

Maximizing the Posterior:

prior ↔ $p(\theta) \propto \theta_1^{25}\theta_2^{25}(1-\theta_1-\theta_2)^{25}$

likelihood ↕

$$p(\mathcal{D}|\theta) = \theta_1^2(1-\theta_1-\theta_2)$$

posterior ↕

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$$\mathcal{D} = \{1, 1, 3\}$$

$$\frac{\partial p(\theta|\mathcal{D})}{\partial \theta_1} = \theta_1^{26}\theta_2^{25}(1-\theta_1-\theta_2)^{24}(27-53\theta_1-27\theta_2)$$

$$\frac{\partial p(\theta|\mathcal{D})}{\partial \theta_2} = \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{24}(25-51\theta_2-25\theta_1)$$

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix}$$

Maximizing the Posterior:

prior ↔ $p(\theta) \propto \theta_1^{25}\theta_2^{25}(1-\theta_1-\theta_2)^{25}$

likelihood ↕

$$p(\mathcal{D}|\theta) = \theta_1^2(1-\theta_1-\theta_2)$$

posterior ↕

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$$\mathcal{D} = \{1, 1, 3\}$$

$$0 = (27 - 53\theta_1 - 27\theta_2)$$

$$0 = (25 - 51\theta_2 - 25\theta_1)$$

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix}$$

Maximizing the Posterior:

prior $\longleftrightarrow$ $$p(\theta) \propto \theta_1^{25}\theta_2^{25}(1-\theta_1-\theta_2)^{25}$$

likelihood $\updownarrow$

$$p(\mathcal{D}|\theta) = \theta_1^2(1-\theta_1-\theta_2)$$

posterior $\updownarrow$

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$$\mathcal{D} = \{1, 1, 3\}$$

$$0 = (27 - 53\theta_1 - 27\theta_2)$$

$$\theta_1 = 1 - \tfrac{51}{25}\theta_2$$

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix}$$

Maximizing the Posterior:

prior ⟷ $p(\theta) \propto \theta_1^{25}\theta_2^{25}(1-\theta_1-\theta_2)^{25}$

likelihood

↕

$$p(\mathcal{D}|\theta) = \theta_1^2(1-\theta_1-\theta_2)$$

posterior

↕

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$$\mathcal{D} = \{1, 1, 3\}$$

$$0 = (27 - 53(1 - \tfrac{51}{25}\theta_2) - 27\theta_2) = -26 + 81.12\theta_2$$

$$\theta_1 = 1 - \tfrac{51}{25}\theta_2$$

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix}$$

Maximizing the Posterior:

prior $\leftrightarrow$ $$p(\theta) \propto \theta_1^{25}\theta_2^{25}(1-\theta_1-\theta_2)^{25}$$

likelihood $\updownarrow$

$$p(\mathcal{D}|\theta) = \theta_1^2(1-\theta_1-\theta_2)$$

posterior $\updownarrow$

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$$\mathcal{D} = \{1, 1, 3\}$$

$$0 = (27 - 53(1 - 51/25\theta_2) - 27\theta_2) = -26 + 81.12\theta_2$$

$$\theta_1 = 1 - 51/25\ \theta_2$$

$$f(,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix} = \begin{bmatrix} 27/78 \\ 25/78 \\ 26/78 \end{bmatrix} = \begin{bmatrix} 0.346 \\ 0.321 \\ 0.333 \end{bmatrix}$$

$$f(\,,\theta) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ 1-\theta_1-\theta_2 \end{bmatrix}$$

Finding the expected values of θ:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto \theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$$p(\theta|\mathcal{D}) = \frac{\Gamma(28+26+27)}{\Gamma(28)\Gamma(26)\Gamma(27)}\theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}$$

$p\,($  $= 1|\mathcal{D}) = \mathbb{E}\left[\theta_1\right] = \int_\theta p($  $= 1|\theta)p(\theta|\mathcal{D})$

$p\,($  $= 1|\mathcal{D}) = \int_0^1\int_0^1 \theta_1 \frac{\Gamma(28+26+27)}{\Gamma(28)\Gamma(26)\Gamma(27)}\theta_1^{27}\theta_2^{25}(1-\theta_1-\theta_2)^{26}d\theta_1 d\theta_2$

$$= \frac{\Gamma(28+26+27)}{\Gamma(28)\Gamma(26)\Gamma(27)}\frac{\Gamma(29)\Gamma(26)\Gamma(27)}{\Gamma(29+26+27)} = \frac{28}{81} = 0.345\ldots$$

Linear model of regression

x | input features → **ML algorithm** **with parameters w** $f(x; w)$ → y | output labels

$$f_w(x) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

model parameters or weights (we also called them θ before)

$[w_0, w_1, \ldots w_D]$

bias or intercept

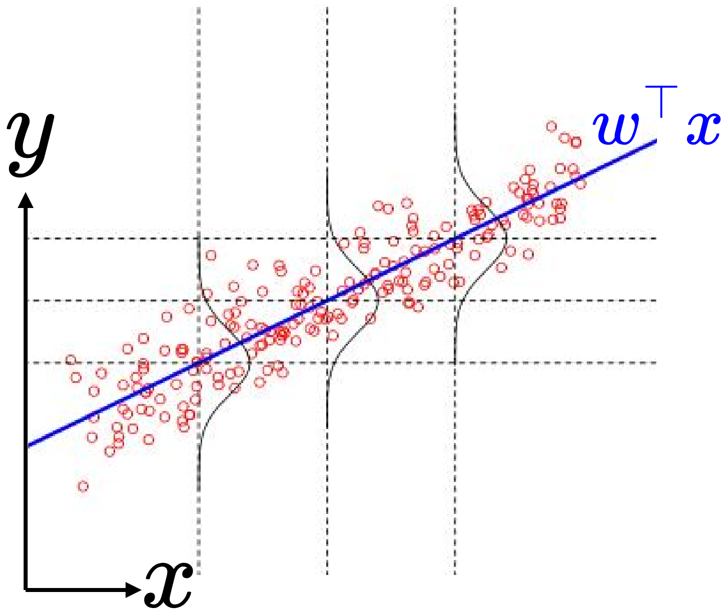
assuming a scalar output $f_w : \mathbb{R}^D \rightarrow \mathbb{R}$

will generalize to a vector later

Probailistic interpretation

idea given the dataset $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$

learn a probabilistic model $p(y|x;w)$



consider $p(y|x;w)$ with the following form

$$p_w(y \mid x) = \mathcal{N}(y \mid w^\top x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^\top x)^2}{2\sigma^2}}$$

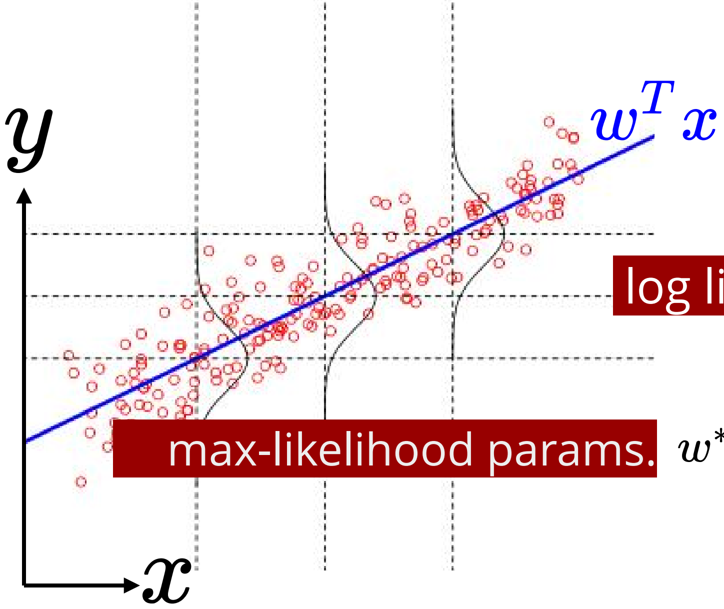
assume a fixed variance, say $\sigma^2 = 1$

Q: how to fit the model?

A: maximize the conditional likelihood!

Maximum likelihood & linear regression

cond. probability $p(y \mid x; w) = \mathcal{N}(y \mid w^\top x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-w^\top x)^2}{2\sigma^2}}$



likelihood $L(w) = \prod_{n=1}^{N} p(y^{(n)} \mid x^{(n)}; w)$

log likelihood $\ell(w) = \sum_n -\frac{1}{2\sigma^2}(y^{(n)} - w^\top x^{(n)})^2 + \text{constants}$

max-likelihood params. $w^* = \arg\max_w \ell(w) = \arg\min_w \frac{1}{2}\sum_n (y^{(n)} - w^\top x^{(n)})^2$

linear least squares!

image from here

whenever we use square loss, we are assuming Gaussian noise!

Logistic function

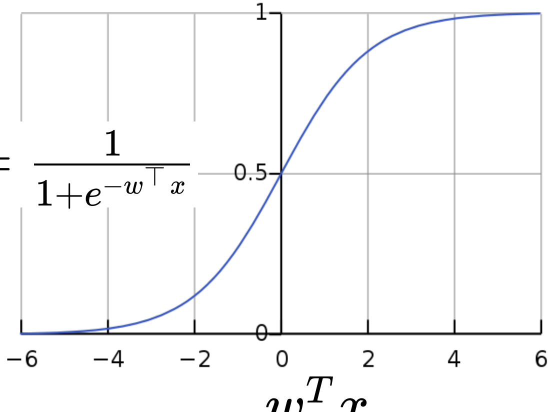
$$p(y=1|x,\omega)/p(y=0|x,\omega) = \exp(\omega^T x)$$

desirable property of $\sigma : \mathbb{R} \to \mathbb{R},\ \sigma(x) := \frac{1}{1+\exp(-x)}$

- all $w^\top x > 0$ are squashed close together
- all $w^\top x < 0$ are squashed together

logistic function (aka The Sigmoid) has these properties

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



the decision boundary is

$$w^\top x = 0 \Leftrightarrow \sigma(w^\top x) = \frac{1}{2}$$

still a linear decision boundary

Logistic regression: The Loss

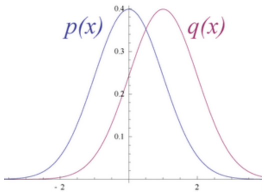
Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$p(y^{(i)}|x^{(i)},\omega) = \begin{cases} \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 1 \\ 1 - \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 0 \end{cases} = \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1 - \sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

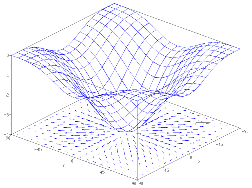
$$L(\mathcal{D},\omega) = p(\mathcal{D}|\omega) = \prod_{x^i, y^i \in \mathcal{D}} \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1 - \sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

$$J(\omega) = -\log\left(L(\mathcal{D},\omega)\right) = -\sum_{x^i, y^i \in \mathcal{D}} y^{(i)} \log\left(\sigma(\omega^T x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - \sigma(\omega^T x^{(i)})\right)$$

Cross-Entropy: $\sum_k -p_k \log(q_k)$



Gradient



how did we find the optimal weights?

(in contrast to linear regression, no closed form solution)

cost: $J(w) = \sum_{n=1}^{N} y^{(n)} \log\left(1 + e^{-w^\top x^{(n)}}\right) + (1 - y^{(n)}) \log\left(1 + e^{w^\top x^{(n)}}\right)$

taking partial derivative $\frac{\partial}{\partial w_d} J(w) = \sum_n -y^{(n)} x_d^{(n)} \frac{e^{-w^\top x^{(n)}}}{1+e^{-w^\top x^{(n)}}} + x_d^{(n)} (1 - y^{(n)}) \frac{e^{w^\top x^{(n)}}}{1+e^{w^\top x^{(n)}}}$

$= \sum_n -x_d^{(n)} y^{(n)} (1 - \hat{y}^{(n)}) + x_d^{(n)} (1 - y^{(n)}) \hat{y}^{(n)} = \sum_n x_d^{(n)} (\hat{y}^{(n)} - y^{(n)})$

gradient $\nabla J(w) = \sum_n x^{(n)} (\hat{y}^{(n)} - y^{(n)})$

$\hat{y}^{(n)} = \sigma(w^\top x^{(n)})$

compare to gradient for linear regression $\nabla J(w) = \sum_n x^{(n)} (\hat{y}^{(n)} - y^{(n)})$

$\hat{y}^{(n)} = w^\top x^{(n)}$

Softmax

generalization of logistic to > 2 classes:

- **logistic**: $\sigma : \mathbb{R} \to (0, 1)$ produces a single probability
  - probability of the second class is $(1 - \sigma(z))$

- **softmax:** $\mathbb{R}^C \to \Delta_C$ recall: probability simplex $p \in \Delta_c \to \sum_{c=1}^{C} p_c = 1$

$$\hat{y}_c = \text{softmax}(z)_c = \frac{e^{z_c}}{\sum_{c'=1}^{C} e^{z_{c'}}} \quad \text{so} \quad \sum_c \hat{y} = 1$$

example $\text{softmax}([1, 1, 2, 0]) = [\frac{e}{2e+e^2+1}, \frac{e}{2e+e^2+1}, \frac{e^2}{2e+e^2+1}, \frac{1}{2e+e^2+1}]$

$\text{softmax}([10, 100, -1]) \approx [0, 1, 0]$

if input values are large, softmax becomes similar to argmax

similar to logistic this is also a squashing function

Implementing the cost function

softmax cross entropy cost function is the negative of the log-likelihood

similar to the binary case

$$J(\{w_c\}) = -\left(\sum_{n=1}^{N} (y^{(n)})^\top z^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}} \right))$$

naive implementation of log-sum-exp causes over/underflow

we could run into very large or small numbers inside the exponential

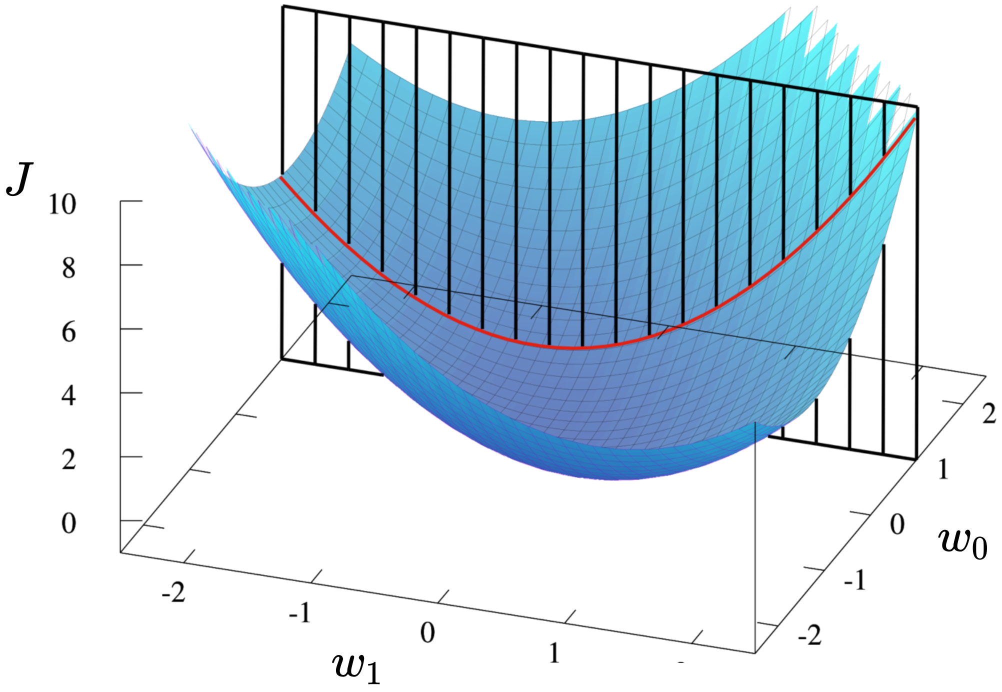
prevent this using this one trick!

$$\log \sum_c e^{z_c} = \bar{z} + \log \sum_c e^{z_c - \bar{z}}$$

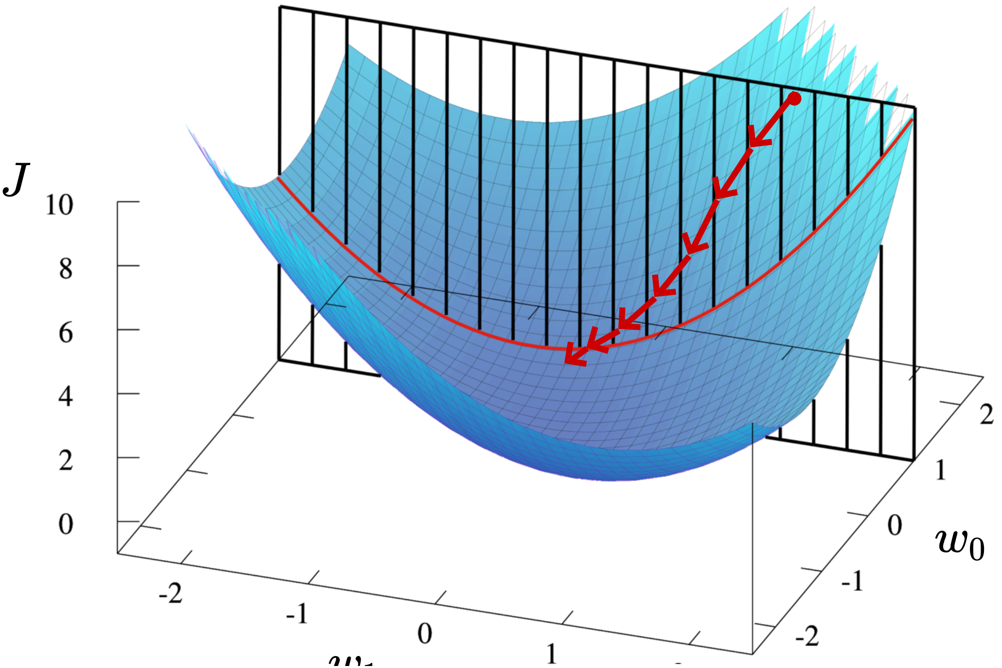
where $\bar{z} \leftarrow \max_c z_c$

this bring the numbers in exponent close to zero and makes the log-sum-exp numerically stable

Gradient descent



Gradient descent



$$w^{\{t+1\}} \leftarrow w^{\{t\}} - \alpha \nabla J(w^{\{t\}})$$

learning rate

$$\nabla J(w) = [\frac{\partial}{\partial w_1} J(w), \cdots \frac{\partial}{\partial w_D} J(w)]^T$$

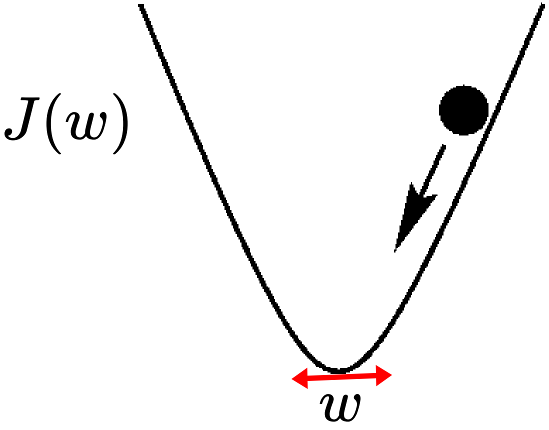
Minimum of a convex function

Convex functions are easier to minimize:

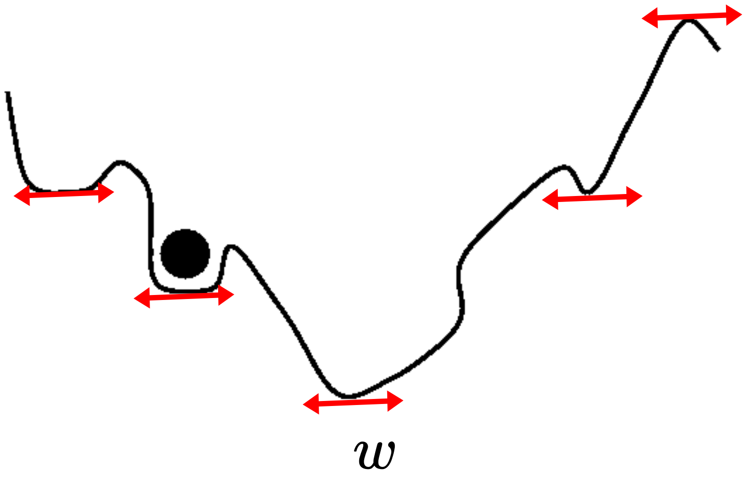
- critical points are global minimum
- gradient descent can find it

$$w^{\{t+1\}} \leftarrow w^{\{t\}} - \alpha \nabla J(w^{\{t\}})$$

convex



non-convex: gradient descent may find a local optima



(Batch) Gradient Descent

use all training data to produce gradient estimates

$$\nabla J = \frac{1}{|\mathcal{D}|} \sum_{n \in \mathcal{D}} \nabla J_n(w)$$

$\mathcal{D}$ the whole training dataset

(Mini Batch) Stochastic Gradient Descent

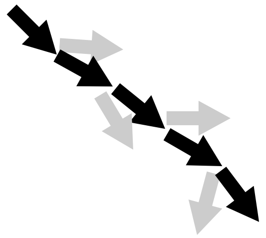
use a minibatch to produce gradient estimates

$$\nabla J_{\mathbb{B}} = \frac{1}{|\mathbb{B}|} \sum_{n \in \mathbb{B}} \nabla J_n(w)$$

$\mathbb{B} \subseteq \{1, \ldots, N\}$ a subset of the dataset

HOW TO SAMPLE THE MINIBATCHES??

Momentum



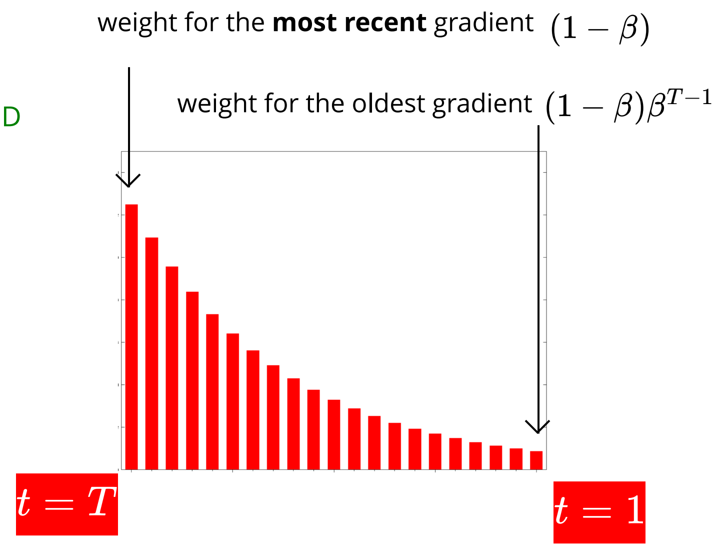
to help with oscillations:

- use a **running average** of gradients
- more recent gradients should have higher weights

$$\Delta w^{\{t\}} \leftarrow \beta \Delta w^{\{t-1\}} + (1-\beta)\nabla J_{\mathbb{B}}(w^{\{t-1\}})$$

$$w^{\{t\}} \leftarrow w^{\{t-1\}} - \alpha \Delta w^{\{t\}}$$

momentum of 0 reduces to SGD
common value > .9



is effectively an exponential moving average

$$\Delta w^{\{T\}} = \sum_{t=1}^{T} \beta^{T-t}(1-\beta)\nabla J_{\mathbb{B}}(w^{\{t\}})$$

there are other variations of momentum with similar idea

Example

(Adaptive Step Size) RMSprop

$$S^{\{t\}} \leftarrow \gamma S^{\{t-1\}} + (1-\gamma)\left(\nabla J(w^{\{t-1\}})\right)^2$$

$$w^{\{t\}} \leftarrow w_{\{t-1\}} - \frac{\alpha}{\sqrt{S^{\{t\}} + \epsilon}} \nabla J(w^{\{t-1\}})$$

note that here is a vector and with the square root is element-wise

Regularization and Maximum a Posteriori (MAP)

can we do Bayesian inference instead of maximum likelihood?

$$p(w|y, X) \propto p(w)p(y|w, X)$$

posterior prior likelihood

in general, this is expensive, but there's a cheap compromise:

MAP estimate $w^{MAP} = \arg\max_w p(w)p(y|X, w)$

$$= \arg\max_w \log p(y|X, w) + \log p(w)$$

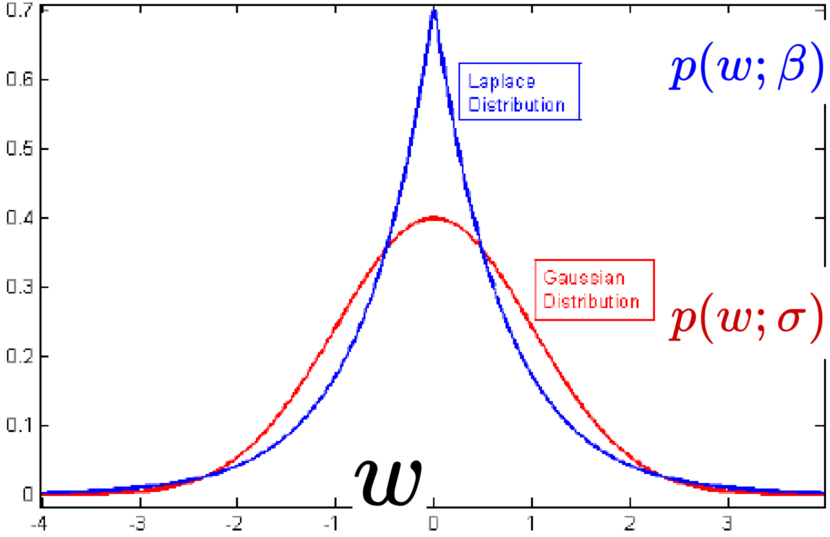
likelihood: original objective prior

all that is changing is the additional penalty on w

Gaussian or Laplace prior

Gaussian Prior: L2 regularization: $J(w) \leftarrow J(w) + \lambda||w||_2^2$

Laplace Prior: L1 regularization: $J(w) \leftarrow J(w) + \lambda||w||_1$



$$p(w;\beta) = \frac{1}{2\beta} e^{-\frac{|w|}{\beta}}$$

$$p(w;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{w^2}{2\sigma^2}}$$

image from here

$$\mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - y)^2] = \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - y + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]$$

$f(x) + \epsilon$

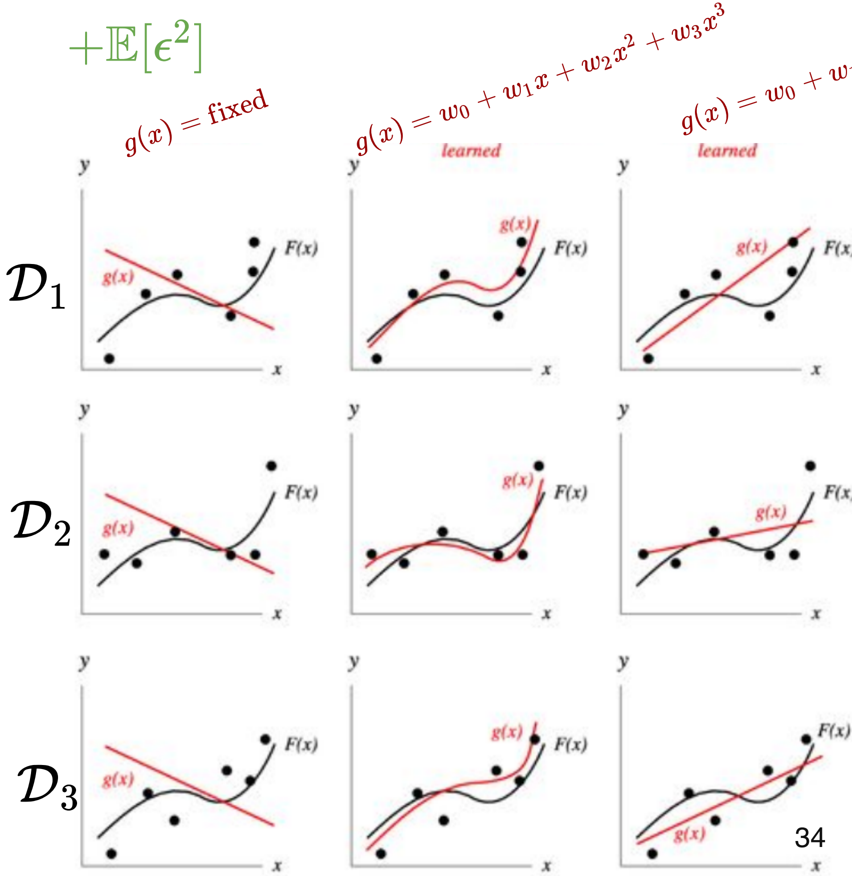
$\hat{f}_{\mathcal{D}}(x) + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]$ add and subtract a term

unavoidable noise error

$$= \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[\epsilon^2]$$

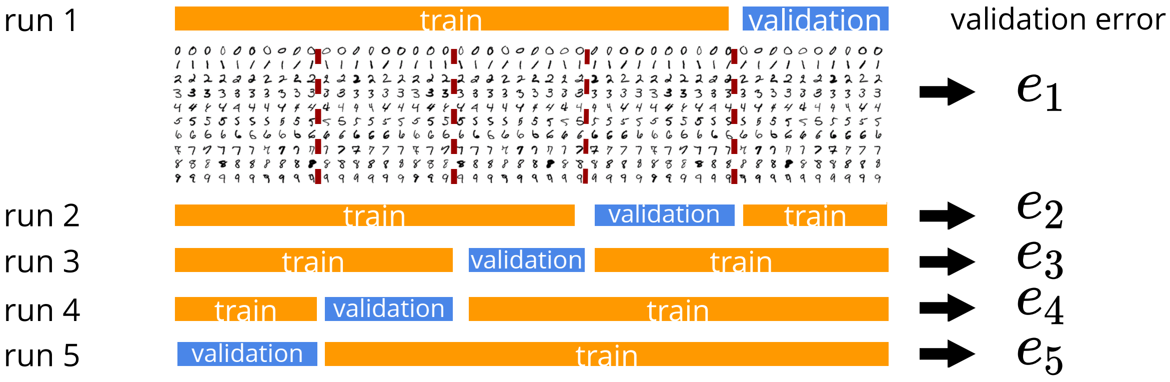
variance

bias^2



Cross validation

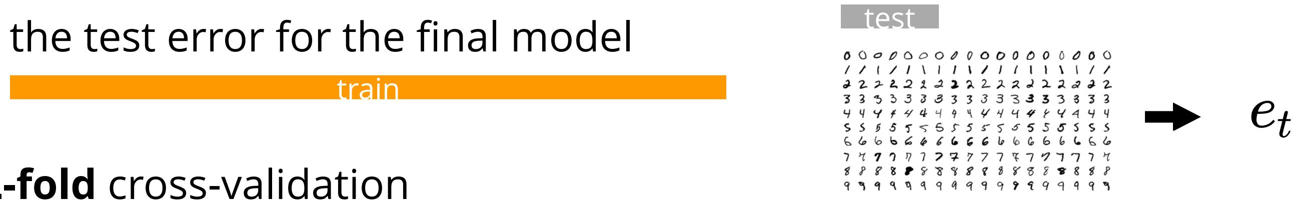
- divide the (training + validation) data into L parts
- use one part for validation and L-1 for training



- use the **average** validation error and its variance (uncertainty) to pick the best model

$$\bar{e} = \frac{1}{5}\sum_{i=1}^{5} e_i$$

- report the test error for the final model



this is called **L-fold** cross-validation

in **leave-one-out** cross-validation L=N (only one instance is used for validation)

Cross validation